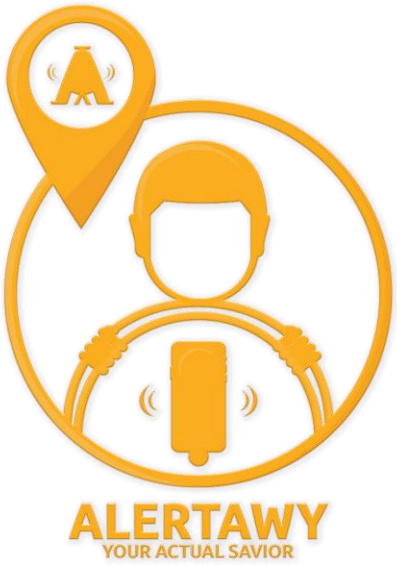
HELWAN UNIVERSITY

FACULTY OF COMPUTERS AND ARTIFICIAL INTELLIGENCE

COMPUTER SCIENCE DEPARTMENT



AI-Powered Driver Monitoring System

A graduation project dissertation by

Abdulrahman Khalid

Shreen Muhammad

Abdulkareem Anwar

Sohaila Hassan

Abdulrahman Emad

Islam Reda

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computers and Artificial Intelligence, at the Computer Science Department, Faculty of Computers and Artificial Intelligence, Helwan University

Supervised by Dr. Amal Abou-Tabl

June 2023

Abstract

Road accidents caused by drowsy and distracted driving pose a significant threat, leading to a high number of fatalities and devastating consequences for families. In response, this project presents Alertawy, an integrated AI-Powered monitor system comprising a mobile application and a web application. The mobile application leverages the power of AI to detect driver distractions, drowsiness, and lack of focus on the road. It provides timely alerts to drivers who are using their phones, experiencing drowsiness, or displaying signs of distraction. Additionally, Alertawy assists drivers in evading radar detection and exceeding speed limits. Using computer vision techniques and deep learning models, the system predicts the driver's current state, ensuring privacy by performing all inferences on the device itself.

The web application serves as an administration platform, enabling transportation companies to effectively monitor driver reports and evaluate their performance. By analyzing detailed ride data and generating insightful reports, the web application empowers companies to proactively address potential issues and ensure a higher level of driver safety. The project aims to provide proof of concept for an AI-Powered Driver Monitoring System, with the overarching objective of reducing daily accidents caused by driver inattentiveness, drowsiness, or distractions. By monitoring and analyzing the drivers' states while they are on the road, Alertawy contributes to the creation of safer driving environments for all.

Keywords

Driver Monitoring System, Distraction Detection, Gaze Estimation, Drowsiness Detection, Radars Avoidance

Acknowledgement

We would like to express our sincere gratitude to our Supervisor, Dr. Amal Abou-Table, for their invaluable guidance, support, and expertise throughout the project, also we would like to thank Dr. Yehya Abou-Elnaga for generously granting us a license to access the distraction dataset from the Machine Intelligence group at the American University in Cairo (MI-AUC).

Furthermore, we would like to express our appreciation to the developers behind:

1. *Google Collaboratory*
2. *Flutter Framework*
3. *Firebase Firetore*
4. *Google Maps API*

The contributions of these open tools have been instrumental in the development and execution of our project. We deeply appreciate the efforts and innovations made by their creators, as these tools have substantially enhanced the quality and functionality of our work.

Table of Contents

Contents

Chapter1: Introduction

1.1 Overview

Addressing the Critical Issues of Drowsiness and Distraction in Driving

According to the American National Highway Traffic Safety Administration (NHTSA) report on May 17, 2022, the alarming rise in drowsy and distracted driving has resulted in a devastating toll on human lives. Last year alone, there were 42,915 fatalities in motor vehicle traffic crashes, representing a 10.5% increase from the previous year's 38,824 fatalities. These numbers mark the highest recorded fatalities since 2005, and the annual percentage increase is the largest in the history of the Fatality Analysis Reporting System. Behind these statistics are the tragic loss of lives and the profound impact on the families left behind.

Drowsiness in driving refers to the difficulty of staying awake, even during activities. It is closely linked to the natural sleep-wake cycle, known as the circadian rhythm, and is not directly related to the task at hand. Throughout a 24-hour cycle, the human body tends to experience a higher propensity for sleep during specific times, particularly between midnight and approximately 6 a.m. During these hours, the body naturally gravitates towards sleep, leading to a decrease in vigilance. Currently, there is no standardized tool to measure vigilance levels, making it necessary to observe signs of hypervigilance exhibited by the driver and analyze them. These signs can be categorized into behavioral and physiological indicators.

Behavioral signs include:

- Slowness of reaction
- Inattention to the environment (road signs, obstacles, pedestrians, etc.)
- Coordination errors
- Inability to maintain a steady speed or trajectory.

Physiological signs manifest as abnormal expressions, primarily on the driver's face, and include:

- Excessive blinking
- Stiff neck or back pain
- Frequent yawning
- Difficulty in keeping the eyes open and head in an upright position.
- Periods of micro-sleep

When any of these signs become evident, it is crucial for the driver to take a break of at least 15 minutes before resuming driving. Unfortunately, drivers often overestimate their level of alertness and tend to ignore these warning signs, leading to potentially dangerous situations on the road.

Distraction, on the other hand, refers to the challenge of maintaining focus while engaged in driving. It is associated with physical actions performed by the driver that divert their attention from the task of driving, consequently leading to accidents. Common distractions include:

- Cell phone use
- Conversations with passengers
- Diverted attention from the road.

Addressing the issues of drowsiness and distraction in driving requires innovative solutions that prioritize safety and promote responsible driving practices. Our project aims to develop an AI-Powered driver monitoring system that tackles these challenges head-on, offering real-time alerts and proactive measures to enhance driver safety. By leveraging advanced technologies, we aspire to create a safer driving environment and save lives on the roads.

1.2 Problem Statement

Addressing the Consequences of Drowsy and Distracted Driving

The prevailing problem we aim to tackle is the widespread issue of drowsy and distracted driving, which has grave consequences, including a high number of accidents, tragedies, and loss of precious lives. As mentioned earlier in the overview section, the American National Highway Traffic Safety Administration (NHTSA) reported a staggering increase in fatalities resulting from drowsy and distracted driving, reaching a record high in recent years.

Drowsiness poses a significant risk to road safety as drivers struggle to stay awake and alert behind the wheel. The natural sleep-wake cycle, known as the circadian rhythm, influences the body's tendency to become drowsy at certain times, particularly during late-night hours. However, the absence of standardized tools to measure vigilance levels makes it challenging to identify drowsiness accurately. Consequently, drivers may underestimate their fatigue levels, leading to reduced reaction times, inattention to the road, coordination errors, and compromised driving performance.

In addition to drowsiness, driver distraction presents another pressing concern. With the increasing use of smartphones and other technological devices, drivers frequently engage in activities that divert their attention away from the primary task of driving. Phone usage, conversations with passengers, and distractions from the road environment contribute to reduced focus and impaired driving abilities. These distractions have been proven to significantly increase the risk of accidents and jeopardize the safety of all road users.

It is imperative to address these critical issues to create a safer driving environment and mitigate the devastating consequences of drowsy and distracted driving. By developing an AI-Powered driver monitoring system, we aim to provide an effective solution that enhances driver safety, reduces accidents, and ultimately saves lives.

1.3 Scope and Objectives

Enhancing Driver Safety through AI-Powered Monitoring System.

The scope of our project is to address the critical issue of accidents caused by driver distractions, drowsiness, and lack of attention to the road. Our ultimate objective is to make driving safer worldwide through the implementation of an AI-Powered driver monitoring system.

Our project aims to achieve the following goals:

1. Proof of Concept and Functional Implementation: We will propose and develop a robust proof of concept that demonstrates the effectiveness of AI-Powered driver monitoring in mitigating the risks associated with drowsy or distracted driving.
2. Integrated System Components: Our solution consists of two interconnected components: a mobile application for drivers and a

web application for administration. This integrated system ensures comprehensive monitoring and management of driver behavior.

The key features of our mobile application, Alertawy, include:

- Real-Time Alerts: Alertawy utilizes advanced Deep Learning Techniques to detect and alert drivers if they engage in risky behaviors such as talking on the phone, experiencing drowsiness, or getting distracted from the road.
- Radar Avoidance Assistance: By monitoring driving behavior, Alertawy helps drivers avoid radar detection by providing timely alerts when they engage in unsafe driving practices or exceed speed limits.
- Localized AI Processing: We prioritize privacy by employing computer vision methods and deep learning models that perform state prediction locally on the driver's device, ensuring that sensitive data remains secure.

The web application serves as the administration platform, providing administrators with a wider view of drivers' rides and reports. It facilitates comprehensive evaluation and analysis, empowering administrators to make informed decisions and take proactive measures to enhance driver safety.

Our project strives to deliver a solution that effectively combats driver distractions and drowsiness, ultimately reducing the number of accidents and making roads safer for everyone. By integrating advanced technologies and ensuring privacy, we aim to revolutionize driver monitoring and set new standards for road safety.

1.4 Work Methodology

Agile Scrum Framework with Jira Software Platform

Our project is managed using the Agile methodology, specifically the Scrum framework, supported by the Jira software platform. Scrum enables us to break down big and complex projects into manageable iterations called sprints. As Megan Cook, Group Product Manager for Jira Software at Atlassian, explains, "With Scrum, a product is built in a series of sprints that break down big, complex projects into bite-sized pieces."

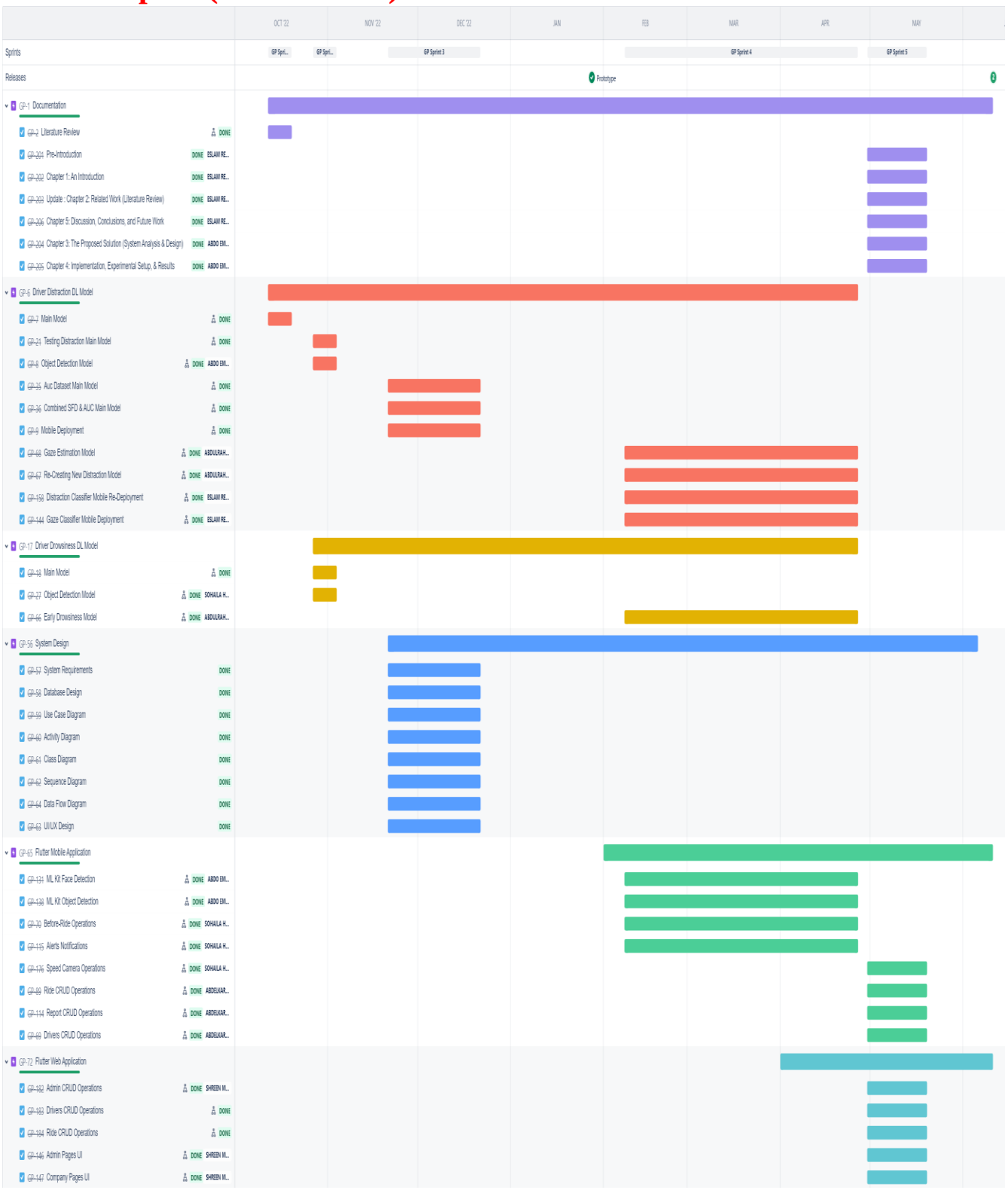
The use of sprints allows us to work in a more organized and efficient manner, delivering high-quality work at a faster and more frequent pace. It also provides us with the flexibility to adapt to changes and evolving requirements throughout the project.

Throughout our graduation year, our project has been divided into several sprints, each focusing on specific areas or goals. These sprints align with six main epics:

- Documentation
- Driver Distraction Deep Learning Models
- Driver Drowsiness Deep Learning Models
- System Design
- Flutter Mobile Application
- Flutter Web Application

In addition to the Scrum processes, we also uphold the core Scrum Values that form the foundation of the framework. These values include Commitment, Focus, Openness, Respect, and Courage. By embracing these values, we foster a collaborative and productive work environment that encourages teamwork, transparency, and continuous improvement.

1.5 Work plan (Gantt Chart)



Chapter 2: Literature Review (Related Work)

2.1 Introduction

Driving has become an indispensable part of modern life providing a high level of convenient mobility. However, this strong dependency on driving also leads to an increased number of road accidents. According to the World Health Organization's estimates, 1.25 million people die in road accidents per year, and up to 50 million people injure. Human factors are the main contributing cause in almost 90% of the road accidents having distraction as the main factor for around 68% of them. Accordingly, the development of a reliable Driver Monitoring System (DMS), which can supervise a driver's performance, alertness, and driving intention, contains utmost importance to prevent human-related road accidents also according to the American National Highway Traffic Safety Administration (NHTSA) report on May 17, 2022, [1] drowsy and distracted driving was responsible for 42,915 people died in motor vehicle traffic crashes last year, a 10.5% increase from the 38,824 fatalities in 2020. The projection is the highest number of fatalities since 2005 and the largest annual percentage increase in the Fatality Analysis Reporting System's history. Behind each of these numbers is a life tragically lost, and a family left behind.

Table 1: Fatalities and Fatality Rate by Quarter, Full Year, and the Percentage Change From the Corresponding Quarter or Full Year in the Previous Year

Quarter	1st Quarter (Jan–Mar)	2nd Quarter (Apr–Jun)	3rd Quarter (Jul–Sep)	4th Quarter (Oct–Dec)	Total (Full Year)
Fatalities and Percentage Change in Fatalities for the Corresponding Quarter and Total From the Previous Year					
2011	6,726 [-0.4%]	8,227 [-3.5%]	8,984 [-2.6%]	8,542 [+0.5%]	32,479 [-1.6%]
2012	7,521 [+11.8%]	8,612 [+4.7%]	9,171 [+2.1%]	8,478 [-0.7%]	33,782 [+4.0%]
2013	7,166 [-4.7%]	8,207 [-4.7%]	9,024 [-1.6%]	8,496 [+0.2%]	32,893 [-2.6%]
2014	6,856 [-4.3%]	8,179 [-0.3%]	8,799 [-2.5%]	8,910 [+4.9%]	32,744 [-0.5%]
2015	7,370 [+7.5%]	8,823 [+7.9%]	9,805 [+11.4%]	9,486 [+6.5%]	35,484 [+8.4%]
2016	8,154 [+10.6%]	9,563 [+8.4%]	10,078 [+2.8%]	10,011 [+5.5%]	37,806 [+6.5%]
2017	8,301 [+1.8%]	9,460 [-1.1%]	10,081 [+0.0%]	9,631 [-3.8%]	37,473 [-0.9%]
2018	8,203 [-1.2%]	9,323 [-1.4%]	9,934 [-1.5%]	9,375 [-2.7%]	36,835 [-1.7%]
2019	7,832 [-4.5%]	9,193 [-1.6%]	9,994 [+0.6%]	9,336 [-0.4%]	36,355 [-1.3%]
2020	7,893 [+0.8%]	9,141 [-0.6%]	11,315 [+13.2%]	10,475 [+12.2%]	38,824 [+6.8%]
2021†	8,935 [+13.2%]	11,135 [+21.8%]	11,780 [+4.1%]	11,065 [+5.6%]	42,915 [+10.5%]

2.2 Anomaly detection

Anomaly detection [2], also known as. outlier detection or novelty detection, is referred to as the process of detecting data instances that significantly deviate from the majority of data instances. Anomaly detection has been an active research area for several decades, with early exploration dating back as far as to 1960s. Due to the increasing demand and applications in broad domains, such as risk management, compliance, security, financial surveillance, health and medical risk, and AI safety, anomaly detection plays increasingly important roles, highlighted in various communities including data mining, machine learning, computer vision and statistics. In recent years, deep learning has shown tremendous capabilities in learning expressive representations of complex data such as high-dimensional data, temporal data, spatial

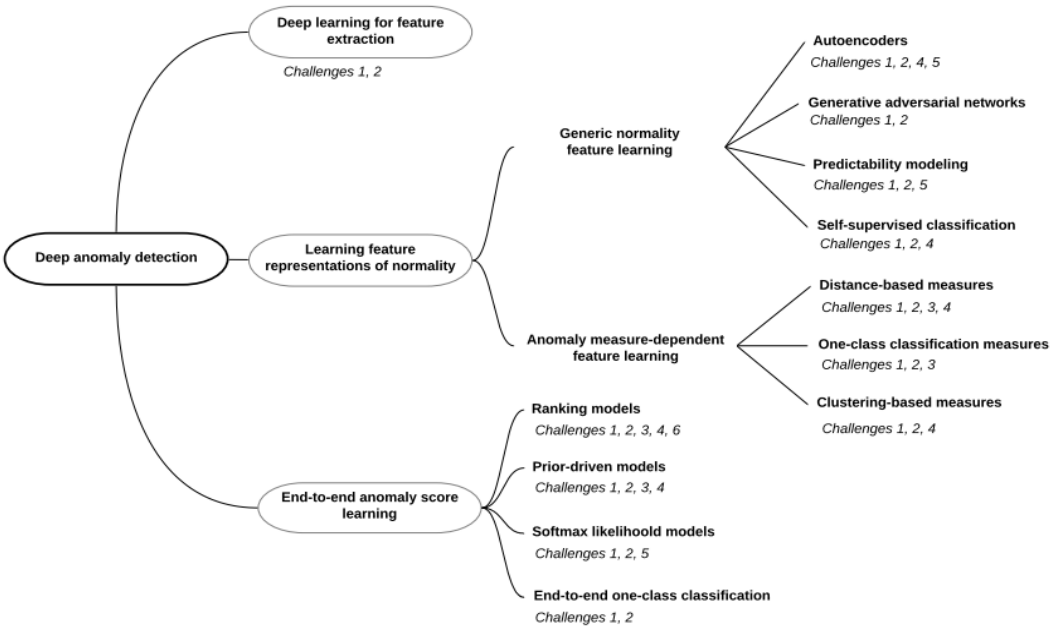
data, and graph data, pushing the boundaries of different learning tasks. Deep learning for anomaly detection, deep anomaly detection for short, aim at learning feature representations or anomaly scores via neural networks for the sake of anomaly detection. A large number of deep anomaly detection methods have been introduced, demonstrating significantly better performance than conventional anomaly detection on addressing challenging detection problems in a variety of real-world applications. This work aims to provide a comprehensive review of this area.

Anomaly explanation: In many safety-critical domains there may be some major risks if anomaly detection models are directly used as black-box models. For example, the rare data instances reported as anomalies may lead to possible algorithmic bias against the minority groups presented in the data, such as under-represented groups in drowsiness and distraction detection systems. An effective approach to mitigate this type of risk is to have anomaly explanation algorithms that provide straightforward clues about why a specific data instance is identified as anomaly. Human experts can then investigate and correct the bias. Providing such explanation can be as important as detection accuracy in some applications. However, most anomaly detection studies focus on detection accuracy only, ignoring the capability of providing explanation of the identified anomalies. To derive anomaly explanation from specific

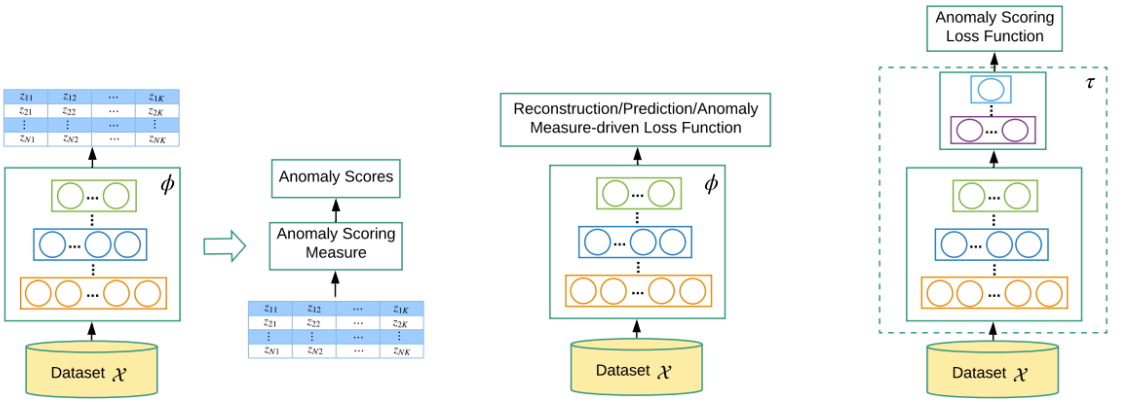
detection methods is still a largely unsolved problem, especially for complex models. Developing inherently interpretable anomaly detection models is also crucial, but it remains a main challenge to well balance the model's interpretability and effectiveness.

2.2.1 Categorization of Deep Anomaly Detection

To have a thorough understanding of the area, we introduce a hierarchical taxonomy to classify deep anomaly detection methods into three main categories, an overview of the taxonomy of the methods is shown in Fig 1. Specifically, deep anomaly detection consists of three conceptual paradigms - Deep Learning for Feature Extraction, Learning Feature Representations of Normality, and End-to-end Anomaly Score Learning



The procedure of these three paradigms is presented in Fig. 2. As shown in Fig. 2(a), deep learning and anomaly detection are fully separated in the first main category, so deep learning techniques are used as some independent feature extractors only. The two modules are dependent on each other in some form in the second main category presented in Fig. 2(b), with an objective of learning expressive representations of normality. This category of methods can be further divided into two subcategories based on how the representations are learned, i.e., whether using existing shallow anomaly measures (e.g., distance- and clustering-based measures) to guide the learning or not. These two subcategories encompass seven fine-grained categories of methods, with each category taking a different approach to formulate its objective function. The two modules are fully unified in the third main category presented in Fig. 2(c), in which the methods are dedicated to learning anomaly scores via neural networks in an end-to-end fashion. This category is further broken down into four subcategories based on the formulation of the anomaly scoring learning. In the following three sections we review these three paradigms in detail.



(a) Deep Learning for Feature Extraction (b) Learning Feature Representations of Normality (c) End-toend Anomaly Score Learning

Fig. 2. Conceptual Frameworks of Three Main Deep Anomaly Detection Approaches

2.2.2 Deep Learning for Feature Extraction

This category of methods aims at leveraging deep learning to extract low-dimensional feature representations from high-dimensional and/or non-linearly separable data for downstream anomaly detection. The feature extraction and the anomaly scoring are fully disjointed and independent from each other. Thus, the deep learning components work purely as dimensionality reduction only. Formally, the approach can be represented as

$$z = \phi\ (x; \Theta)$$

where ϕ: X $\mapsto\!\!\rightarrow$ Z is a deep neural network-based feature mapping function, with X $\in$ R D, Z $\in$ R K and normally $D \gg K$. An anomaly scoring method f that has no connection to the feature mapping ϕ is then applied onto the new space to calculate anomaly scores.

Compared to the dimension reduction methods that are popular in anomaly detection, such as principal component analysis (PCA) and random projection, deep learning techniques have been demonstrating substantially better capability in extracting semantic rich features and non-linear feature relations. Assumptions. The feature representations extracted by deep learning models preserve the discriminative information that helps separate anomalies from normal instances. One research line is to directly uses popular pre-trained deep learning models, such as AlexNet, VGG and ResNet, to extract low-dimensional features. This line is explored in anomaly detection in complex high-dimensional data such as image data and video data. One interesting work of this line is the unmasking framework for online anomaly detection. The key idea is to iteratively train a binary classifier to separate one set of video frames from its subsequent video frames in a sliding window, with the most discriminant features removed in each iteration step. This is analogous to an unmasking process. The framework assumes the first set of video frames as normal and evaluates its separability from its subsequent video frames. Thus, the training classification accuracy is expected to be high if the subsequent video frames are abnormal, and low otherwise. Clearly the power of the unmasking framework relies heavily on the quality of the features, so it is essential to have quality features to represent the video frames. The VGG model pre-trained on the ILSVRC benchmark is shown to be

effective to extract expressive appearance features for this purpose. In, the masking framework is formulated as a two-sample test task to understand its theoretical foundation. They also show that using features extracted from a dynamically updated sampling pool of video frames is found to improve the performance of the framework. Additionally, similar to many other tasks, the feature representations extracted from the deep models pre-trained on a source dataset can be transferred to fine-tune anomaly detectors on a target dataset. As shown in, one-class support vector machines (SVM) can be first initialized with the VGG models pre-trained on ILSVRC and then fine-tuned to improve anomaly classification on the MNIST data. Similarly, the ResNet models pre-trained on MNIST can empower abnormal event detection in various video surveillance datasets.

According to that there's a team from Technische Universität München [3] who made a model of driver anomaly detection which discover the drowsiness and we will see it in next section.

2.2.3 Driver Anomaly Detection (DAD) Dataset

There are several vision-based drivers monitoring datasets that are publicly available, but for the task of open set recognition such that normal driving should still be distinguished from unseen anomalous actions, there has been none. In order to fill this research gap, we have

recorded the Driver Anomaly Detection (DAD) dataset, which contains the following properties:

- The DAD dataset is large enough to train a Deep Neural Network architectures from scratch.
- The DAD dataset is multi-modal containing depth and infrared modalities such that system is operable at different lightning conditions.
- The DAD dataset is multi-view containing front and top views. These two views are recorded synchronously and complement each other.
- The videos are recorded with 45 frame-per-second providing high temporal resolution.

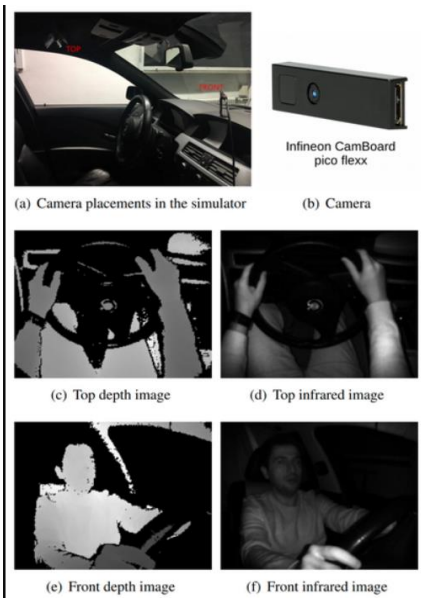


Fig.4 *Environment for data collection. (a) Driving simulator with camera placements. (b) Infineon CamBoard pico flexx camera installed for front and top views. Examples of (c) top depth, (d) top infrared, (e) front depth and (f) front infrared recordings.*

the DAD dataset using a driving simulator that is shown in Fig. 4. The driving simulator contains a real BMW car cockpit, and the subjects are instructed to drive in a computer game that is projected in front of the car. Two Infineon CamBoard pico flexx cameras are placed on top and in front of the driver. The front camera is installed to record the driver's head, body and visible part of the hands (left hand is mostly obscured by the driving wheel), while top camera is installed to focus on the drivers' hand movements. The dataset is recorded in synchronized depth and infrared modalities with the resolution of 224 x 171 pixels and frame rate of 45 fps. Example recordings for the two views and two modalities are shown in Fig. 4.

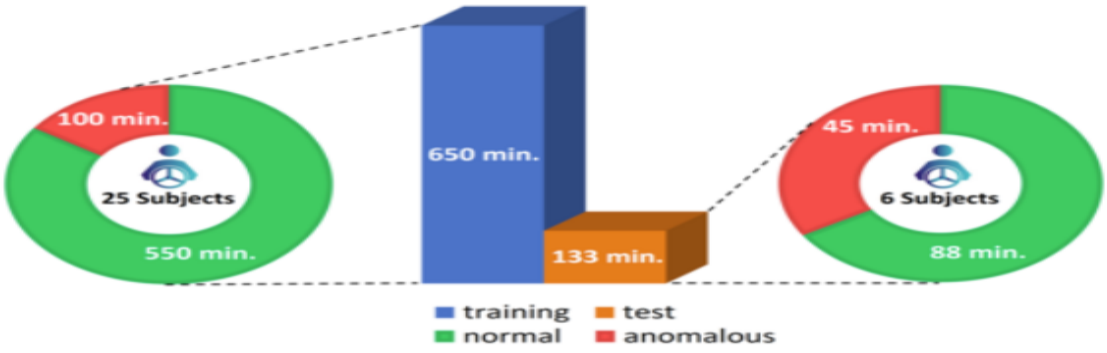
Anomalous Actions in Training Set	Anomalous Actions in Test Set		
Talking on the phone-left	Talking on the phone-left	Adjusting side mirror	Wearing glasses
Talking on the phone-right	Talking on the phone-right	Adjusting clothes	Taking off glasses
Messaging left	Messaging left	Adjusting glasses	Picking up something
Messaging right	Messaging right	Adjusting rear-view mirror	Wiping sweat
Talking with passengers	Talking with passengers	Adjusting sunroof	Touching face/hair
Reaching behind	Reaching behind	Wiping nose	Sneezing
Adjusting radio	Adjusting radio	Head dropping (dozing off)	Coughing
Drinking	Drinking	Eating	Reading

Table 1: Anomalous actions in the training and test sets. 16 actions in the test set that are not available in the training set are highlighted in red color.

For the dataset recording, 31 subjects are asked to drive in a computer game performing either *normal driving* or *anomalous driving*. The training set contains recordings of 25 subjects and each subject has 6 normal driving and 8 anomalous driving video recordings. Each normal driving video lasts about 3.5 minutes and each anomalous driving video

lasts about 30 seconds containing a different distracting action. The list of distracting actions recorded in the training set can be found in Table 1. In total, there are around 550 minutes recording for normal driving and 100 minutes recording of anomalous driving in the training set.

The test set contains 6 subjects, and each subject has 6 video recordings lasting around 3.5 minutes. Anomalous actions occur randomly during the videos. Most importantly, there are 16 distracting actions in the test set that are not available in the training set, which can be found in Table 1. *Because of these additional distracting actions, the networks need to be trained according to open set recognition task and distinguish normal driving no matter what the distracting action is*. The complete test consists of 88 minutes recording for normal driving and 45 minutes recording of anomalous driving. The test set constitutes 17% of the complete DAD dataset, which is around 95 GB. The dataset statistics can be found in Fig. 5.



2.3 Drowsiness Detection

2.3.1 Definition

Road crashes and related forms of accidents are a common cause of injury and death among the human population. Drowsiness is one of the most factors of road accidents, leading to severe injuries and deaths every year, so drowsiness means difficulty staying awake, which can lead to falling asleep, even while performing activities. It is related to the biological wakefulness and sleeps linked to the circadian rhythm; it is therefore not directly related to the accomplishment of a task. During a 24-hour cycle, the human body tends to sleep at certain times than at others. This is mainly the case during the night, between midnight and about 6 a.m., when the human body naturally turns towards sleep and vigilance decreases accordingly. take thousands of lives each year worldwide. According to 2015 data from the World Health Organization, road traffic injuries resulted in approximately 1.25 million deaths worldwide, approximately every 25 seconds an individual will experience a fatal crash. While the cost of traffic accidents in Europe is estimated at around 160 billion Euros, driver drowsiness accounts for approximately 100,000 accidents per year in the United States alone as reported by The American National Highway Traffic Safety Administration (NHTSA). According to 2022 data from the World Health Organization, approximately 1.3 million people die each year as a result of road traffic crashes, between 20 and 50 million more people

suffer non-fatal injuries, with many incurring a disability as a result of their injury. And road traffic injuries cause considerable economic losses to individuals, their families, and to nations as a whole. These losses arise from the cost of treatment as well as lost productivity for those killed or disabled by their injuries, and for family members who need to take time off work or school to care for the injured. Road traffic crashes cost most countries 3% of their gross domestic product. And in data analyzed from these countries drowsy driving represents 10 to 30% of all crashes, this fact clearly exhibits a need for a drowsiness detection application in order to reduce such accidents and enhance the safety of the driver and the passengers, that can help prevent such accidents and ultimately save lives. So, Detecting Driver Drowsiness is a car safety technology which helps prevent accidents caused by the driver getting drowsy, and driver drowsiness detection systems have been worked on and developed by various researchers across the world.

2.3.2 Types

Some of the current systems learn driver patterns and can detect when a driver is becoming drowsy in different way. But there is no standard tool to measure the level of vigilance; the only solution is to observe the signs of hypervigilance emitted by the driver and analyze them. These signs can be divided into behavioral and physiological signs. Behavioral signs are manifested by abnormal behavior of the driver Various

technologies can be used to try to detect driver drowsiness, and These can be broadly categorized to depend on the following methods:

2.3.2.1 Vehicle Based
2.3.2.2 Behavioral Based
2.3.2.3 Physiological Based

2.3.2.1 Vehicle based drowsiness detection systems.

work by monitoring the vehicle's lane changes, steering wheel rotation, speed, pressure on accelerator pedal. There is a lot of examples of Vehicle based, some of them is:

a) Steering pattern monitoring

Primarily uses steering input from electric power steering system. Monitoring a driver this way only works as long as a driver actually steers a vehicle actively instead of using an automatic lane-keeping system.

b) Vehicle position in lane monitoring

Uses a lane monitoring camera. Monitoring a driver this way only works as long as a driver actually steers a vehicle actively instead of using an automatic lane-keeping system.

2.3.2.2) Behavior based drowsiness detection systems.

It depends on the behavior of the driver. To be more specific eye closure, yawn, and head posture are monitored through a camera to detect drowsiness in such systems.

There is a lot of examples of Behavior based, some of them is:

a) Driver eye/face monitoring

Uses computer vision to observe the driver's face if he is blinking, stiff neck or back pain, Frequent yawning, Difficulty in keeping the eyes open and the head in a frontal position, either using a built-in camera or on mobile devices.

2.3.2.3 Physiological based drowsiness detection systems

It is relied on the correlation between physiological signals ECG (Electrocardiogram) and EOG (Electrooculogram) to detect driver drowsiness. There is a lot of examples of Physiological based, some of them is:

a) Physiological measurement

Requires body sensors to measure parameters like brain activity, heart rate, skin conductance, muscle activity or head movements.

All of these technologies just alert for the driver that it is essential to take a break for at least 15 minutes before resuming driving.

Unfortunately, drivers tend to overestimate their level of alertness and very often ignore these signs.

2.3.3 Related works

[4] in this paper using mobile application and deep learning architecture (convolutional neural networks in computer vision) the results on the test dataset and real-world testing, we conclude that the proposed methodology of treating the task of drowsiness detection as an object detection task is practical and reliable with a good accuracy. Moreover, this technique works in real time on an Android device, which makes it very accessible.

it is just needed to enhance the Drowsy dataset and adding low light images (in near infra-red light) to enable the model to detect drowsiness in low light conditions because it didn't work at night so this would be beneficial as drowsiness related accidents have a high chance of occurrence during night-time driving.

[5] in this paper introduces a literature review of driver drowsiness detection systems based on an analysis of physiological signals, facial features, and driving patterns.

The physiological signals analysis technique is the most accurate method of all proposed techniques for driver drowsiness detection. but using it is limited because of the equipment's price, and that requires many sensors to be attached to the driver's body, which is quite intrusive for the driver.

but methods based on driving patterns analysis are not intrusive. some information directly related to driving activity is already available in the vehicle through the vehicle control network (Controller Area Network or CAN), such as accelerator and brake pedal pressure, steering wheel angle, On the other hand, driving assistance systems, such as lane keeping assistance, already rely on information related to the vehicle's position in the lane. However, lane lines are not always visible. As a result, SDLP or the time to cross the line is not always available in real time. This is because SDLP requires robust detection of the trackside lines, which depending on the road conditions, is not always possible. It is also difficult to compare the numerical values and results presented by different studies as not all of them differentiate between bias and accuracy.

after all they agreed that the advantage of facial features analysis based-driver drowsiness these measurements, because that they are easily accessible in a non-intrusive way as there is no need for

sensors in contact with the participant. Therefore, they can be easily used in real life situations. Nevertheless, these measures rely on the fact that the acquisitions are based on image processing algorithms to detect the head, mouth, eyes and gaze. As a result, the quality of the estimation of the driver's state is highly dependent on the quality of the image processing. But it is still difficult in the case of driving, especially for

people wearing glasses, withstanding changes in light or in low light situations. In addition, there are a number of constraints related to

Class	Hardware	Advantages	Limitations
Physiological Signals	Electroencephalogram (EEG) and Electrocardiogram (ECG) Sensors	Accurate, Reliable	Intrusive
Facial Features	External camera	Non-intrusive	Not possible in low brightness and all varying light conditions
Driving Patterns	Steering angle sensor for (SWM) External camera for (SDLP) OBD-2 for Acceleration, Speed, Gravity, and (RPM)	Non-intrusive	Unreliable, Less Accurate

the placement of the camera in the vehicle and the field of view covered by the lens. Figure 6 provides the advantages and limitations of each class.

To take advantage of each technique, a hybrid system that combines two and more techniques will be efficient and able to be used in real-time. It just needs to focus on designing and implementing an embedded system combining facial features and pattern analysis approaches to detect driver drowsiness, estimate the probability of accident risk and then alert the driver.

[6] in this paper recognizing driver drowsiness expression by using facial dynamic fusion information and a deep belief network (DBN) to

address the aforementioned problem. First, the landmarks and textures of the facial region are extracted from videos captured using a high-definition camera. Then, a DBN is built to classify facial drowsiness expressions. Finally, the authors' method is tested on a driver drowsiness dataset, which includes different genders, ages, head poses and illuminations. Certain experiments are also carried out to investigate the effects of different facial subregions and temporal resolutions on the accuracy of driver fatigue recognition. Results demonstrate the validity of the proposed method, which has an average accuracy of 96.7%. the disadvantage is in the large head rotations and occlusions can reduce the effectiveness of the proposed method.so it just needs to address this limitation by combining other useful information (e.g., head and pupil movements). Also, by extend the dataset to complex and diverse actual driving environments to improve the generalization capability of the classifier and obtain reliable experimental results.

[7] In this paper using deep learning method that can be implemented on Android applications with high accuracy. the system is to detect facial landmark from images and deliver the obtained data to the trained model to identify the driver's state. The purpose of the method is to reduce the model's size considering that current applications cannot be used in embedded systems due to their limited calculation and storage capacity. According to the experimental results, the size of the used model is

small while having the accuracy rate of 81%. Hence, it can be integrated into advanced driver-assistance systems, the Driver drowsiness detection system, and mobile applications.

[8] in This paper Convolutional Neural Networks (CNN) are used with regarding the two goals of real-time application, including high accuracy and fastness. Three networks introduced as a potential network for eye status classification in which one of them is a Fully Designed Neural Network (FD-NN), and others use Transfer Learning in VGG16 and VGG19 with extra designed layers (TL-VGG). in the first step of the system, the system captures a stream of frames, and in the preprocessing unit, landmark points applied to access ROI, then eye region is selected, and the preprocessing unit selects the eye front of the camera. Then, the image converts to gray image, and contrast of eye equalizes. Finally, the image resizes to 24×24 pixels. The authors use the output of this step as input to the network to eye state classification. If the network detects the eye is closed for more than 12 successive images, an alarm will be sent for the driver. Otherwise, it considers it as blinking. The experimental results indicated that the FD-NN network reaches 98.15% accuracy.

just need to focus on yawning analysis for fatigue detection by landmark points of mouth. Also, a different level of drowsiness can investigate without limitations of being in two situations because the boundaries among different levels of drowsiness are so narrow.

[9] in this paper provides details of behavioral, vehicular, and physiological parameters-based drowsiness detection techniques. The comparative analysis showed that none of these techniques provide full accuracy, but physiological parameters-based techniques give more accurate results than others. The advantage of this approach is that it alerts the driver to take some rest, before the physical symptoms of drowsiness appear, their non-intrusiveness can be reduced using

Hybrid approach	Hybrid parameters	Method & Classifiers	Accuracy
Physiological, Behavioral and vehicular	Heart rate, BP, temperature, speed eyelid closure ratio	HRV color model, Fuzzy Bayesian network	94%-99%
Physiological and behavioral	Heart rate, Eye blinking duration	PSG, NASA lead method	Nil
Behavioral and vehicular	Eye status, lateral position, steering wheel angle	KSS, Neutral network, BPL algorithm.	87.78%-94.63%
Physiological and behavioral	Head movement, heart rate	Frame difference algorithm, R-peak detection algorithm	Nil
Physiological and behavioral	Heart rate, Eyelid closure ratio	PERCLOS, Burg method, Kruskal-Wallis test,	86%-96%

wireless sensors on the drivers' body, driving seat, seat cover, steering wheel, etc. Hybrid of these techniques like in figure 7 such as

physiological measures combined with vehicular or behavioral measures, helps in overcoming the problem associated with individual technique thus results in improved drowsiness detection results like the combination of ECG and EEG features achieves the high-performance results emphasizing the fact that combining the physiological signals improves the performance instead of using them alone. The top supervised learning techniques have been presented. A comparative study of classifier shows in figure 8 different accuracy in various situations. However, SVM is the mostly commonly used classifiers

which gives better accuracy and speed in most of the situations, but not suitable for large datasets. HMM shows a less error rate, but both CNN and HMM are slow in training and expensive as compared to SVM classifier.

Technique	Parameters	Pros	Cons
HMM	Probabilities, hidden states, eye state, head position, eye blinks, eye closure	Capture dependencies between the measurement, able to represent the variance of the appliance's power	Limited by the intrinsic structure
CNN	Eye state, visual features, eye gaze,	Accuracy in the image recognition	Slow to train, need lots of training data, high computational cost.
SVM	Eye Closure, Eye State, Eye Closure and yawning	Guarantees optimal, abundance of implementation, implicitly, good out-of-sample generalization.	Not suitable for large data set, less effective on nosier dataset

Also based on the error rates of SVM, HMM, and CNN classification methods, Lesser error rate leads to greater efficiency. so that HMM is more accurate as it has a lesser error rate as compared to the other two. But SVM is easy to use, so it is the most widely used classification method.

2.4 Distraction Detection

2.4.1 Definition:

Distracted driving is any activity that diverts attention from driving, including talking or texting on your phone, eating and drinking, talking to people in your vehicle, fiddling with the stereo, entertainment, or navigation system — anything that takes your attention away from the task of safe driving. Distractions are shown to compromise the safety of the driver, passengers, pedestrians, and people in other vehicles. Cellular

device use while behind the wheel is one of the most common forms of distracted driving. According to the United States Department of Transportation, “texting while driving creates a crash risk 23 times higher than driving while not distracted". Studies and polls regularly find that over 30% of United States drivers had recently texted and driven. Distracted driving is particularly common among, but not exclusive to, younger drivers. Texting is the most alarming distraction. Sending or reading a text takes your eyes off the road for 5 seconds. You cannot drive safely unless the task of driving has your full attention. Any non-driving activity you engage in is a potential distraction and increases your risk of crashing.

2.4.2 Types of distraction:

Anything that takes your attention away from driving can be a distraction. Sending a text message, talking on a cell phone, using a navigation system, and eating while driving are a few examples of distracted driving. Any of these distractions can endanger you, your passengers, and others on the road.

There are three main types of distraction:

- **Visual**
- **Manual**
- **Cognitive**

a- Visual distraction:

Visual distractions involve taking one's eyes off the road, such as looking at a GPS system, looking at roadside billboards, or checking a child's seat belt in the rear-view mirror.

b- Manual or Physical distraction:

involve taking one's hands off the wheel, such as searching for something in a bag, eating or drinking, grooming, checking text messages, putting on makeup, or changing radio stations.

c- Cognitive:

Cognitive distractions occur when an individual is not mentally focused on the act of driving, such as if a driver is worrying about a sick relative, she may not pay close attention when making turns or changing lanes. [10]

Some distractions can combine some or all these groups, such as texting and calling on one's cell phone.

Driving distractions can greatly vary in form and severity. They range from the use of cell phones and other electronics to rubbernecking, carrying passengers including children and pets in the vehicle, eating while driving and searching for misplaced items. Distractions within the vehicle itself can be problematic. With all the new adaptations to technology in our vehicles, there is a higher chance of looking at a screen and taking your attention off of the road. There is another

distraction factor to put in place here: driving with fatigue or being so out of focus that you become drowsy. The extended use of the new automation systems may cause the driver to over rely on the system and become disengaged completely from the wheel as well as the road ahead. An experienced driver that is used to the automation systems will be actively engaged in distracted driving.

2.4.3 Distraction Rates:

Distracted driving impacts thousands of Americans each year

- In the United States, over 3,100 people were killed and about 424,000 were injured in crashes involving a distracted driver in 2019.
- About 1 in 5 of the people who died in crashes involving a distracted driver in 2019 were not in vehicles—they were walking, riding their bikes, or otherwise outside a vehicle.[10]

Using a cell phone while driving creates enormous potential for deaths and injuries on **United States** roads. In 2020, 3,142 people were killed in motor vehicle crashes involving distracted drivers. **Nine people in the United States are killed every day in crashes that are reported to involve a distracted driver**.

2.4.4 Related work:

[11] In this paper, CNN was developed based on method to detect distracted driver and identify the cause of distractions like talking, sleeping or eating by means of face and hand localization. Four architectures namely CNN, VGG-16, ResNet50 and MobileNetV2 have been adopted for transfer learning. To verify the effectiveness, the proposed model is trained with thousands of images from a publicly available dataset containing ten different postures or conditions of a distracted driver and analyzed the results using various performance metrics. The performance results showed that the pretrained MobileNetV2 model has the best classification efficiency. Extensive experiments have been conducted with an ideal dataset to verify the model. It is found that the ResNet50 and MobileNetV2 provide higher accuracy of 94.50% and 98.12% respectively. The proposed model results can be used to design real-time driver distraction detection systems. Though the proposed model provides high accuracy, a real-time test-bed implementation can make the work more effective.

[12] In this paper, An effective camera-based framework was proposed for real-time identification of drivers distraction by using a deep learning approach with edge and cloud computing technologies. More specifically, the framework consists of three modules, including the distraction detection module deployed on edge devices in the vehicle environment, the training module deployed in the cloud environment, and finally the analyzing module implemented in the monitoring environment (administrator side) connected with a telecommunication network. The proposed framework is developed using two deep learning models. The first is a custom deep convolutional neural network (CDCNN) model, and the second one is a visual geometry group-16 (VGG16)-based fine-tuned model. Several experiments are conducted on a public large-scale driver distraction dataset to evaluate the two models. The experimental results show that the accuracy rates were 99.64% for the first model and 99.73% for the second model using a holdout test set of 10%. In addition, the first and second models have achieved accuracy rates of 99.36% and 99.57% using a holdout test set of 30%. The results confirmed the applicability and appropriateness of the adopted deep learning models for designing the proposed driver distraction detection framework.

[14] In this paper, Driver gaze has been shown to be an excellent surrogate for driver attention in intelligent vehicles. With the recent surge of highly autonomous vehicles, driver gaze can be useful for determining the handoff time to a human driver. While there has been significant improvement in personalized driver gaze zone estimation systems, a generalized system which is invariant to different subjects, perspectives and scales is still lacking. We take a step towards this generalized system using Convolutional Neural Networks (CNNs). We finetune 4 popular CNN architectures for this task and provide extensive comparisons of their outputs. We additionally experiment with different input image patches, and examine how image size affects performance. For training and testing the networks, we collect a large naturalistic driving dataset comprising of 11 long drives, driven by 10 subjects in two different cars. Our best performing model achieves an accuracy of 95.18% during cross-subject testing, outperforming current state of the art techniques for this task. Finally, we evaluate our best performing model on the publicly available Columbia Gaze Dataset comprising of images from 56 subjects with varying head pose and gaze directions. Without any training, our model successfully encodes the different gaze directions on this diverse dataset, demonstrating good generalization capabilities.

[13] In this research, a proposed hybrid approach is presented. The approach is based on deep learning to detect the driver's actions and

eliminate the driver's distraction as a packed solution. The detection is performed by analyzing the driver's actions and his head pose. The elimination is made by using voice commands that are based on trigger words, speech to text, and text classification models to access the car's functions such as the air-conditioning, radio, etc. The results of the driver's actions classification showed 94.1% accuracy on the AUC benchmark database for driver distraction achieving the state-of-the-art accuracy on this benchmark. The results of the command to text classification are 95.19% while the results of the head pose estimation show a 6.21-degree MAE in face angles detection. With using our car commands dataset, the domain of the speech recognition output is more focused on car commands. The previously mentioned algorithms are beneficial to the safety of the driver. He can use his voice to operate the car accessories. His alert state is monitored, and he is warned through an alarm if a distraction is detected. However, this research is not concerned with the detection of retinal abnormalities such as sleeping with eyes open. The results of real-time testing show 0.080 second response time for the driver's behavior classification and command following with the use of graphical processing units.

2.5 Conclusion

We have reviewed the recent driver distraction and drowsiness detection techniques used in Advanced Driving Assistance System (ADAS)

applications. The optimal use of these techniques in vehicles can prevent most accidents caused by distracted and drowsy driving every day. The presented and detailed techniques are divided into three classes according to the processed signals, including physiological signals, facial features, and driving patterns. The advantages and limitations for each class of techniques were also discussed in terms of accuracy, reliability, intrusiveness, and hardware requirement. As a result, each class of techniques has advantages and limitations. To take advantage of each technique, a hybrid system that combines two and more techniques will be efficient and able to be used in real-time.

The use of these safety systems which detect distraction and drowsiness is not widespread and is uncommon among drivers because they are usually available in luxury vehicles. An increased embedding and connecting of smart devices equipped with sensors and mobile operating systems like Android, which has the largest installed operating system in cars, was shown by surveys in 2015. In

addition, machine learning has made groundbreaking advances in recent years, especially in the area of deep learning. Thus, the use of these new technologies and methodologies can be an effective way to not only increase the efficiencies of the existing real-time driver distraction and drowsiness detection system but also provide a tool that can be widely used by drivers.

Chapter 3: System Analysis & Design

3.1 Solution Methodology

In this section, we outline the solution methodology for our AI-Powered Driver Monitoring System, which comprises three integrated components.

3.1.1 Deep Learning

To address the detection of distractions, we utilized TensorFlow and Keras to create two custom pre-trained models. The first model focuses on phone distraction detection and was trained on a combination of three distraction datasets. The second model utilizes the LISA Dataset to perform gaze estimation for detecting road distractions. TensorFlow Lite was employed to convert the models into a mobile-compatible format. We integrated the Google ML Kit face detection model to identify the driver's face and generate face landmarks, allowing us to crop the region of interest for gaze estimation. Additionally, the Google ML Kit model was used for drowsiness detection. Furthermore, we developed a dedicated testing application to validate the accuracy of the deep learning monitoring system and evaluate its performance in providing accurate predictions.

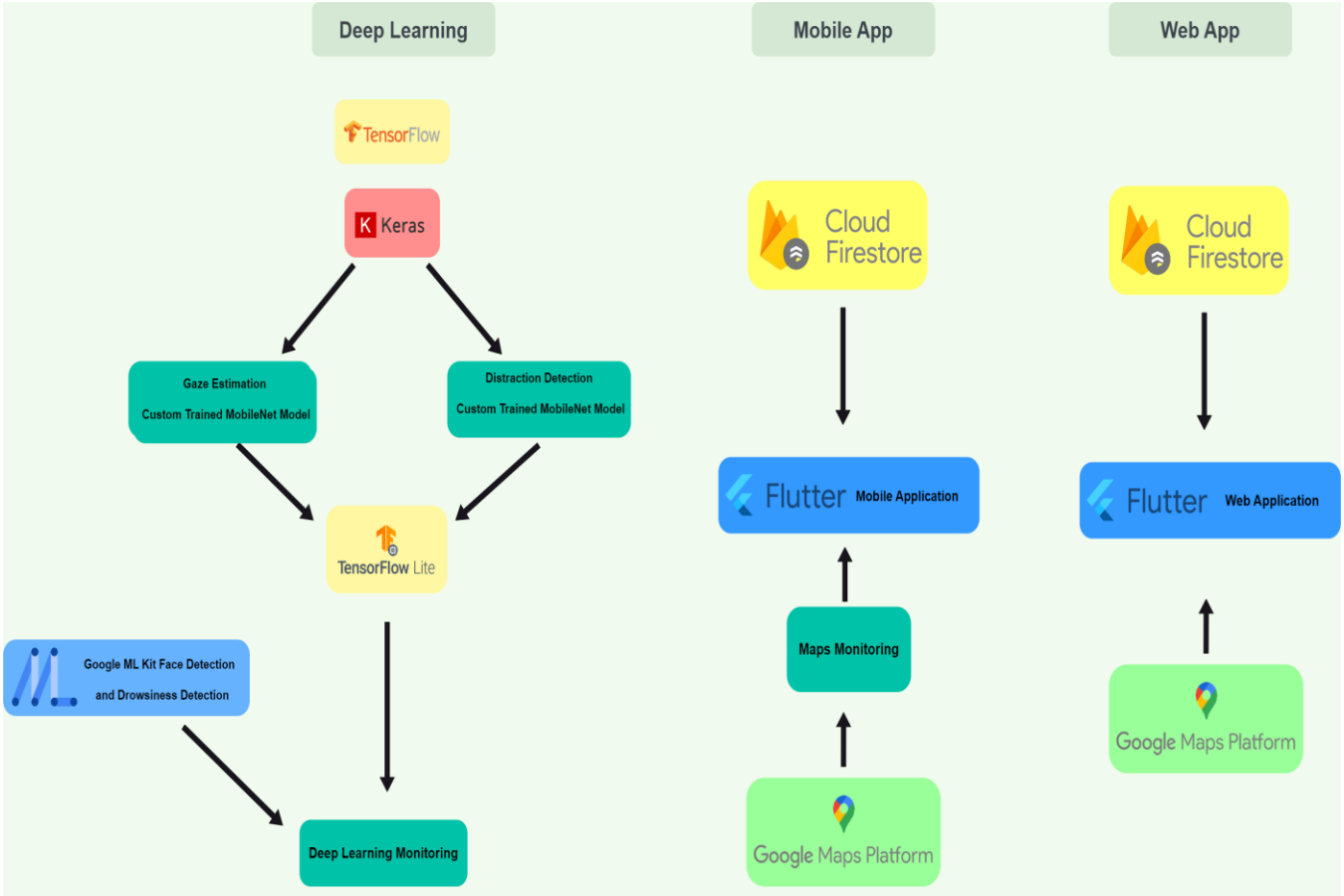
3.1.2 Mobile Application

For the development of our mobile application, we chose the Flutter framework due to its user-friendly nature, easy learning curve, and seamless integration with our system components. Firebase Firestore was selected as our database solution, benefiting from its cloud-based architecture and compatibility with our web application. Leveraging Google Maps API, we incorporated map monitoring functionality into our mobile application, as it offers smooth integration with our system components and serves as the best free option for a maps API with strong support in the Flutter framework.

3.1.3 Web Application

Similar to our mobile application, we utilized the Flutter framework for the development of our web application to maintain consistency across our system components. Firebase Firestore was employed as the database solution, enabling cloud-based storage and seamless integration with our mobile app. To provide map monitoring functionality, we integrated Google Maps API into our web application, leveraging its smooth integration capabilities with our system components.

3.1.4 System Components Diagram

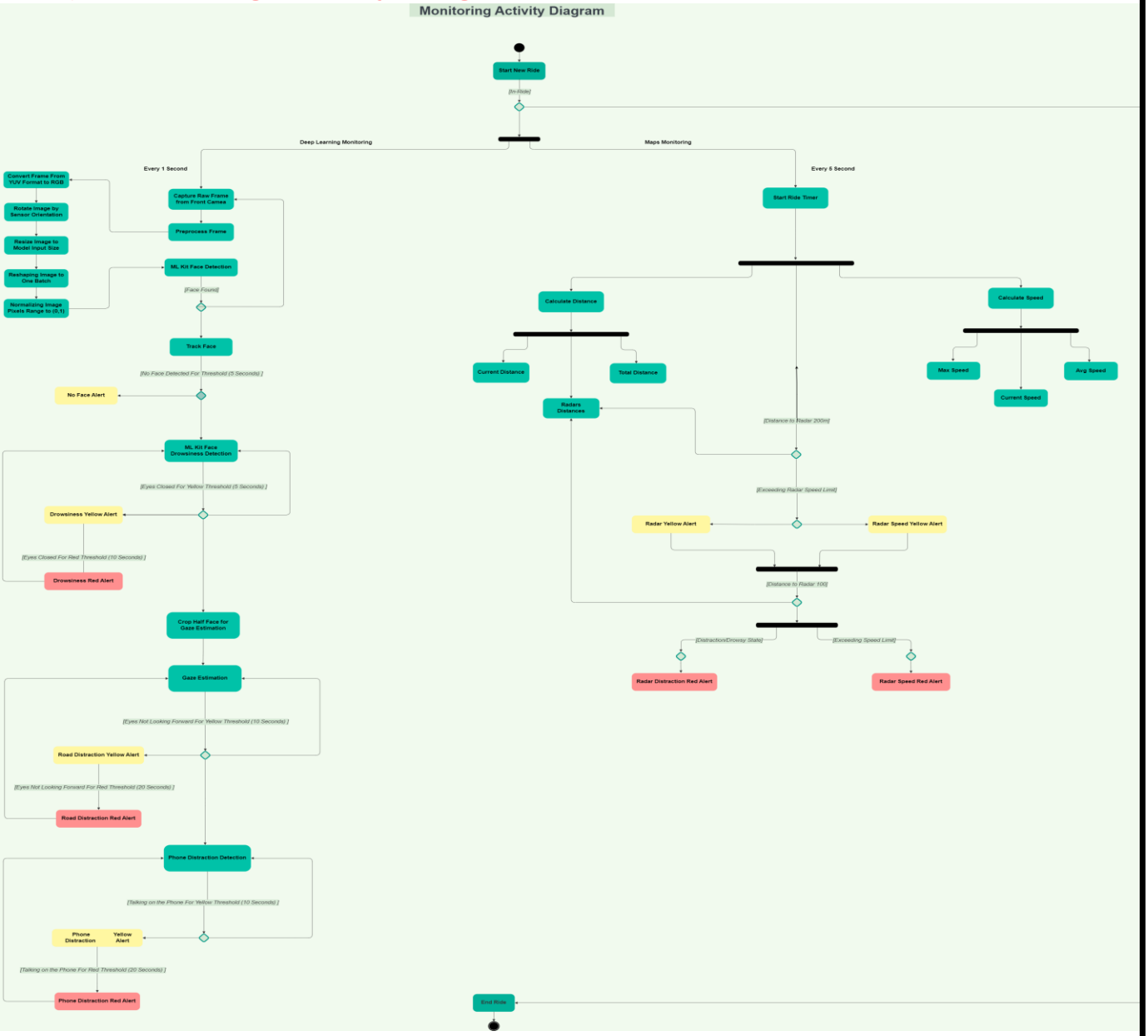


By adopting this solution methodology, we ensured the utilization of cutting-edge deep learning techniques for distraction detection,

streamlined development using the Flutter framework for both the mobile and web applications, and leveraged the power of Firebase Firestore and Google Maps API for efficient data management and map monitoring functionalities.

3.2 Monitoring Methodology

3.2.1) Monitoring Activity Diagram



When the Driver Start the ride, while he is in-ride and as long as he didn't end the ride the monitoring system will work in the background while the driving is navigating through google maps application.

3.2.2) Deep learning Monitoring

In the Deep learning Monitoring section, we present the process of capturing and preprocessing the input image for our deep learning models. The image is obtained from the phone's front camera and undergoes several steps before being passed to the models. Firstly, we preprocess the image by converting it from raw frame YUV format to RGB format for compatibility with flutter plugins. Additionally, we rotate the image based on the mobile phone sensor orientation and crop it to match the expected input size of our models. The image is then reshaped into batches and normalized to a pixel range of 0 to 1, similar to the training data.

Next, we utilize the Google ML Kit face detection model to determine if the driver's face is present. If the face is not detected, the deep learning monitoring stops, and the driver is alerted to adjust the phone's front camera angle. If the driver's face is detected, we employ the same face detection model to detect drowsiness by monitoring the driver's eye state. Additionally, we utilize the gaze estimation model to determine the driver's focus on the road. By utilizing the face landmarks detected by the Google ML Kit model, we accurately select the desired region of the face for gaze estimation. Finally, we check for phone distractions using the distraction detection model, which analyzes the entire processed frame from the front camera to detect activities such as texting or talking.

The deep learning models are run consecutively, rather than in parallel, every second. The average inference time for our deep learning monitoring is approximately 300 milliseconds on most mobile devices,

utilizing only one processor core. If the inference time is less than one second, we wait for the remaining time to make inferences every second. However, if the mobile device's hardware cannot perform inferences within one second, we proceed with inferences as soon as possible.

In terms of alerting the driver during unsafe driving states, we have implemented staged alerts consisting of yellow and red alerts. Yellow alerts are triggered when specific thresholds are reached and serve as normal notifications with an auto-dismiss feature. On the other hand, red alerts are stronger notifications that function like alarms, persisting until the driver interacts with the notification. The thresholds for the staged alerts differ for each of our three deep learning models. For drowsiness, a yellow alert is triggered after 5 seconds of drowsiness, while a red alert is triggered after 10 seconds. Regarding gaze, a yellow alert is produced if the driver looks left or right for 10 seconds, and a red alert is produced after 20 seconds. Lastly, for distraction, a yellow alert is triggered after 10 seconds of texting or talking, and a red alert is produced after 20 seconds.

By implementing this deep learning monitoring approach, we ensure the accurate analysis of driver behavior and the timely delivery of alerts, thereby enhancing the overall safety of the AI-Powered Driver Monitoring System.

3.2.3) Maps monitoring

The Maps monitoring section aims to improve road safety by providing proactive alerts to drivers before they approach radars. Our system alerts drivers when they are approaching radars and offers specialized notifications if they exceed the speed limit or are in an unsafe driving state near radars.

To achieve this, we integrated the Google Maps API, which enables us to monitor the driver's ride on maps effectively. By utilizing various properties and methods, we track the driver's position, distance traveled, and speed, ensuring real-time monitoring. Additionally, we leverage the API's capabilities to alert the driver of any upcoming speed cameras stored in our Firestore Database.

Maps Monitoring tracks the driver's ride every 5 seconds, allowing us to monitor the distance traveled and current speed accurately. This enables us to deliver timely alerts to the driver if they are approaching a speed camera. Our system ensures that the driver receives a notification at least 200 meters before reaching a speed camera, as long as their speed is 144km/h or lower. By providing this advance warning, we promote driver awareness and help prevent accidents caused by speeding.

By incorporating Maps Monitoring into our AI-Powered Driver Monitoring System, we enhance driver safety by proactively notifying them about potential speed cameras, ultimately reducing the risk of accidents and creating a safer driving experience for all road users.

3.3 Functional & Non-Functional Requirements

3.3.1 Functional Requirements

Overview

- **Mobile Interface for Drivers**:

The system should provide a mobile interface specifically designed for drivers.

- **User Story**: As a driver, I want to access the system through a mobile application to perform my rides and interact with the monitoring system.
- **Acceptance Criteria**:
  - The mobile interface shall have a login screen for drivers to authenticate their credentials.
  - Drivers should be able to view their rides and reports on the mobile interface and interact with the monitoring system.
  - The mobile interface shall provide a user-friendly and intuitive interface for drivers to easily interact with the system

- **Web Interface for Company and Admins**:

The system should offer a web interface accessible to the company and administrators.

User Story: As a company representative or administrator, I want to access the system through a web browser to manage and oversee operations.

- **Acceptance Criteria**:
  - The web interface shall have separate logins for company representatives and administrators.
  - Administrators should be able to view and manage driver profiles, rides, and reports to monitor drivers.
  - Company representatives should have access to advanced system settings, admins management, and generate reports.

- **System Actors: Company, Admin, and Driver**:

The system should support three distinct user roles: Company, Admin, and Driver.

- **User Story**: As a user with a specific role, I want to perform tasks and access features relevant to my role.
- **Acceptance Criteria**:
  - The system shall differentiate user roles and provide role-based access control.
  - Company representatives should have access to company-specific functionalities and data.
  - Administrators should have the necessary privileges to manage and monitor drivers.
  - Drivers should have access to features related to their rides, reports, and monitoring system.

Requirements Specification

- **Company Role:**

The company role in the system encompasses the following functionalities:

1. **Login**: The system should allow company representatives to log in using their credentials.
    - **User Story**: As a company representative, I want to log in to the system to access company-specific functionalities.
    - **Acceptance Criteria**:
        - The system shall provide a login screen where company representatives can enter their email and password.
        - Upon successful login, the system shall authenticate the credentials and grant access to company-related features.
2. **View Company Profile**: Company representatives should be able to view the company's profile information within the system.
    - **User Story**: As a company representative, I want to view the company profile to retrieve and review company information.
    - **Acceptance Criteria**:
        - The system shall provide a dedicated section or page to display the company's profile details.
        - Company representatives should be able to access and view information such as company name, address, contact details, and other relevant data.
3. **Edit Company Information**: Company representatives should have the ability to edit and update the company's information stored in the system.
    - **User Story**: As a company representative, I want to modify the company information to keep it up to date.
    - **Acceptance Criteria**:
        - The system shall provide an interface or form where company representatives can edit specific fields of the company's information.
        - After making changes, the system shall save the updated information and reflect the changes across the system.

4. **Add Company Admins**: Company representatives should be able to add new administrators to the system, granting them access to relevant functionalities.
    - **User Story**: As a company representative, I want to add new administrators to the system to manage and monitor company drivers.
    - **Acceptance Criteria**:
        - The system shall provide a form or interface to enter the necessary details for adding a new company administrator.
        - After successful submission, the system shall create a new administrator account with the specified permissions.
5. **View Company Admins**: Company representatives should be able to view a list of company administrators registered in the system.
    - **User Story**: As a company representative, I want to see the list of administrators associated with the company for oversight and management.
    - **Acceptance Criteria**:
        - The system shall provide a designated section or page to display a list of company administrators.
        - Company representatives should be able to view details such as names, timeline information on the system.
6. **Edit Company Admins**: Company representatives should have the capability to modify the details of existing company administrators.
    - **User Story**: As a company representative, I want to edit the information of existing administrators for proper management.
    - **Acceptance Criteria**:
        - The system shall provide an interface or form to edit specific fields of an existing company administrator's information.
        - After making changes, the system shall save the updated information and reflect the modifications in the administrator's profile.
7. **Delete Company Admins**: Company representatives should be able to remove administrators from the system when necessary.

- **User Story**: As a company representative, I want to delete administrators from the system to revoke their access and permissions.
- **Acceptance Criteria**:
  - The system shall provide a mechanism or action to remove a company administrator from the system.
  - After confirming the deletion, the system shall permanently remove the administrator's account and associated data.

8. **View Company Drivers**: Company representatives should have access to view a list of drivers associated with the company within the system.
   - **User Story**: As a company representative, I want to view the list of drivers associated with the company for monitoring and management purposes.
   - **Acceptance Criteria**:
     - The system shall provide a designated section or page to display a list of drivers affiliated with the company.
     - Company representatives should be able to view details such as drivers' names, contact information, and their rides and reports.
9. **View Company Drivers' Past Rides**: Company representatives should have access to view the past ride information of company drivers.
   - **User Story**: As a company representative, I want to access the past ride information of company drivers for monitoring and analysis purposes.
   - **Acceptance Criteria**:
     - The system shall provide a section or page where company representatives can view the past ride details of company drivers.
     - Company representatives should be able to access information such as ride dates, durations, destinations, and other relevant data.
10. **View Company Drivers' Rides History on Map**: Company representatives should be able to view their ride history on a map displaying the source location, destination, and actual end location of each completed ride.

- **User Story**: As a company representative, I want to visualize drivers rides history on a map to gain insights into driver past routes, source locations, destinations, and actual end points.
- **Acceptance Criteria**:
  - The system shall maintain a record of completed rides, including the source location, destination, and actual end location for each ride.
  - The company representative shall have a dedicated section or interface within the system to access and view their ride history on a map.
  - On the map, each completed ride shall be represented by a distinct marker or visual element, indicating the source location, destination, and actual end location.

11. View Company Drivers' Past Rides Reports: Company representatives should have access to view reports generated based on the past rides of company drivers.

- **User Story**: As a company representative, I want to access reports that summarize the past rides of company drivers for monitoring and analysis purposes.
- **Acceptance Criteria**:
  - The system shall provide a reports section or page where company representatives can access, and view reports related to company drivers' past rides.
  - Company representatives should be able to access report information such as driver distraction and drowsiness Information.

12. View Company Drivers' History Report: Company representatives should be able to view a history report that summarizes drivers' overall ride performance over their use of the system.

- **User Story**: As a company representative, I want to access a report that provides insights into driver's overall ride performance across multiple rides.
- **Acceptance Criteria**:

- The system shall generate a history report that aggregates data from multiple rides and presents an overview of the driver's performance.
- Company representatives should be able to access and view the driver's history reports, which may include metrics such as average ratings, distances traveled, and other relevant statistics.

13. View Saved Radars: Company representatives should be granted access to a list of radars, displayed on an integrated map, which are stored in the system's database.

- **User Story**: As a company representative, I want to view a list of radars associated with the system on an integrated map to ensure effective monitoring and management.
- **Acceptance Criteria**:
  - The system shall provide a dedicated section or page where company representatives can access the list of radars.
  - The list of radars shall be visually represented on an integrated map within the system's interface.
  - Each radar entry on the map shall include relevant information, such as location coordinates, radar speed limit, and any additional details stored in the system's database.

14. Logout: Company representatives should have the option to log out of the system, terminating their session.

- **User Story**: As a company representative, I want to log out of the system to ensure the security of my account.
- **Acceptance Criteria**:
  - The system shall provide a logout function or button that allows company representatives to end their current session.
  - After logging out, the system shall redirect the user to an appropriate page or display a logout confirmation message.

Admin Role

1. **Login**: The system should allow administrators to log in using the email and password provided by the company.
    - **User Story**: As an admin, I want to log in to the system using my credentials to access administrative functionalities.
    - **Acceptance Criteria**:
        - The system shall provide a login screen where administrators can enter their email and password.
        - Upon successful login, the system shall authenticate the credentials and grant access to admin-specific features.
2. **View Profile**: Admins should be able to view their profile information within the system.
    - **User Story**: As an admin, I want to view my profile information to retrieve and review my details.
    - **Acceptance Criteria**:
        - The system shall provide a dedicated section or page to display the admin's profile details.
        - Admins should be able to access and view information such as their name, contact details, and other relevant data.
3. **View Company Information**: Admins should have access to view the company's information within the system.
    - **User Story**: As an admin, I want to view the company's information to stay updated about the organization.
    - **Acceptance Criteria**:
        - The system shall provide a section or page where admins can access and view the company's details such as name, address, and other relevant information.
4. **Add Company Drivers**: Admins should be able to add new drivers to the system on behalf of the company.

- **User Story**: As an admin, I want to add new drivers to the system for company-related operations.
- **Acceptance Criteria**:
  - The system shall provide a form or interface where admins can enter the necessary details for adding a new company driver.
  - After successful submission, the system shall create a new driver account associated with the company.

5. **View Company Drivers**: Admins should have access to view a list of drivers associated with the company within the system.
   - **User Story**: As an admin, I want to view the list of drivers associated with the company for management purposes.
   - **Acceptance Criteria**:
     - The system shall provide a designated section or page to display a list of company drivers.
     - Admins should be able to view details such as driver names, contact information, and other relevant data.
6. **Edit Company Drivers Info**: Admins should be able to edit and update the information of company drivers in the system.
   - **User Story**: As an admin, I want to modify the details of company drivers to keep the information up to date.
   - **Acceptance Criteria**:
     - The system shall provide an interface or form where admins can edit specific fields of a company driver's information.
     - After making changes, the system shall save the updated information and reflect the modifications in the driver's profile.
7. **Delete Company Drivers**: Admins should have the capability to delete company drivers from the system.
   - **User Story**: As an admin, I want to remove drivers from the system when necessary.

- **Acceptance Criteria**:
  - The system shall provide a mechanism or action to delete a company driver's account and associated data.
  - After confirmation, the system shall permanently remove the driver's account from the system.

8. **View Company Drivers' Past Rides**: Admins should have access to view the past ride information of company drivers.
   - **User Story**: As an admin, I want to access the past ride information of company drivers for monitoring and analysis purposes.
   - **Acceptance Criteria**:
     - The system shall provide a section or page where admins can view the past ride details of company drivers.
     - Admins should be able to access information such as ride dates, durations, destinations, and other relevant data.
9. **View Company Drivers' Rides History on Map**: Admins should be able to view their ride history on a map displaying the source location, destination, and actual end location of each completed ride.

   - **User Story**: As an admin, I want to visualize drivers ride history on a map to gain insights into driver rides past routes, source locations, destinations, and actual end points.
   - **Acceptance Criteria**:
     - The system shall maintain a record of completed rides, including the source location, destination, and actual end location for each ride.
     - The admin shall have a dedicated section or interface within the system to access and view their ride history on a map.

- On the map, each completed ride shall be represented by a distinct marker or visual element, indicating the source location, destination, and actual end location.

10. **View Company Drivers' Past Rides Reports**: Admins should have access to view reports generated based on the past rides of company drivers.
    - **User Story**: As an admin, I want to access reports that summarize the past rides of company drivers for monitoring and analysis purposes.
    - **Acceptance Criteria**:
        - The system shall provide a reports section or page where admins can access, and view reports related to company drivers' past rides.
        - Admins should be able to access report information such as driver distraction and drowsiness Information.

11. **View Company Drivers' History Report:** Admins should be able to view a history report that summarizes drivers' overall ride performance over their use of the system.
    - **User Story**: As an admin, I want to access a report that provides insights into driver's overall ride performance across multiple rides.
    - **Acceptance Criteria**:
        - The system shall generate a history report that aggregates data from multiple rides and presents an overview of the driver's performance.
        - Admins should be able to access and view driver's history reports, which may include metrics such as average ratings, distances traveled, and other relevant statistics.
12. **Add New Radars**: Admins should have the ability to add new radars to the system's database, including radar coordinates and speed limit information.

- **User Story**: As an admin, I want to be able to add new radars to the system's database to ensure accurate tracking and monitoring of radar locations and speed limits.
- **Acceptance Criteria**:
  - The system shall provide a feature on an integrated map that allows company representatives to input radar information for adding new radars to the database.
  - The radar information to be provided should include accurate coordinates (latitude and longitude) to specify the exact location of the radar. Along with the speed limit associated with the radar.
  - The system shall validate and verify the radar information provided, ensuring that the coordinates are within the acceptable range and the speed limit is within a reasonable range.
  - Upon successfully adding a new radar to the system's database, the system shall store the radar information securely and associate it with the relevant geographical location and speed limit.

13. View Saved Radars on Maps: Admins should be granted access to a list of radars, displayed on an integrated map, which are stored in the system's database.

- **User Story**: As an admin, I want to view a list of radars associated with the system on an integrated map to ensure effective monitoring and management.
- **Acceptance Criteria**:
  - The system shall provide a dedicated section or page where admins can access the list of radars.
  - The list of radars shall be visually represented on an integrated map within the system's interface.
  - Each radar entry on the map shall include relevant information, such as location coordinates, radar speed limit, and any additional details stored in the system's database.

14. **Delete Saved Radars**: Admins should have the ability to delete radars that are saved in the system's database.

    - **User Story**: As an admin I want to be able to delete radars from the system's database to ensure accurate and up-to-date radar information.
    - **Acceptance Criteria**:
        - The system shall provide a feature or interface that allows company representatives to select and delete radars from the system's database.
        - Upon selecting a radar for deletion, the system shall display a confirmation prompt to ensure the representative's intention to delete the radar.
        - Once confirmed, the system shall remove the selected radar from the database, permanently deleting its information.
        - The system shall validate the deletion process, ensuring that only authorized company admins can delete radars from the database.

15. **Logout**: Admins should have the option to log out of the system, terminating their session.
    - **User Story**: As an admin, I want to log out of the system to ensure the security of my account and data.
    - **Acceptance Criteria**:
        - The system shall provide a logout function or button that allows admins to end their current session.
        - After logging out, the system shall redirect the user to an appropriate page or display a logout confirmation message.

Driver Role

1. **Register**: Drivers should be able to register in the system.

- **User Story**: As a driver, I want to register in the system to access driver-specific functionalities.
- **Acceptance Criteria**:
  - The system shall provide a registration process for drivers, allowing them to create an account by providing necessary details.
  - Upon successful registration, the system shall generate a unique driver ID for identification in the Authentication API.

2. **Login**: Drivers should be able to log in to the system using their email and password or the credentials provided by company admins.
   - **User Story**: As a driver, I want to log in to the system to access driver-related features and information.
   - **Acceptance Criteria**:
     - The system shall provide a login screen where drivers can enter their email and password, or the credentials provided by company admins.
     - Upon successful login, the system shall authenticate the credentials and grant access to driver-specific functionalities.
3. **View Profile**: Drivers should be able to view their profile information within the system.
   - **User Story**: As a driver, I want to view my profile information to retrieve and review my details.
   - **Acceptance Criteria**:
     - The system shall provide a dedicated section or page where drivers can view their profile information.
     - Drivers should be able to access and view details such as their name, rating, join date, and other relevant data.
4. **Edit Information**: Drivers should have the ability to edit and update their information in the system.
   - **User Story**: As a driver, I want to modify my information to keep it up to date.
   - **Acceptance Criteria**:

- The system shall provide an interface or form where drivers can edit specific fields of their information.
- After making changes, the system shall save the updated information and reflect the modifications in the driver's profile.

5. **Delete Account**: Drivers should have the capability to delete their account from the system.
   - **User Story**: As a driver, I want to remove my account and information from the system when necessary.
   - **Acceptance Criteria**:
     - The system shall provide a mechanism or action to delete a driver's account and associated data.
     - After confirmation, the system shall permanently remove the driver's account from the system.
6. **View Past Rides**: Drivers should be able to view information about their past rides within the system.
   - **User Story**: As a driver, I want to access details about my previous rides for review and reference.
   - **Acceptance Criteria**:
     - The system shall provide a section or page where drivers can access information such as ride dates, durations, and destinations for their past rides.
7. **View Rides History on Map**: Drivers should be able to view their ride history on a map displaying the source location, destination, and actual end location of each completed ride.

   - **User Story**: As a driver, I want to visualize my ride history on a map to gain insights into my past routes, source locations, destinations, and actual end points.
   - **Acceptance Criteria**:

- The system shall maintain a record of completed rides, including the source location, destination, and actual end location for each ride.
- The driver shall have a dedicated section or interface within the system to access and view their ride history on a map.
- On the map, each completed ride shall be represented by a distinct marker or visual element, indicating the source location, destination, and actual end location.

8. **View Past Rides Reports**: Drivers should be able to view reports summarizing their past rides in the system.
    - **User Story**: As a driver, I want to access reports that provide insights into my previous rides for analysis.
    - **Acceptance Criteria**:
        - The system shall generate reports that include information such as ride ratings, dates, and graphical representations of ride-related metrics (e.g., time, distance, speed, drowsy and distracted times).
        - Drivers should be able to access and view these reports for their past rides.
9. **View Saved Radars on Maps**: Drivers should be granted access to a list of radars, displayed on an integrated map, which are stored in the system's database.
    - **User Story**: As a driver, I want to view a list of radars associated with the system on an integrated map to ensure effective monitoring and management.
    - **Acceptance Criteria**:
        - The system shall provide a dedicated section or page where drivers can access the list of radars.
        - The list of radars shall be visually represented on an integrated map within the system's interface.
        - Each radar entry on the map shall include relevant information, such as location coordinates, radar speed limit, and any additional details stored in the system's database.
10. **Start New Rides**: Drivers should have the ability to start a new ride in the system.

- **User Story**: As a driver, I want to initiate a new ride to my destination.
- **Acceptance Criteria**:
  - The system shall provide an interface or button that allows drivers to start a new ride.
  - Upon starting a ride, the system shall record the start time and update the driver's current status.

11. **View Current Location on Maps**: Drivers should be able to view their current location on maps within the system.
    - **User Story**: As a driver, I want to see my current location on maps to be aware of my position before the ride.
    - **Acceptance Criteria**:
      - The system shall display the driver's current location on integrated maps.
      - The driver's current location should be accurately reflected on the maps interface.
12. **Search for Destination**: Drivers should be able to search for a destination on maps and set it as the destination for their ride.
    - **User Story**: As a driver, I want to search for a destination and set it as my ride's destination for navigation purposes.
    - **Acceptance Criteria**:
      - The system shall integrate maps functionality that allows drivers to search for a destination.
      - After selecting a destination, the system shall store the destination information for navigation and route calculation purposes.
13. **Display Optimal Route and Radar Locations on Maps**: Drivers should have the capability to view the optimal route on maps within the system, along with the locations of radars they will encounter along the route.
    - **User Story**: As a driver, I want to have access to the most optimal route to my destination displayed on maps, while also being aware of radar locations along the route.

- **Acceptance Criteria**:
  - The system shall acquire and display the best route based on factors such as distance, traffic conditions, and other relevant criteria.
  - The displayed route on maps should provide accurate directions and guidance to the driver.
  - Along with the route, the system shall mark the locations of radars that the driver will encounter during the ride, along with radar speed limit.

14. **Redirect to Google Maps for Navigation**: Drivers should have the ability to be redirected to Google Maps for navigation to their destination.
    - **User Story**: As a driver, I want the ability to use Google Maps for navigation to ensure accurate and reliable directions.
    - **Acceptance Criteria**:
      - The system shall provide a feature or button that allows drivers to switch to Google Maps for navigation purposes.
      - When selected, the system shall open Google Maps with the destination pre-set for navigation.
15. **View System Floating Bubble**: When the driver is redirected to Google Maps for navigation, the system should display a floating bubble that indicates the system is still running in the background. The floating bubble can be tapped to bring the driver back to the app and provide quick access to end the ride.
    - **User Story**: As a driver, I want a floating bubble to indicate that the system is running in the background while I navigate with Google Maps, so I can easily return to the app and end the ride when needed.
    - **Acceptance Criteria**:
      - When the driver is redirected to Google Maps for navigation, the system shall display a floating bubble on the device's screen, indicating that the system is still active and running in the background.

- The floating bubble should be visible and accessible from within the Google Maps interface, ensuring that the driver can easily locate and interact with it.
- When the driver taps the floating bubble, it shall bring the driver back to the system's app interface, allowing for quick access to essential features, such as ending the ride.

16. **Receive Staged Alerts for Unsafe Driving State**: Drivers should receive alerts when they show signs of drowsiness, distraction with the phone, or distraction from the road for a specific period.
    - **User Story**: As a driver, I want to receive staged alerts to ensure my safety and maintain focus during the ride.
    - **Acceptance Criteria**:
        - The system shall continuously monitor driver's behavior and detect signs of drowsiness or distraction based on predefined criteria.
        - When a specific threshold is reached, the system shall trigger alerts to notify the driver and encourage them to regain focus on the road.
        - The alerts shall be distinct and informative, providing clear instructions and warnings to the driver.
        - The driver shall have the option to acknowledge or dismiss the alerts based on their judgment and take appropriate actions to ensure safe driving.
17. **Receive Staged Alerts before Approaching Radars**: Drivers should receive alerts when approaching radars along with specialized alerts when they exceed the speed limit or when they were in a unsafe driving state before approaching radars.
    - **User Story**: As a driver, I want to be alerted when approaching radars, exceeding the speed limit, or when I exhibit unsafe driving behavior before approaching radars, in order to ensure compliance with traffic regulations and enhance safety.
    - **Acceptance Criteria**:

- The system shall continuously monitor the driver's position and speed to detect the proximity to radars and speed limit violations.
- When approaching a radar or exceeding the speed limit, the system shall trigger alerts to notify the driver and prompt them to adjust their speed.
- Additionally, if the system detects unsafe driving behavior (e.g., drowsiness, distraction) before approaching radars, it shall also trigger alerts to notify the driver and encourage them to regain focus on the road.
- The alerts shall be distinct and informative, providing clear instructions and warnings to the driver.
- The driver shall have the option to acknowledge or dismiss the alerts based on their judgment and take appropriate actions to ensure safe driving.

18. **End the Ride**: Drivers should have the capability to end their current ride within the system and be redirected to the mobile app to view the report summarizing the details of the ended ride.
    - **User Story**: As a driver, I want to mark the completion of a ride and have access to a comprehensive report summarizing the ride's details.
    - **Acceptance Criteria**:
        - The system shall provide an interface or button that allows drivers to end the ride.
        - Upon ending the ride, the system shall record the end time and finalize ride-related calculations and statistics.
        - To end the ride, the system provides a floating bubble that redirects the driver to the mobile app when tapped, where they can access the report for the ended ride.
19. **Get Rating for Overall Rides Performance**: Drivers should receive a rating that represents their overall ride performance within the system.
    - **User Story**: As a driver, I want to receive a rating that reflects my overall performance as a driver.

- **Acceptance Criteria**:
  - The system shall calculate and assign a rating to each driver based on factors such as driver safe driving state, adherence to regulations, and other relevant criteria.
  - Drivers should be able to view their rating within their profile in the system.

20. **View History Report for Overall Rides Performance**: Drivers should be able to view a history report that summarizes their overall ride performance over their use of the system.
    - **User Story**: As a driver, I want to access a report that provides insights into my overall ride performance across multiple rides.
    - **Acceptance Criteria**:
      - The system shall generate a history report that aggregates data from multiple rides and presents an overview of the driver's performance.
      - Drivers should be able to access and view this history report, which may include metrics such as average ratings, distances traveled, and other relevant statistics.
21. **Logout**: Drivers should have the option to log out of the system, terminating their session.
    - **User Story**: As a driver, I want to log out of the system to ensure the security of my account and data.
    - **Acceptance Criteria**:
      - The system shall provide a logout function or button that allows drivers to end their current session.
      - After logging out, the system shall redirect the user to an appropriate page or display a logout confirmation message.

3.3.2 Non-Functional Requirements

1. **Performance**: The system should demonstrate high performance in processing deep learning monitoring and maps monitoring tasks in less than one second for deep learning monitoring and five seconds for maps monitoring tasks.
2. **Accuracy**: The system's deep learning monitoring and maps monitoring components should exhibit a high level of accuracy in their results and predictions, any prediction accuracy lower than 90% is not acceptable .
3. **Responsiveness**: The system shall respond promptly to user actions and interactions, providing real-time updates and feedback. This includes quick loading of pages, seamless navigation, and immediate response to user input, ensuring a smooth and interactive user experience.
4. **Usability**: The system shall be designed with a user-centric approach, prioritizing intuitive interfaces, clear navigation, and consistent layouts. This ensures that users can easily understand and navigate the system, perform tasks efficiently, and access information without confusion or unnecessary complexity.
5. **Reliability**: The system's deep learning monitoring and maps monitoring functionalities should operate reliably, ensuring consistent and dependable performance.
6. **Transparency**: The system shall provide transparency in its processes and operations, enabling users to understand how data is collected, stored, and processed. This includes clear documentation, visibility into system activities, and adherence to data privacy regulations.
7. **Compatibility**: The system shall be compatible with a wide range of devices, operating systems, and web browsers, ensuring seamless access and functionality across different platforms. It should adapt to various screen sizes and resolutions, providing a consistent user experience regardless of the device used.
8. **Privacy**: The system shall prioritize the privacy and confidentiality of user data, implementing appropriate measures to safeguard personal information. This includes secure

storage, encryption, user consent mechanisms, and adherence to privacy laws and regulations.

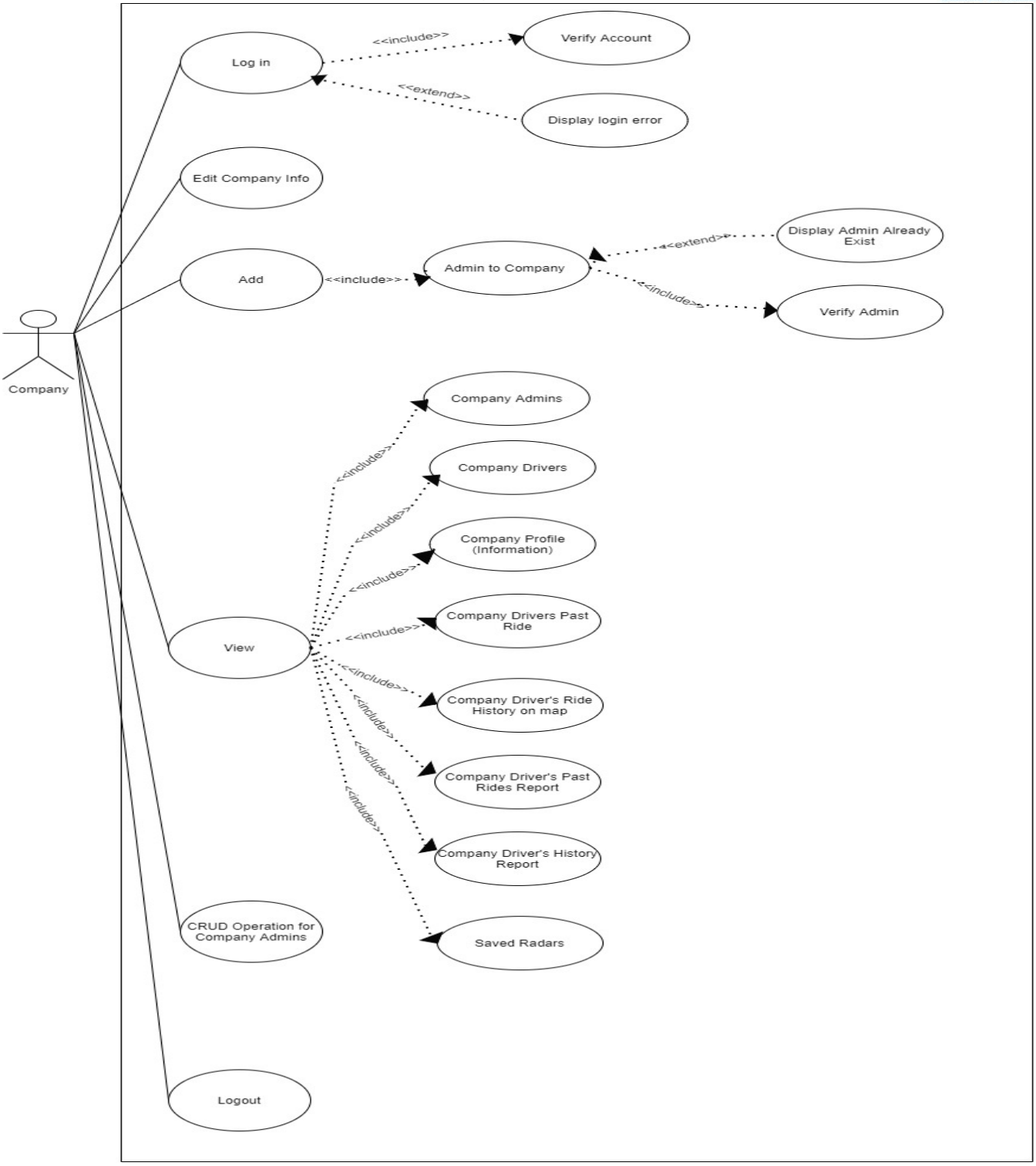
9. **Security**: The system should employ robust security measures to safeguard against unauthorized access, data breaches, and other potential security threats.
10. **Testability**: The system shall be designed and implemented in a way that allows effective testing and quality assurance processes. This includes providing necessary tools, documentation, and interfaces to facilitate testing, as well as ensuring modularity and code maintainability for efficient bug fixing and updates.

3.4 Use Case Diagrams

The use case diagrams provide a visual representation of the various use cases for the three roles in our system: the driver, admin, and company representatives. These diagrams outline the functionalities and interactions available to each role, facilitating a comprehensive understanding of the system's capabilities.

3.4.1 Company Role Use Case Diagram.

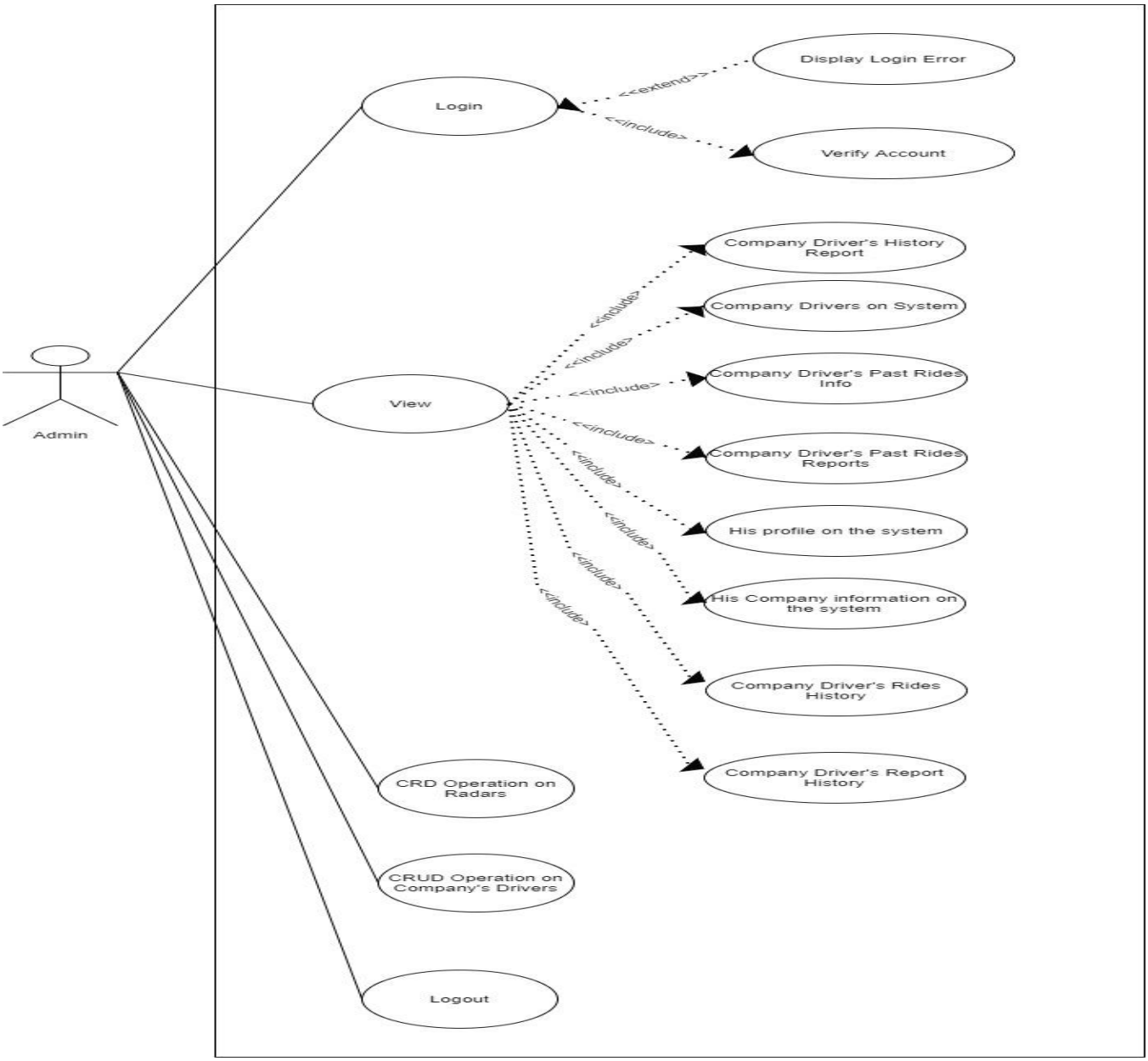
The use case diagram for company representatives outlines the specific use cases relevant to their role through the web application. These include accessing driver reports associated with their respective companies, evaluating driver performance, and generating company-specific reports. By visualizing the use cases tailored to company representatives, the diagram emphasizes their unique interactions within the system and the value they derive from the provided functionalities.



3.4.2 Admin Role Use Case Diagram

The admin role use case diagram presents the functionalities available to system administrators through the web application. These functionalities

include managing user accounts, monitoring driver reports, generating performance analysis reports, and accessing system settings. The diagram visually represents the key tasks and responsibilities of the admin role, providing a clear overview of their interactions with the system and the scope of their administrative capabilities.

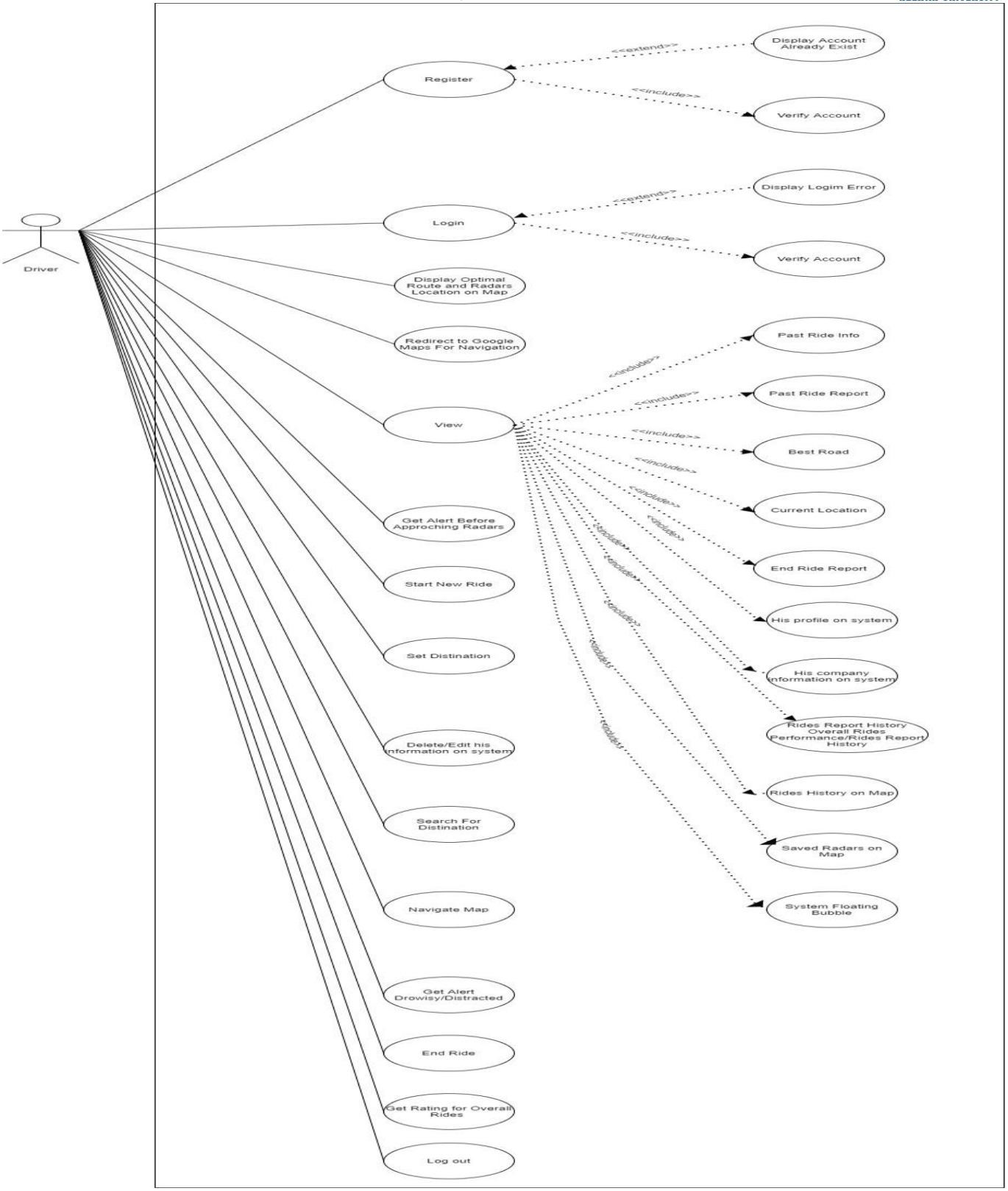


3.4.3 Driver Role Use Case Diagram

Lastly, For the driver role, the use case diagram illustrates the primary actions and features accessible to drivers through the mobile application. These include registering an account, logging in, initiating a trip, receiving real-time alerts for distractions or drowsiness, and accessing trip reports. By capturing the essential interactions between the driver and the system, the use case diagram highlights the driver's involvement and the features that enhance their driving experience and safety.

These use case diagrams serve as a valuable reference for understanding the system's functionality from the perspective of each role. They offer insights into the key features and interactions specific to each user group, allowing for a comprehensive view of the system's capabilities and facilitating effective communication and collaboration among all stakeholders involved.

Through the development of these use case diagrams, we ensure that the AI-Powered Driver Monitoring System fulfills the diverse needs of drivers, admins, and company representatives, contributing to a safer and more efficient driving experience for all users.

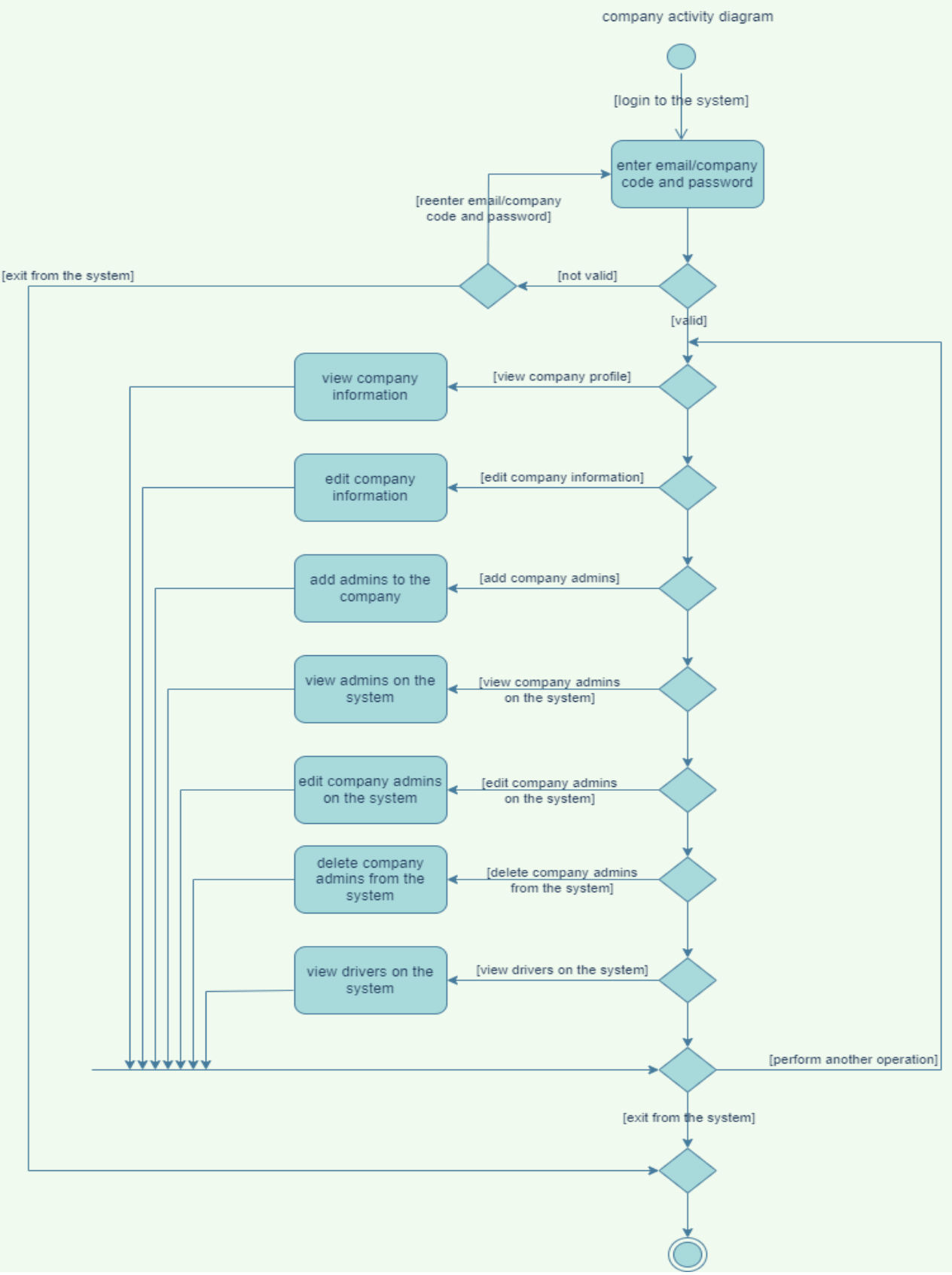

Driver
Register
Login
Display Optimal Route and Radars Location on Map
Redirect to Google Maps For Navigation
View
Get Alert Before Approching Radars
Start New Ride
Set Distination
Delete/Edit his information on system
Search For Distination
Navigate Map
Get Alert Drowisy/Distracted
End Ride
Get Rating for Overall Rides
Log out
Display Account Already Exist
Verify Account
Display Logim Error
Verify Account
Past Ride Info
Past Ride Report
Best Road
Current Location
End Ride Report
His profile on system
His company information on system
Rides Report History Overall Rides Performance/Rides Report History
Rides History on Map
Saved Radars on Map
System Floating Bubble
<<extend>>
<<include>>
<<extend>>
<<include>>

3.5 Activity Diagrams

The activity diagrams provide a visual representation of the activities performed by the three roles in our system: the driver, admin, and company representatives. These diagrams illustrate the flow of actions and decision points within each role, allowing for a better understanding of the system's functionality and the sequence of activities undertaken by users.

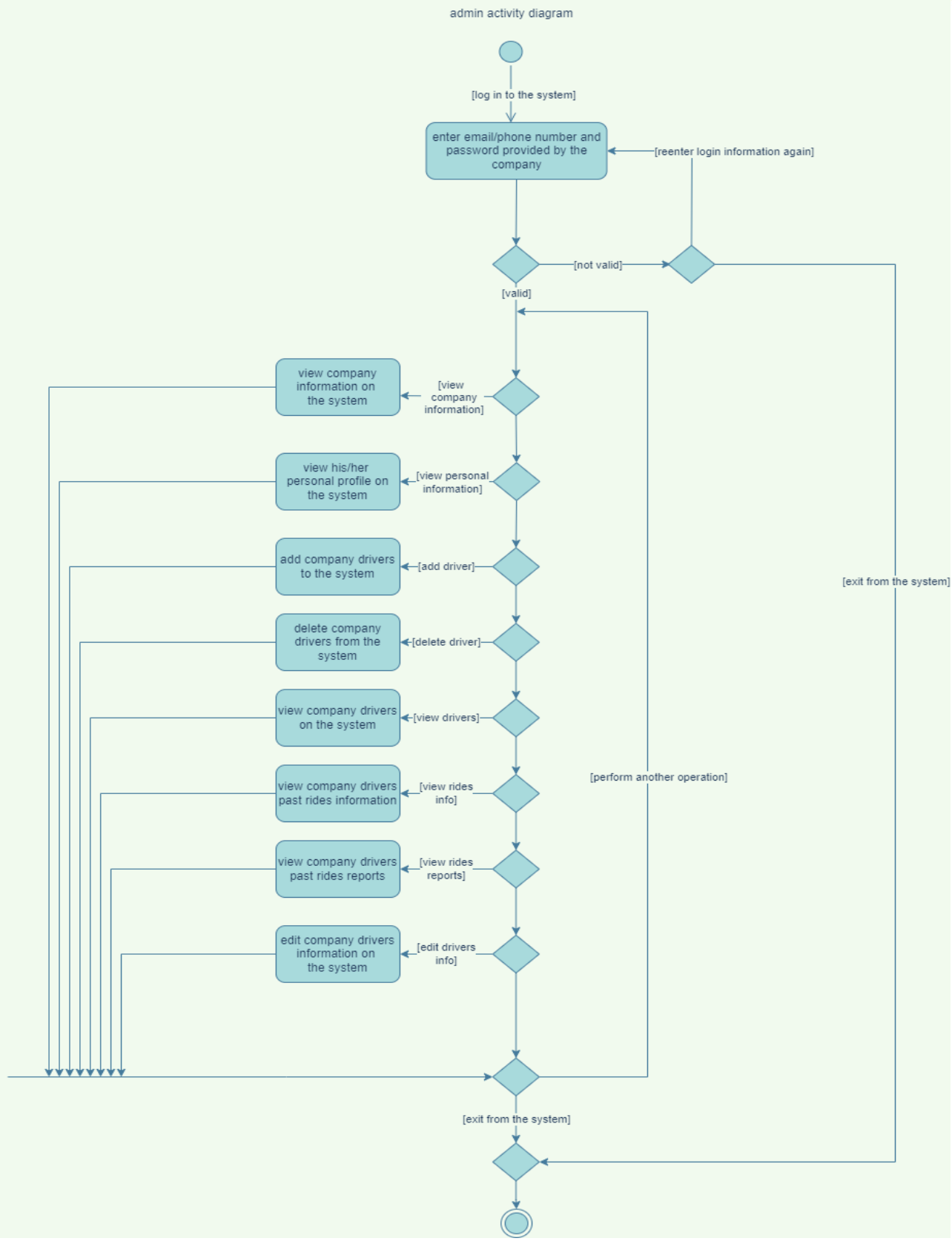
3.5.1 Company Role Activity Diagram

The activity diagram for company representatives demonstrates the activities specific to their role within the web application. These activities include accessing driver reports, evaluating performance metrics, generating company-specific reports, and interacting with the system's features relevant to their company's operations. By visually presenting the flow of activities, the diagram provides a clear overview of the tasks performed by company representatives and their interaction with the system.


company activity diagram
[login to the system]
enter email/company code and password
[reenter email/company code and password]
[not valid]
[exit from the system]
[valid]
view company information
[view company profile]
edit company information
[edit company information]
add admins to the company
[add company admins]
view admins on the system
[view company admins on the system]
edit company admins on the system
[edit company admins on the system]
delete company admins from the system
[delete company admins from the system]
view drivers on the system
[view drivers on the system]
[perform another operation]
[exit from the system]

3.5.2 Admin Role Activity Diagram

The activity diagram for the admin role showcases the activities performed within the web application. These activities involve managing user accounts, monitoring driver reports, generating performance analysis reports, and adjusting system settings. By visually mapping out the sequence of actions and decision points, the activity diagram facilitates a comprehensive understanding of the admin role's workflow and their involvement in the system.

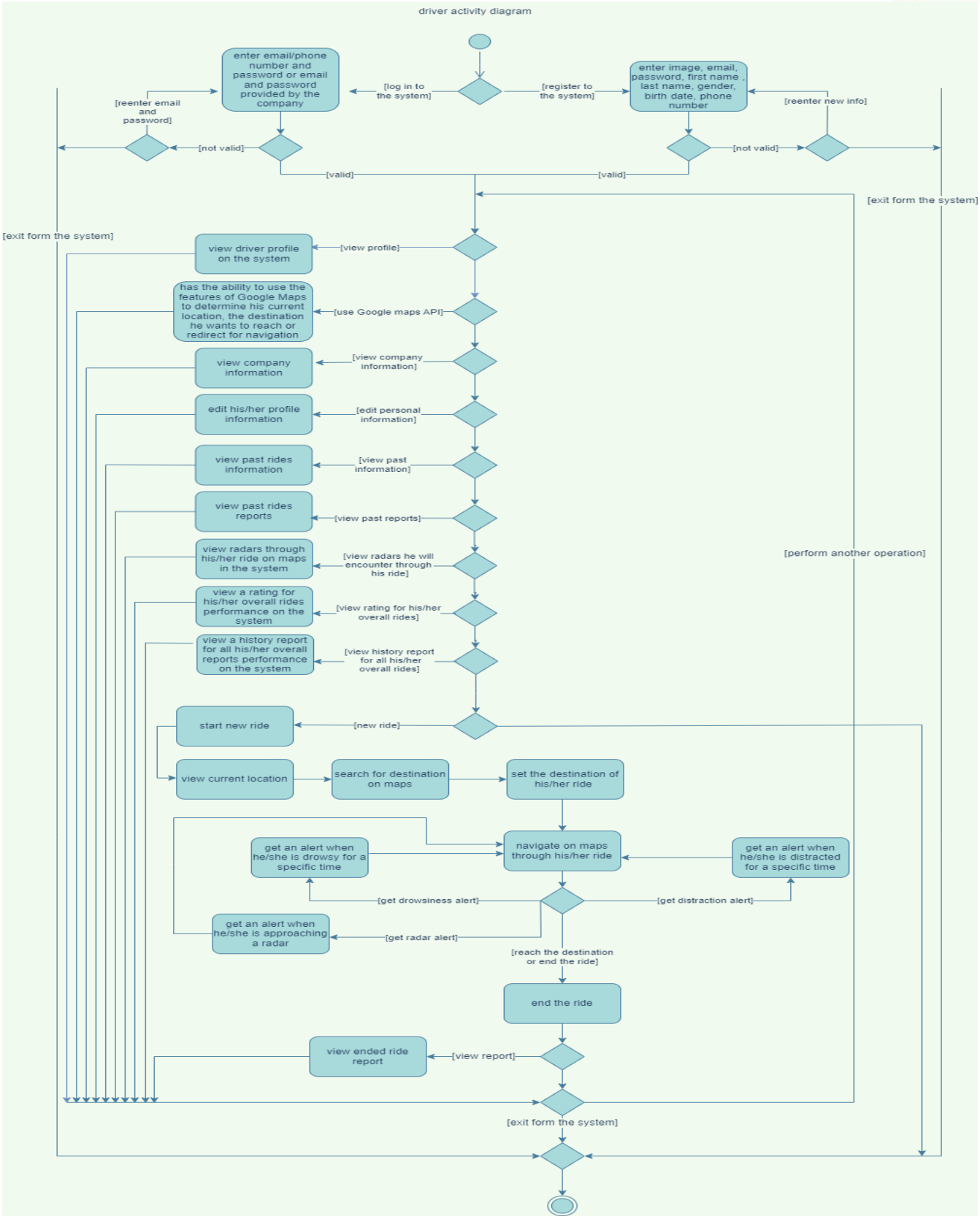

admin activity diagram
[log in to the system]
enter email/phone number and password provided by the company
[reenter login information again]
[not valid]
[valid]
view company information on the system
[view company information]
view his/her personal profile on the system
[view personal information]
add company drivers to the system
[add driver]
delete company drivers from the system
[delete driver]
view company drivers on the system
[view drivers]
view company drivers past rides information
[view rides info]
view company drivers past rides reports
[view rides reports]
edit company drivers information on the system
[edit drivers info]
[perform another operation]
[exit from the system]
[exit from the system]

3.5.3 Driver Role Activity Diagram

Lastly, For the driver role, the activity diagram depicts the series of actions involved in their interaction with the mobile application. These activities include logging in, initiating a ride, receiving real-time alerts, responding to alerts, and accessing ride reports. By outlining the flow of activities, decision points, and potential branches, the activity diagram highlights the step-by-step process followed by drivers and enables a clear understanding of their engagement with the system.

These activity diagrams serve as a valuable tool for understanding the system's functionality from the perspective of each role. They provide a visual representation of the activities performed by drivers, admins, and company representatives, enabling stakeholders to grasp the sequence of actions, decision points, and potential paths within the system.

By developing these activity diagrams, we ensure that the AI-Powered Driver Monitoring System aligns with the needs and workflows of its users. The diagrams contribute to a more intuitive understanding of the system's operations, fostering efficiency, and enabling effective communication and collaboration among all stakeholders involved.

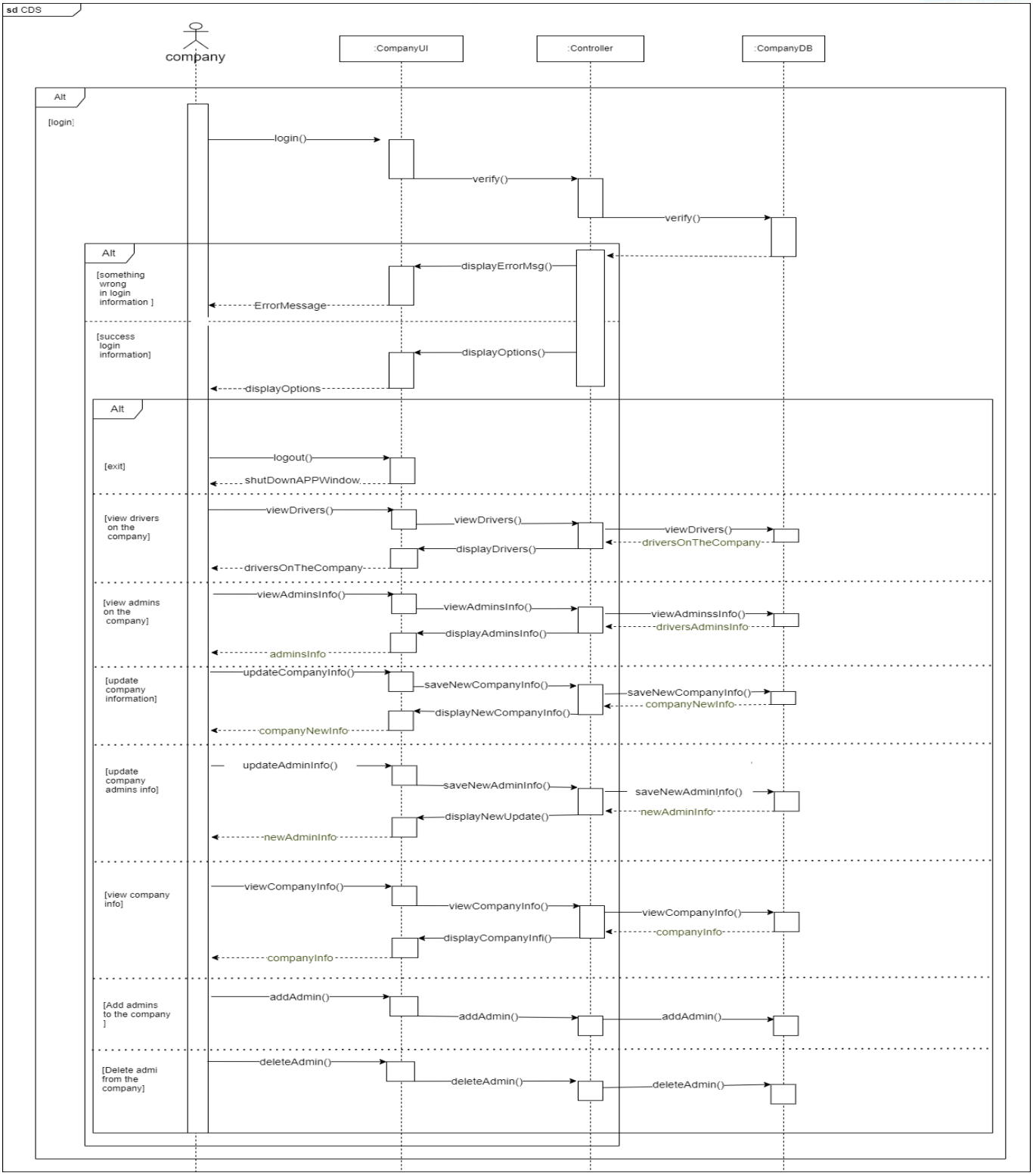

driver activity diagram
enter email/phone number and password or email and password provided by the company
[log in to the system]
[register to the system]
enter image, email, password, first name , last name, gender, birth date, phone number
[reenter email and password]
[not valid]
[reenter new info]
[not valid]
[valid]
[valid]
[exit form the system]
[exit form the system]
view driver profile on the system
[view profile]
has the ability to use the features of Google Maps to determine his current location, the destination he wants to reach or redirect for navigation
[use Google maps API]
view company information
[view company information]
edit his/her profile information
[edit personal information]
view past rides information
[view past information]
view past rides reports
[view past reports]
view radars through his/her ride on maps in the system
[view radars he will encounter through his ride]
[perform another operation]
view a rating for his/her overall rides performance on the system
[view rating for his/her overall rides]
view a history report for all his/her overall reports performance on the system
[view history report for all his/her overall rides]
start new ride
[new ride]
view current location
search for destination on maps
set the destination of his/her ride
get an alert when he/she is drowsy for a specific time
navigate on maps through his/her ride
get an alert when he/she is distracted for a specific time
[get drowsiness alert]
[get distraction alert]
get an alert when he/she is approaching a radar
[get radar alert]
[reach the destination or end the ride]
end the ride
view ended ride report
[view report]
[exit form the system]

3.6 Sequence Diagrams

The sequence diagrams provide a detailed representation of the interactions and sequence of messages exchanged between the system's three roles: driver, admin, and company representatives. These diagrams highlight the chronological order of events and the communication flow between actors, allowing for a better understanding of the system's behavior and the interactions among different components.

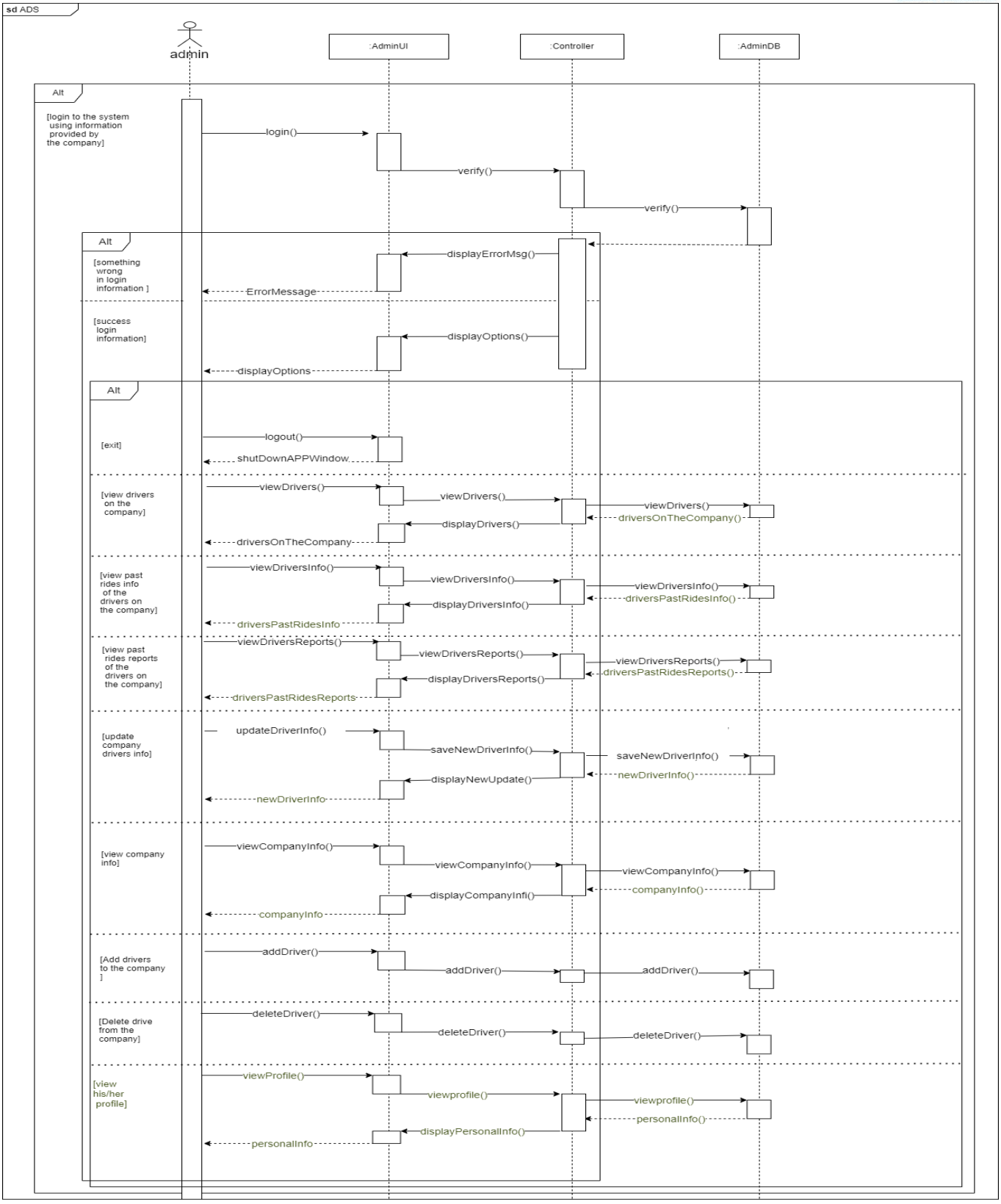
3.6.1 Company Role Sequence Diagram

The sequence diagram for company representatives captures the interactions between their web interface and the system's components. It demonstrates the sequence of messages exchanged during activities like accessing driver reports, evaluating performance metrics, generating company-specific reports, and utilizing features specific to their company's operations. By visualizing the communication flow, dependencies, and synchronization between the representative interface and other system components, the sequence diagram provides a detailed view of their involvement in the system.


sd CDS
company
:CompanyUI
:Controller
:CompanyDB
Alt
[login]
login()
verify()
verify()
Alt
[something wrong in login information]
displayErrorMsg()
ErrorMessage
[success login information]
displayOptions()
displayOptions
Alt
[exit]
logout()
shutDownAPPWindow
[view drivers on the company]
viewDrivers()
viewDrivers()
viewDrivers()
driversOnTheCompany
displayDrivers()
driversOnTheCompany
[view admins on the company]
viewAdminsInfo()
viewAdminsInfo()
viewAdminssInfo()
driversAdminsInfo
displayAdminsInfo()
adminsInfo
[update company information]
updateCompanyInfo()
saveNewCompanyInfo()
saveNewCompanyInfo()
companyNewInfo
displayNewCompanyInfo()
companyNewInfo
[update company admins info]
updateAdminInfo()
saveNewAdminInfo()
saveNewAdminInfo()
newAdminInfo
displayNewUpdate()
newAdminInfo
[view company info]
viewCompanyInfo()
viewCompanyInfo()
viewCompanyInfo()
companyInfo
displayCompanyInfi()
companyInfo
[Add admins to the company]
addAdmin()
addAdmin()
addAdmin()
[Delete admi from the company]
deleteAdmin()
deleteAdmin()
deleteAdmin()

3.6.2 Admin Role Sequence Diagram

The sequence diagram for the admin role showcases the sequence of interactions between the admin's web application and the system's components. It represents the flow of messages exchanged during activities such as managing user accounts, monitoring driver reports, generating performance analysis reports, and adjusting system settings. By illustrating the communication flow and dependencies between the admin application and other system components, the sequence diagram enables a comprehensive understanding of the admin's role and their involvement in the system.

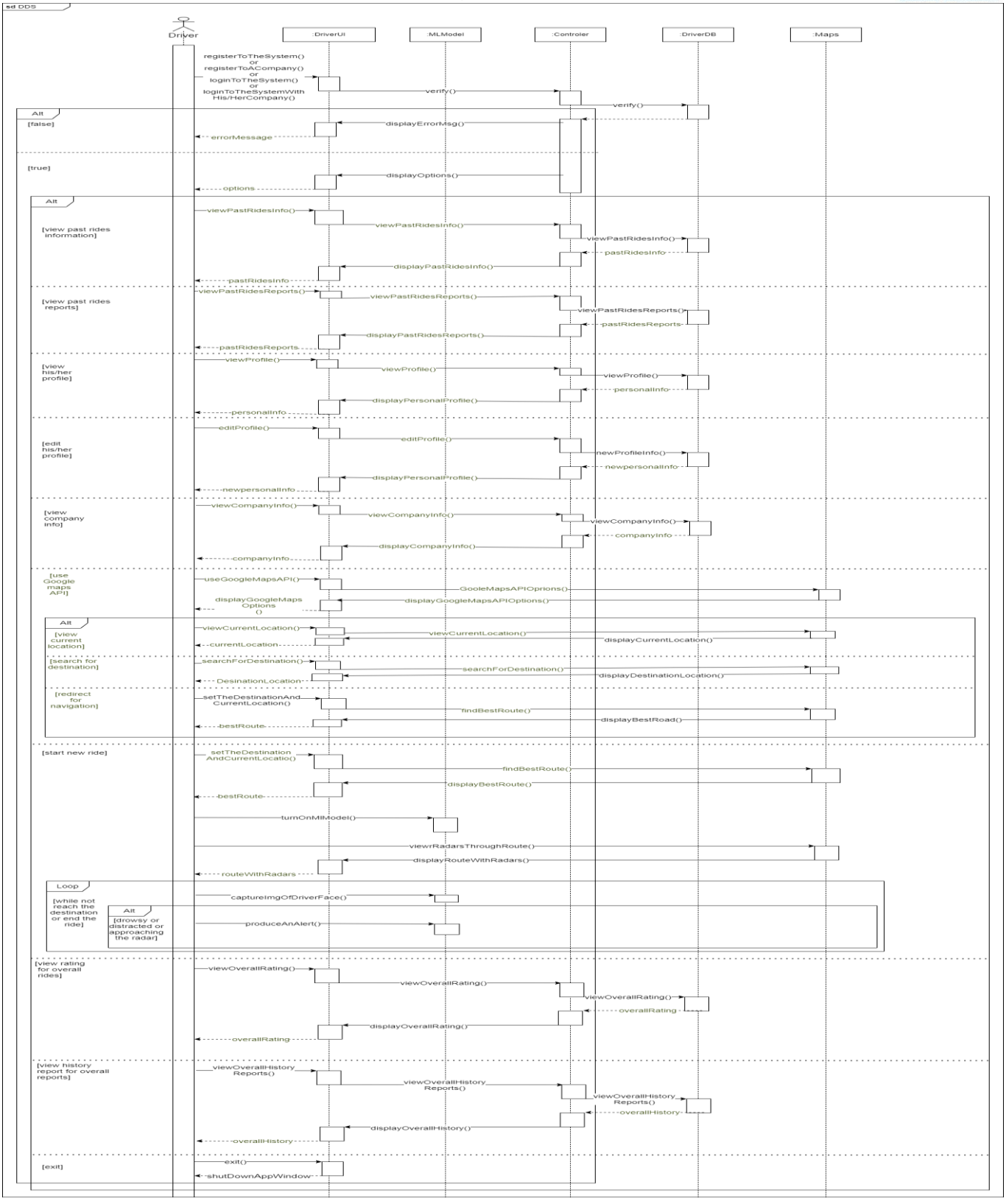

sd ADS
admin
:AdminUI
:Controller
:AdminDB
Alt
[login to the system using information provided by the company]
login()
verify()
verify()
Alt
[something wrong in login information]
displayErrorMsg()
ErrorMessage
[success login information]
displayOptions()
displayOptions
Alt
[exit]
logout()
shutDownAPPWindow
[view drivers on the company]
viewDrivers()
viewDrivers()
viewDrivers()
driversOnTheCompany()
displayDrivers()
driversOnTheCompany
[view past rides info of the drivers on the company]
viewDriversInfo()
viewDriversInfo()
viewDriversInfo()
driversPastRidesInfo()
displayDriversInfo()
driversPastRidesInfo
[view past rides reports of the drivers on the company]
viewDriversReports()
viewDriversReports()
viewDriversReports()
driversPastRidesReports()
displayDriversReports()
driversPastRidesReports
[update company drivers info]
updateDriverInfo()
saveNewDriverInfo()
saveNewDriverInfo()
newDriverInfo()
displayNewUpdate()
newDriverInfo
[view company info]
viewCompanyInfo()
viewCompanyInfo()
viewCompanyInfo()
companyInfo()
displayCompanyInfi()
companyInfo
[Add drivers to the company]
addDriver()
addDriver()
addDriver()
[Delete drive from the company]
deleteDriver()
deleteDriver()
deleteDriver()
[view his/her profile]
viewProfile()
viewprofile()
viewprofile()
personalInfo()
displayPersonalInfo()
personalInfo

3.6.3 Driver sequence diagram

The sequence diagram for the driver role illustrates the step-by-step sequence of interactions between the driver's mobile application and the system's components. It depicts the flow of messages exchanged during actions such as logging in, initiating a ride, receiving real-time alerts, responding to alerts, and accessing ride reports. By visualizing the order and timing of these interactions, the sequence diagram provides insights into how the driver interacts with the system and how different components collaborate to deliver the desired functionality.

These sequence diagrams serve as essential tools for understanding the dynamic behavior of the AI-Powered Driver Monitoring System from the perspective of each role. They offer a visual representation of the sequence of events, message exchanges, and dependencies among actors and system components, facilitating a comprehensive understanding of how the system operates in different scenarios.

By developing these sequence diagrams, we ensure that the AI-Powered Driver Monitoring System functions coherently and meets the expectations of its users. The diagrams contribute to a better understanding of the system's behavior, aiding in system design, implementation, and troubleshooting. They promote effective communication and collaboration among stakeholders, enabling them to work together seamlessly to achieve the system's objectives.


sd DDS
Driver
:DriverUI
:MLModel
:Controler
:DriverDB
:Maps
registerToTheSystem()
or
registerToACompany()
or
loginToTheSystem()
or
loginToTheSystemWith
His/HerCompany()
verify()
verify()
Alt
[false]
displayErrorMsg()
errorMessage
[true]
displayOptions()
options
Alt
viewPastRidesInfo()
[view past rides information]
viewPastRidesInfo()
viewPastRidesInfo()
pastRidesInfo
displayPastRidesInfo()
pastRidesInfo
viewPastRidesReports()
[view past rides reports]
viewPastRidesReports()
viewPastRidesReports()
pastRidesReports
displayPastRidesReports()
pastRidesReports
viewProfile()
[view his/her profile]
viewProfile()
viewProfile()
personalInfo
displayPersonalProfile()
personalInfo
editProfile()
[edit his/her profile]
editProfile()
newProfileInfo()
newpersonalInfo
displayPersonalProfile()
newpersonalInfo
viewCompanyInfo()
[view company info]
viewCompanyInfo()
viewCompanyInfo()
companyInfo
displayCompanyInfo()
companyInfo
[use Google maps API]
useGoogleMapsAPI()
GooleMapsAPIOprions()
displayGoogleMapsAPIOptions()
displayGoogleMaps Options ()
Alt
[view current location]
viewCurrentLocation()
viewCurrentLocation()
displayCurrentLocation()
currentLocation
[search for destination]
searchForDestination()
searchForDestination()
displayDestinationLocation()
DesinationLocation
[redirect for navigation]
setTheDestinationAnd CurrentLocation()
findBestRoute()
displayBestRoad()
bestRoute
[start new ride]
setTheDestination AndCurrentLocatio()
findBestRoute()
displayBestRoute()
bestRoute
turnOnMlModel()
viewRadarsThroughRoute()
displayRouteWithRadars()
routeWithRadars
Loop
[while not reach the destination or end the ride]
captureImgOfDriverFace()
Alt
[drowsy or distracted or approaching the radar]
produceAnAlert()
[view rating for overall rides]
viewOverallRating()
viewOverallRating()
viewOverallRating()
overallRating
displayOverallRating()
overallRating
[view history report for overall reports]
viewOverallHistory Reports()
viewOverallHistory Reports()
viewOverallHistory Reports()
overallHistory
displayOverallHistory()
overallHistory
[exit]
exit()
shutDownAppWindow

3.7 Class Diagrams

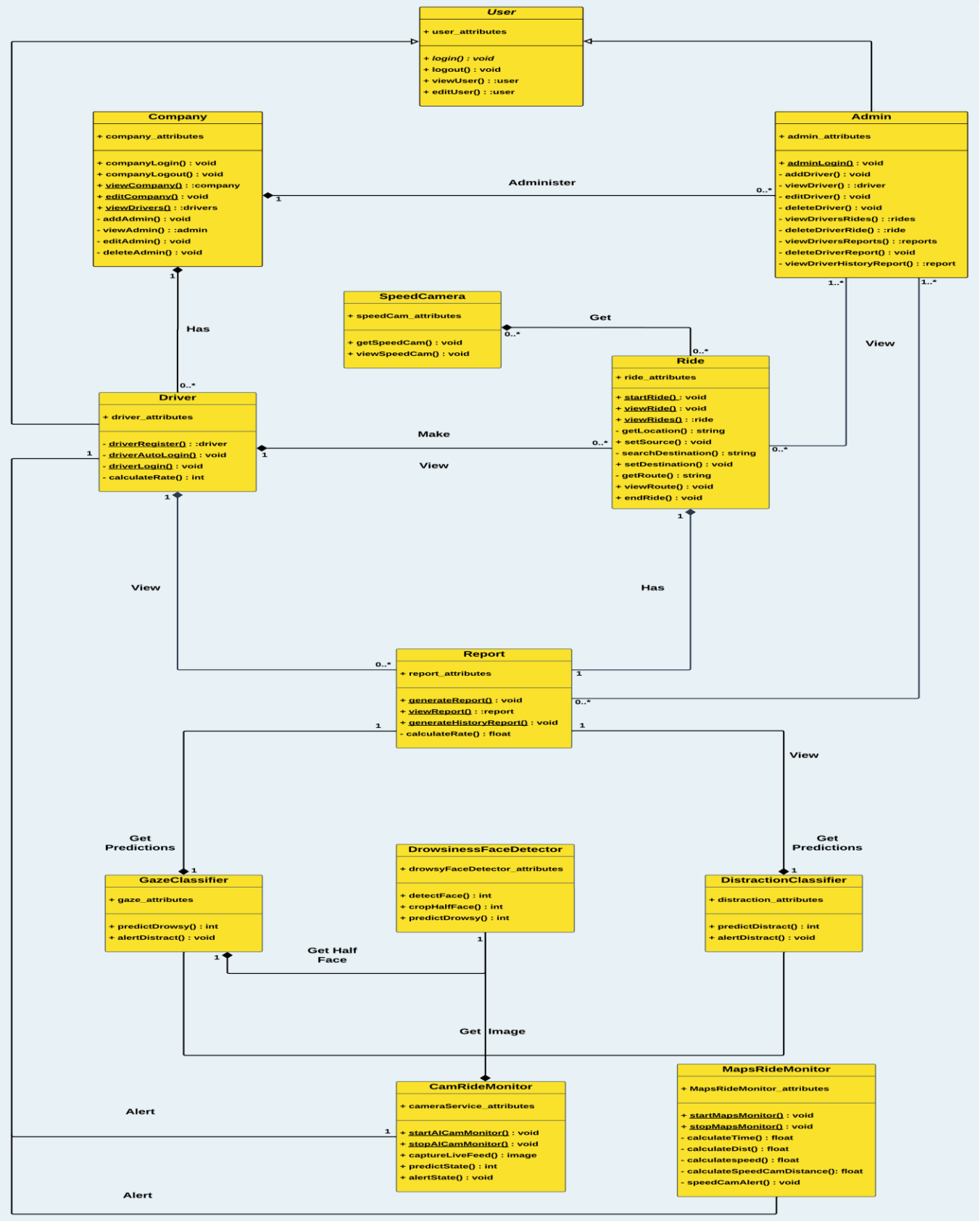
The class diagrams provide a visual representation of the classes used in the AI-Powered Driver Monitoring System for the three roles: driver, admin, and company representatives. These diagrams illustrate the structure of the system, including the classes, their attributes, methods, and relationships.

3.7.1 System Logical Class Diagram

The system logical class diagram serves as an initial representation of the entire system, showcasing the main classes and their interactions. It captures the core components and their relationships, providing an overview of the system's structure. By visualizing the classes and their connections, the system logical class diagram offers a high-level understanding of the system's organization.

These class diagrams facilitate system understanding, design, and implementation by visually representing the classes, their attributes, methods, and relationships. They promote effective communication among stakeholders, ensuring a shared understanding of the system's structure and behavior. Additionally, the class diagrams serve as valuable references during system maintenance and updates, aiding in the identification of areas for modification or extension.

By developing and analyzing these class diagrams, we establish a solid foundation for the AI-Powered Driver Monitoring System, ensuring its coherence, scalability, and maintainability. The diagrams contribute to a comprehensive understanding of the system's structure and behavior, supporting efficient development and seamless collaboration among stakeholders.

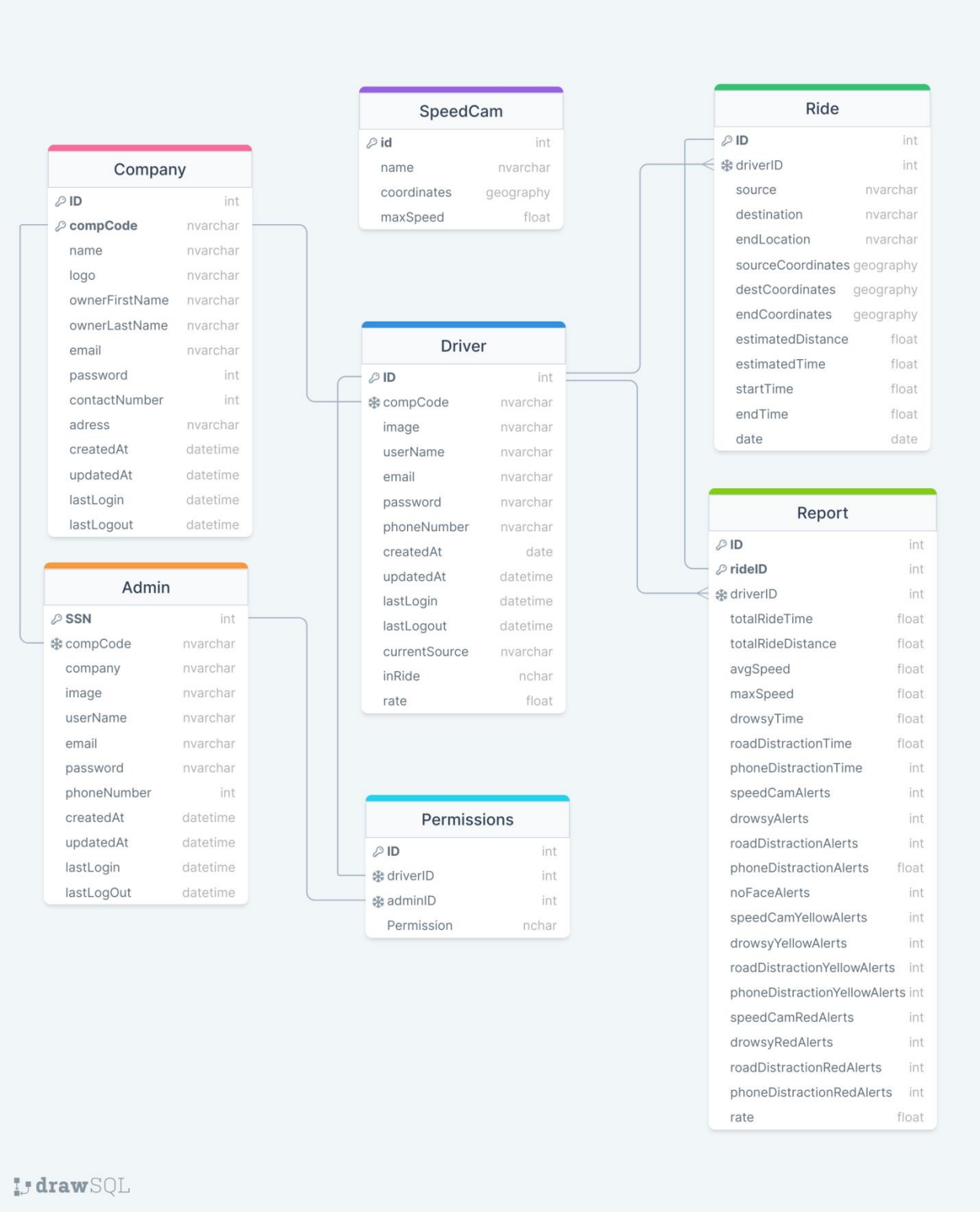

User
+ user_attributes
+ login() : void
+ logout() : void
+ viewUser() : :user
+ editUser() : :user
Company
+ company_attributes
+ companyLogin() : void
+ companyLogout() : void
+ viewCompany() : :company
+ editCompany() : void
+ viewDrivers() : :drivers
- addAdmin() : void
- viewAdmin() : :admin
- editAdmin() : void
- deleteAdmin() : void
Admin
+ admin_attributes
+ adminLogin() : void
- addDriver() : void
- viewDriver() : :driver
- editDriver() : void
- deleteDriver() : void
- viewDriversRides() : :rides
- deleteDriverRide() : :ride
- viewDriversReports() : :reports
- deleteDriverReport() : void
- viewDriverHistoryReport() : :report
Administer
Has
SpeedCamera
+ speedCam_attributes
+ getSpeedCam() : void
+ viewSpeedCam() : void
Get
View
Ride
+ ride_attributes
+ startRide() : void
+ viewRide() : void
+ viewRides() : :ride
- getLocation() : string
+ setSource() : void
- searchDestination() : string
+ setDestination() : void
- getRoute() : string
+ viewRoute() : void
+ endRide() : void
Driver
+ driver_attributes
- driverRegister() : :driver
- driverAutoLogin() : void
- driverLogin() : void
- calculateRate() : int
Make
View
View
Has
Report
+ report_attributes
+ generateReport() : void
+ viewReport() : :report
+ generateHistoryReport() : void
- calculateRate() : float
View
Get Predictions
Get Predictions
DrowsinessFaceDetector
+ drowsyFaceDetector_attributes
+ detectFace() : int
+ cropHalfFace() : int
+ predictDrowsy() : int
GazeClassifier
+ gaze_attributes
+ predictDrowsy() : int
+ alertDistract() : void
DistractionClassifier
+ distraction_attributes
+ predictDistract() : int
+ alertDistract() : void
Get Half Face
Get Image
CamRideMonitor
+ cameraService_attributes
+ startAICamMonitor() : void
+ stopAICamMonitor() : void
+ captureLiveFeed() : image
+ predictState() : int
+ alertState() : void
MapsRideMonitor
+ MapsRideMonitor_attributes
+ startMapsMonitor() : void
+ stopMapsMonitor() : void
- calculateTime() : float
- calculateDist() : float
- calculatespeed() : float
- calculateSpeedCamDistance(): float
- speedCamAlert() : void
Alert
Alert

3.8 Database Desing Diagrams

Initially, we designed our database to be SQL-based. However, after careful consideration, we opted to utilize Firebase Firestore, a NoSQL database. As a result, we transformed our SQL relational schema into a NoSQL document-based collection schema to better align with the capabilities and advantages of Firestore.

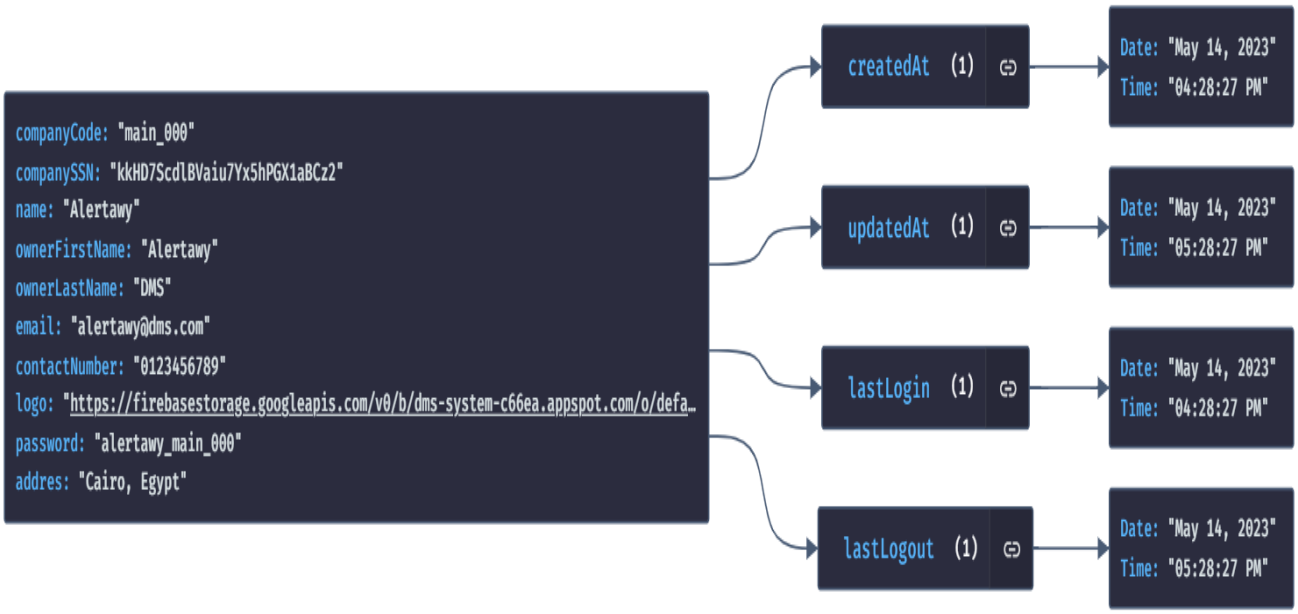
The database diagram showcases the structure and relationships of the document collections within Firestore. It illustrates how the data is organized and interconnected, facilitating efficient data retrieval and storage. By leveraging the flexibility and scalability of Firestore's NoSQL approach, we are able to efficiently manage and manipulate data, ensuring optimal performance for our AI-Powered Driver Monitoring System.

3.8.1 SQL

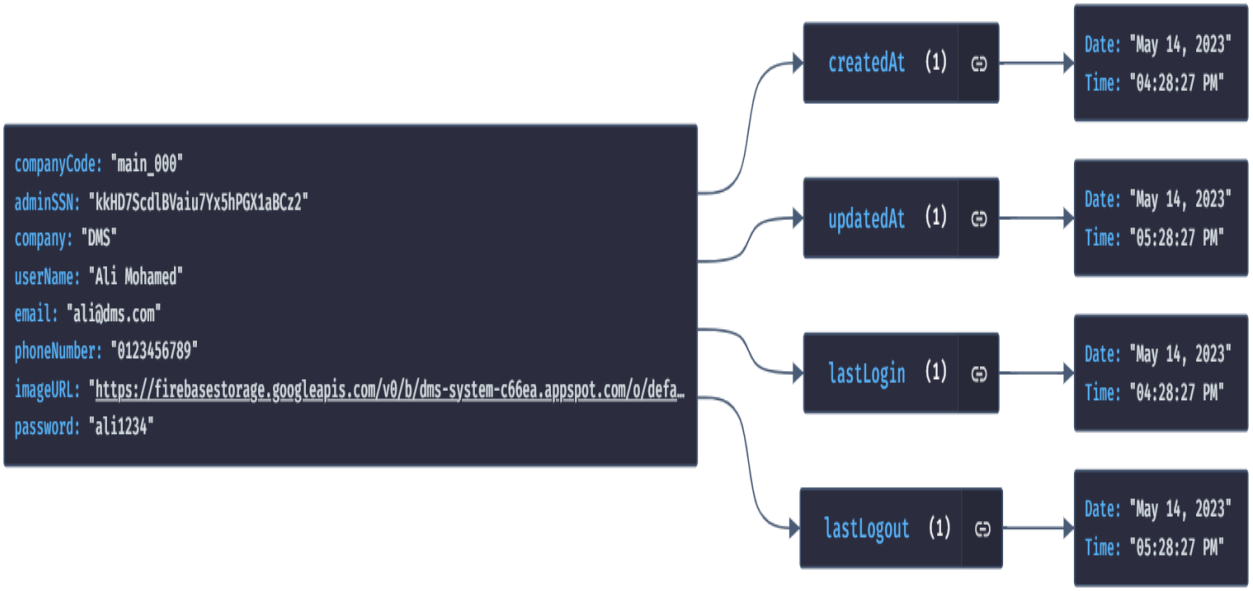

SpeedCam
id int
name nvarchar
coordinates geography
maxSpeed float
Ride
ID int
driverID int
source nvarchar
destination nvarchar
endLocation nvarchar
sourceCoordinates geography
destCoordinates geography
endCoordinates geography
estimatedDistance float
estimatedTime float
startTime float
endTime float
date date
Company
ID int
compCode nvarchar
name nvarchar
logo nvarchar
ownerFirstName nvarchar
ownerLastName nvarchar
email nvarchar
password int
contactNumber int
adress nvarchar
createdAt datetime
updatedAt datetime
lastLogin datetime
lastLogout datetime
Driver
ID int
compCode nvarchar
image nvarchar
userName nvarchar
email nvarchar
password nvarchar
phoneNumber nvarchar
createdAt date
updatedAt datetime
lastLogin datetime
lastLogout datetime
currentSource nvarchar
inRide nchar
rate float
Report
ID int
rideID int
driverID int
totalRideTime float
totalRideDistance float
avgSpeed float
maxSpeed float
drowsyTime float
roadDistractionTime float
phoneDistractionTime int
speedCamAlerts int
drowsyAlerts int
roadDistractionAlerts int
phoneDistractionAlerts float
noFaceAlerts int
speedCamYellowAlerts int
drowsyYellowAlerts int
roadDistractionYellowAlerts int
phoneDistractionYellowAlerts int
speedCamRedAlerts int
drowsyRedAlerts int
roadDistractionRedAlerts int
phoneDistractionRedAlerts int
rate float
Admin
SSN int
compCode nvarchar
company nvarchar
image nvarchar
userName nvarchar
email nvarchar
password nvarchar
phoneNumber int
createdAt datetime
updatedAt datetime
lastLogin datetime
lastLogOut datetime
Permissions
ID int
driverID int
adminID int
Permission nchar
drawSQL

3.8.2) NOSQL

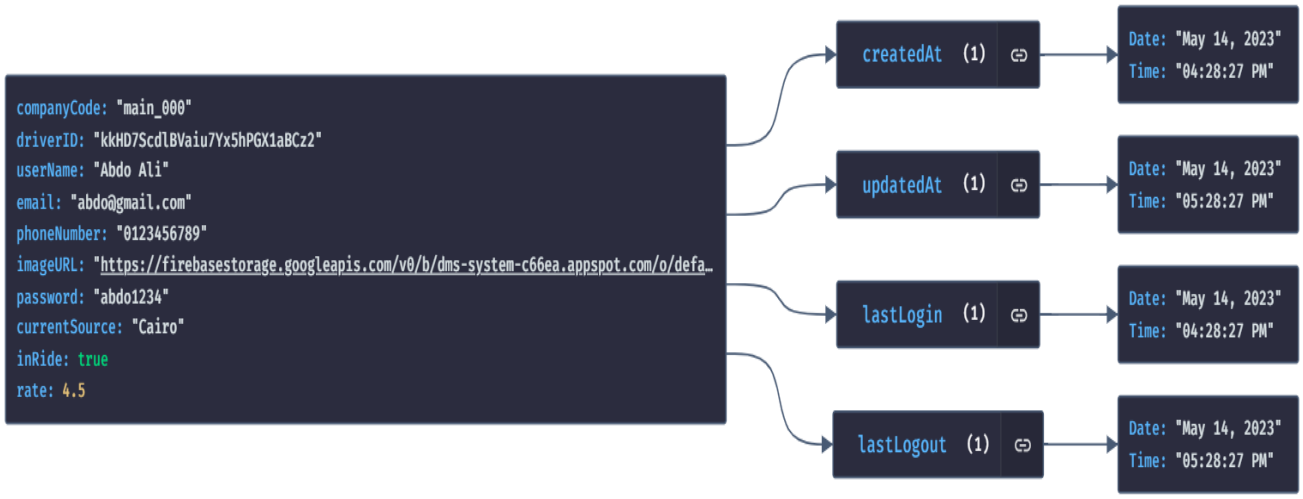
3.8.2.1 Company Collection



3.8.2.2 Admin Collection

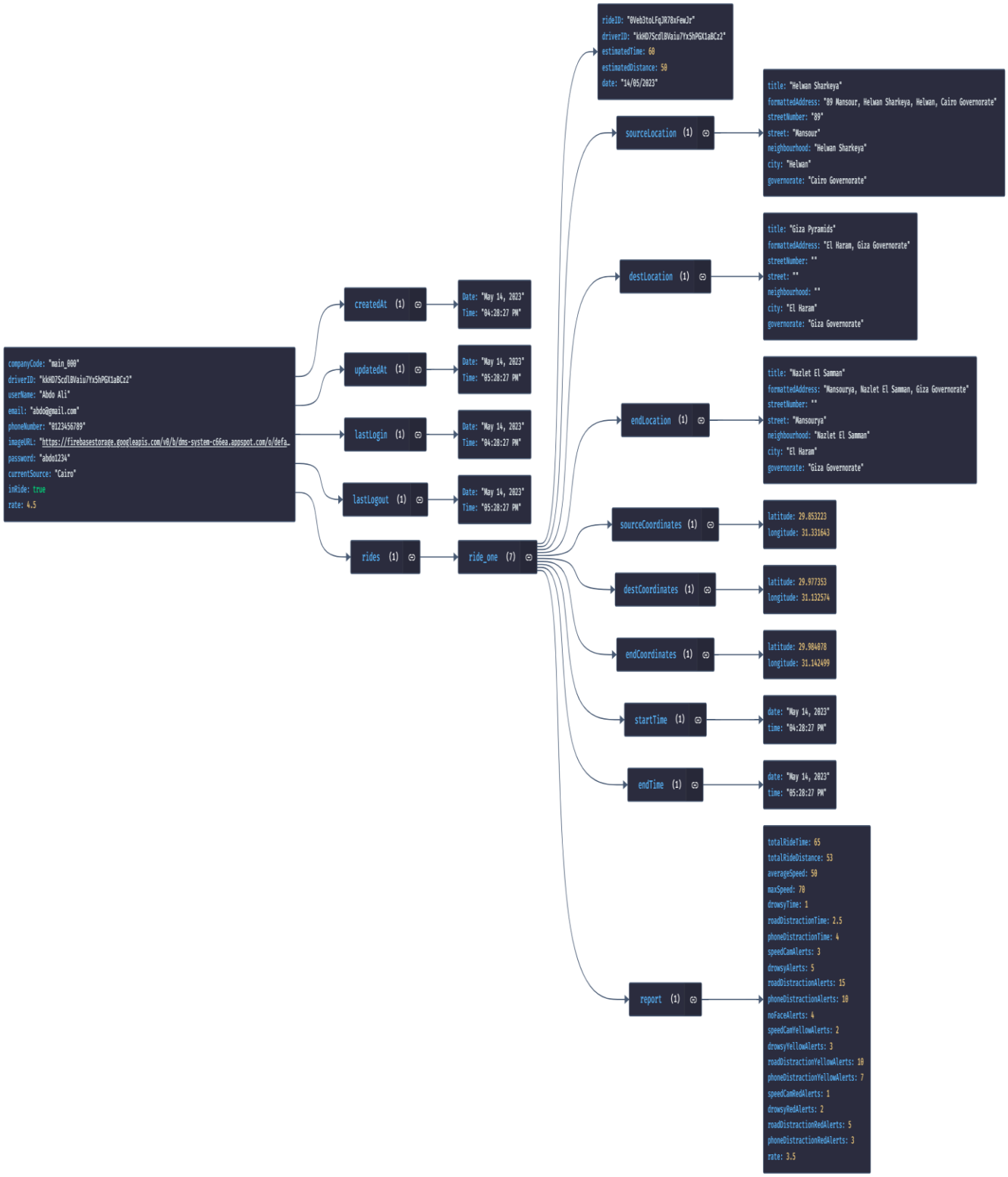


3.8.2.3) Driver Collection

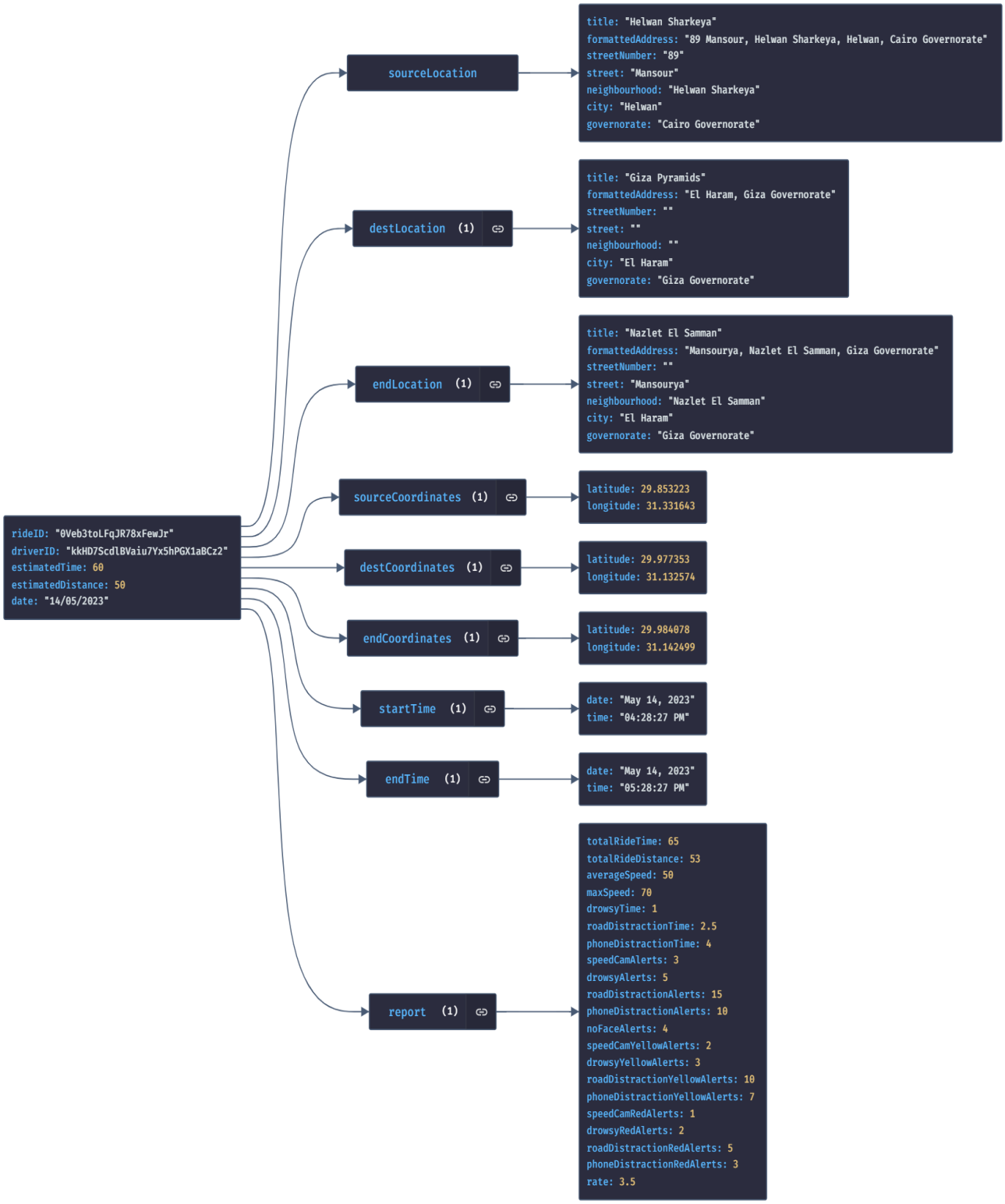


3.8.2.4) Speed Camera Collection

3.8.2.5) Driver rides



3.8.2.6) Rides Collection



The diagrams represent the different document collections, their fields, and the relationships between them. It highlights how various entities, such as drivers, trips, alerts, and system settings, are stored and linked within the database. The NoSQL document-based approach allows for dynamic and nested data structures, enabling us to store and retrieve data in a more flexible and scalable manner.

By adopting Firestore as our database solution, we can effectively handle the storage and retrieval of large volumes of data generated by our system. The NoSQL nature of Firestore offers advantages such as horizontal scalability, real-time synchronization, and ease of integration with our mobile and web applications.

Through careful design and optimization of our database schema in Firestore, we ensure efficient data management and support the seamless functioning of our AI-Powered Driver Monitoring System.

Chapter 4: Implementation & Results

4.1 Deep Learning Models

In the implementation of our deep learning models, we initially developed two custom-trained models: one for gaze estimation to determine the driver's eye direction and alert them if they are not focused on the road, and the other for detecting driver distractions related to phone use.

In the following section, we will delve into the training details, experiments, and results of our models, demonstrating the advancements and effectiveness achieved through our approach.

4.1.1 Distraction Detection Custom-Trained Model

At First, we started with Distraction Detection Model, we created two models: a Vanilla CNN Model and a fine-tuned MobileNetV2 Model.

4.1.1.1 Models Building

The architecture of the Vanilla CNN Model follows a VGG-like architecture, with each CNN block consisting of two convolutional layers followed by a max pooling layer.

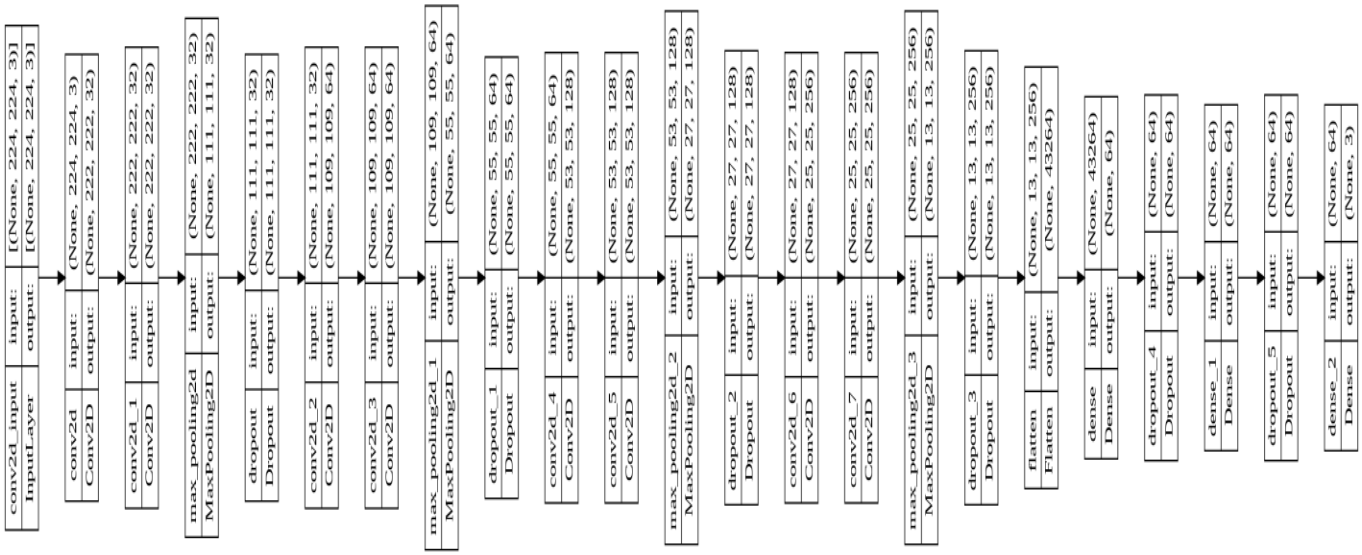


Figure 1: Vanilla CNN Model

Next, we modified the MobileNetV2 Model by removing its last block to reduce complexity and the risk of overfitting. We then fine-tuned the model's architecture by adding fully connected dense layers with 256 nodes and applying L2 regularization with a penalty of 0.05 to the dense layer.

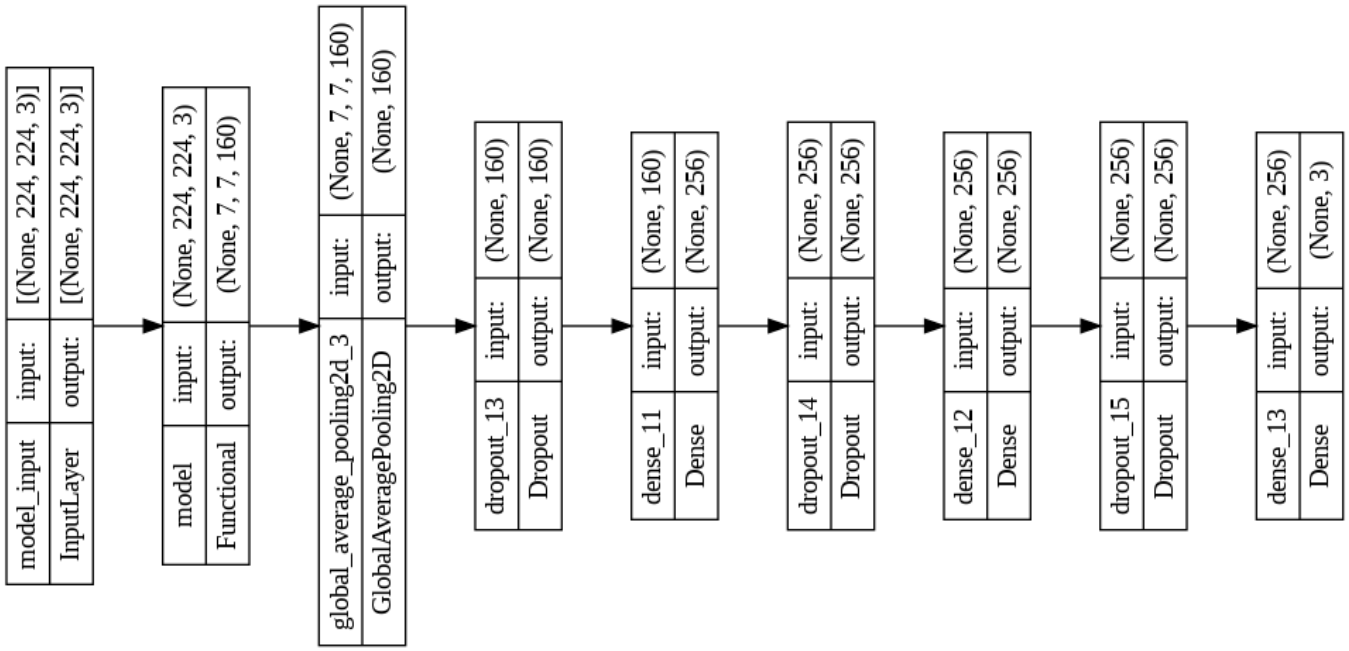


Figure 2: Fine-Tuned MobileNet Model

Although both models showed promising results, the fine-tuned MobileNetV2 Model exhibited higher accuracy and faster inference times. Therefore, we focused on further developing and training this model for both gaze estimation and distraction detection.

4.1.1.2 Distraction Detection Dataset

In Distraction Detection we utilized and combined three datasets:

- State Farm Distraction Kaggle Competition Dataset
- AUC Distracted Driver Dataset
- Rabat University HowDriver3D Distracted Driver Dataset

These datasets shared the same 10 distraction classes, making the combination process straightforward.

Figure 3 Combined Dataset Distraction Classes

The State Farm Dataset consisted of approximately 11,745 images with 26 driver images, while the AUC Distracted Driver Dataset contained around 8,404 images from 31 drivers. The HowDrive3D Dataset contributed approximately 38,000 images from 9 drivers. In total, we obtained about 75 thousand images from 66 drivers.

These datasets provided a diverse range of drivers, including individuals of different races, nationalities, ages, and lighting conditions. Some drivers were captured in the morning, while others were photographed during the middle of the day.

Figure 4: State Farm Distraction Dataset Samples

Figure 5: AUC Distracted Driver Dataset Samples

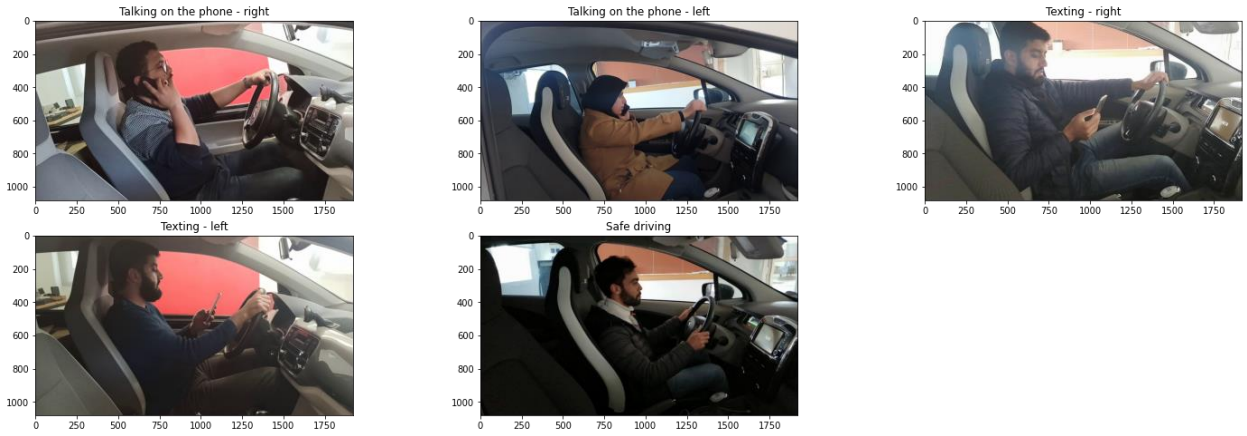


Figure 6 HowDrive3D Dataset Samples

However, since our focus was on detecting phone use, we only selected specific classes that aligned with our desired camera angle. We excluded classes that did not suit our intended viewpoint, resulting in a dataset that primarily consisted of images related to safe driving, texting, and talking on the phone. We combined the left and right classes of texting and talking, as the specific phone placement was less important to us compared to detecting phone use itself. After merging and filtering the datasets, we were left with a final dataset consisting of approximately 40,000 images.

Figure 7: Classes of Interest

Figure 8: Classes After Combination

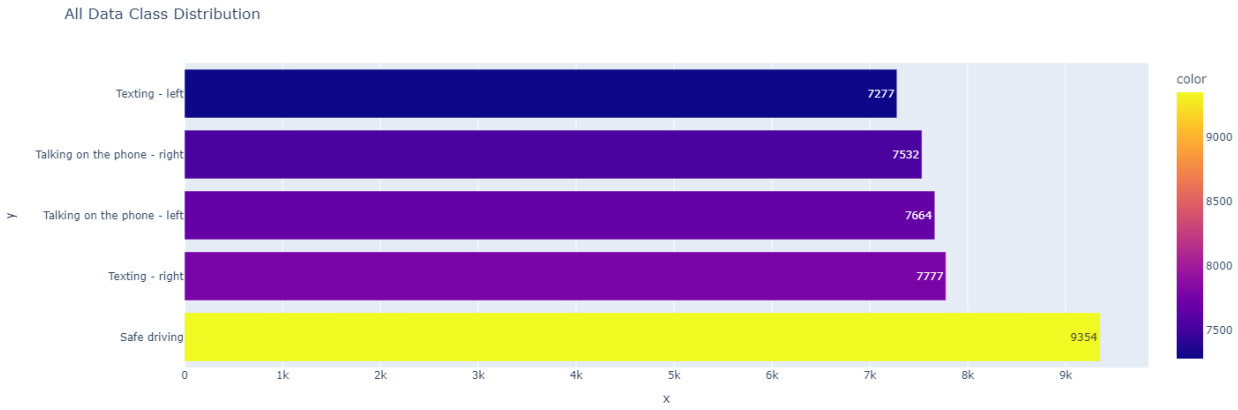


Figure 9: Combined Data Classes Distribution

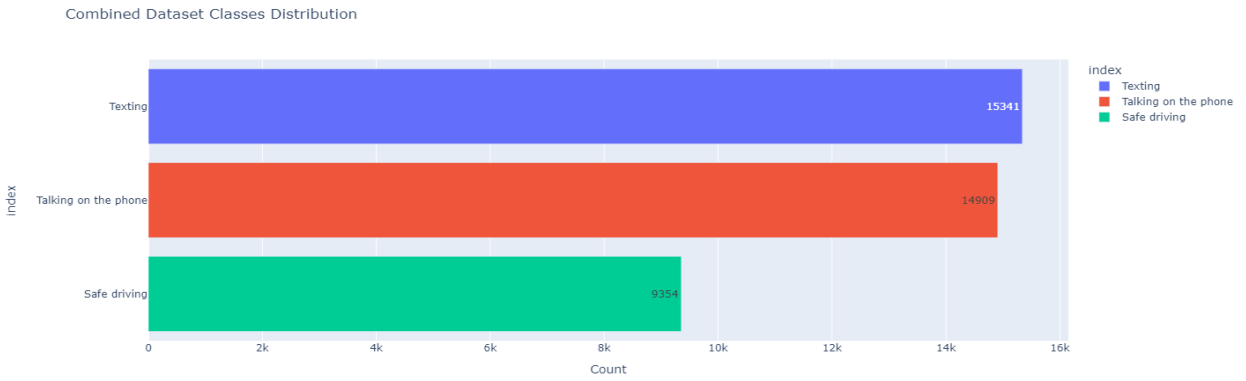


Figure 10: Final Combined Distraction Dataset

One of the significant challenges we encountered was the mismatch between the camera angles of the datasets and our desired camera angle. The datasets were primarily captured from the side of the driver near the passenger seat mirror. To address the challenge of camera angle discrepancy between the datasets and our desired camera placement near the driver, we employed comprehensive data augmentation techniques. This augmentation process aimed to align the dataset's angles as closely as possible with our desired camera angle, enhancing the model's ability to classify the driver's behavior accurately.

Figure 11: Dataset Camera Angle vs Desired Camera Angle

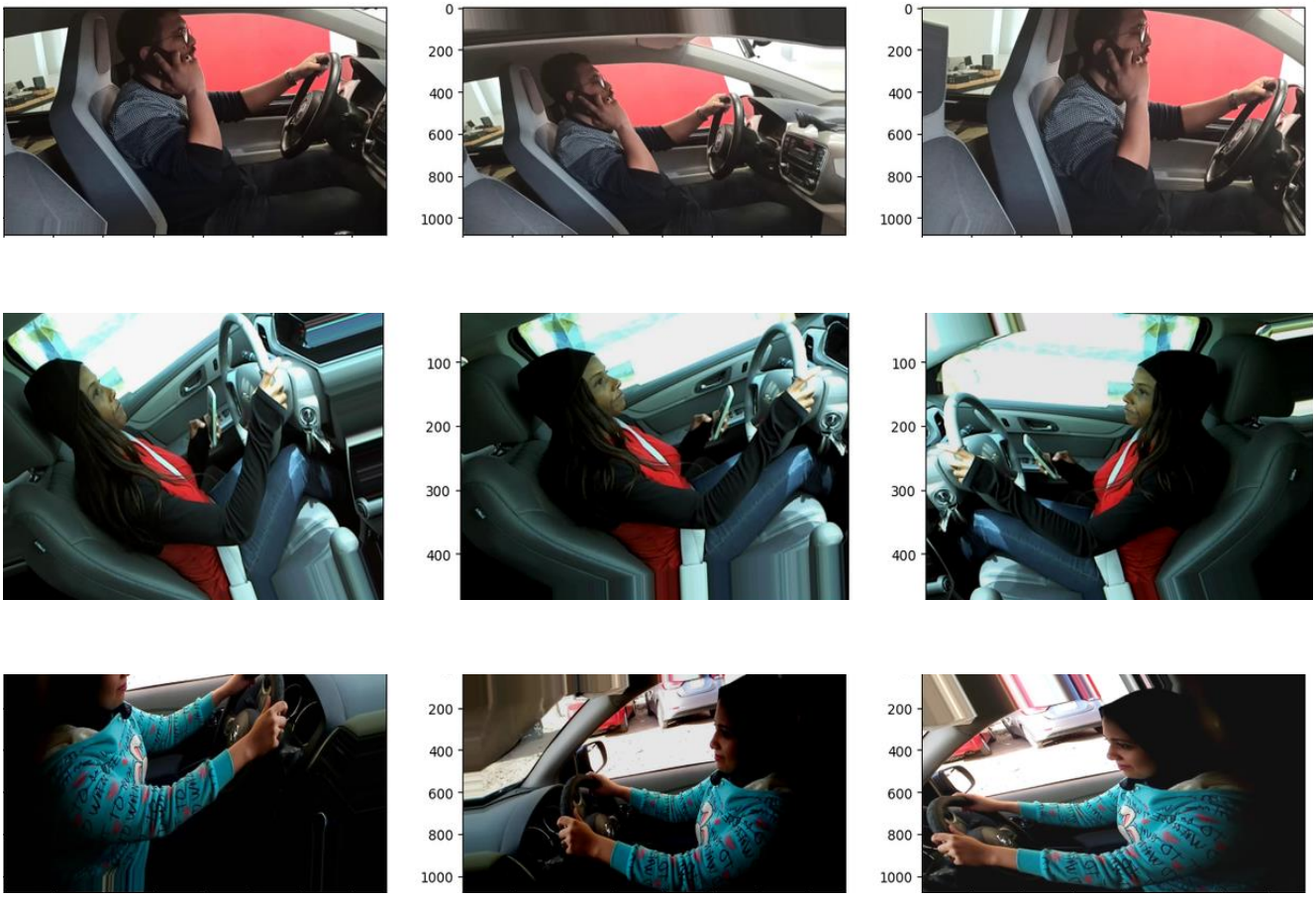


Figure 12: Samples of Data Augmentation

4.1.1.3 Models Training & Evaluation

To begin with, we divided our combined dataset into training, validation, and testing sets. Instead of random splits, we based the division on different drivers to ensure the models wouldn't be overfit to specific driver features or light conditions. We also made sure to maintain the same distribution of classes across all sets. This approach provided a diverse and representative dataset for training our models.

Following that, we performed data preprocessing by resizing all the images to a uniform size of 224x224 and normalizing the pixel values to fall within the range of [0, 1]. This step was crucial for improved training performance. Additionally, we implemented the data augmentation techniques mentioned eariler as part of a comprehensive data preprocessing pipeline to further enhance the training process.

Training our models on the combined dataset was a computationally intensive task that required significant time and resources. We leveraged the GPU capabilities of Google Colab for training, but even then, a single epoch took approximately 20 minutes to complete. Despite the limitations, we managed to train our models for 30 epochs with a batch size of 256 on 224x224 RGB images.

The initial results with the vanilla CNN model showed promising accuracy, achieving around 94% on the testing set and 95% on the validation set. However, we aimed for even higher accuracy and faster inference times, leading us to focus on fine-tuning the MobileNetV2 pre-trained model.

The fine-tuned MobileNetV2 model followed the same training process as the vanilla CNN model, using similar fitting parameters. However, it demonstrated significantly improved performance. The fine-tuned model achieved an accuracy of 98% on the testing set and 97% on the validation set.

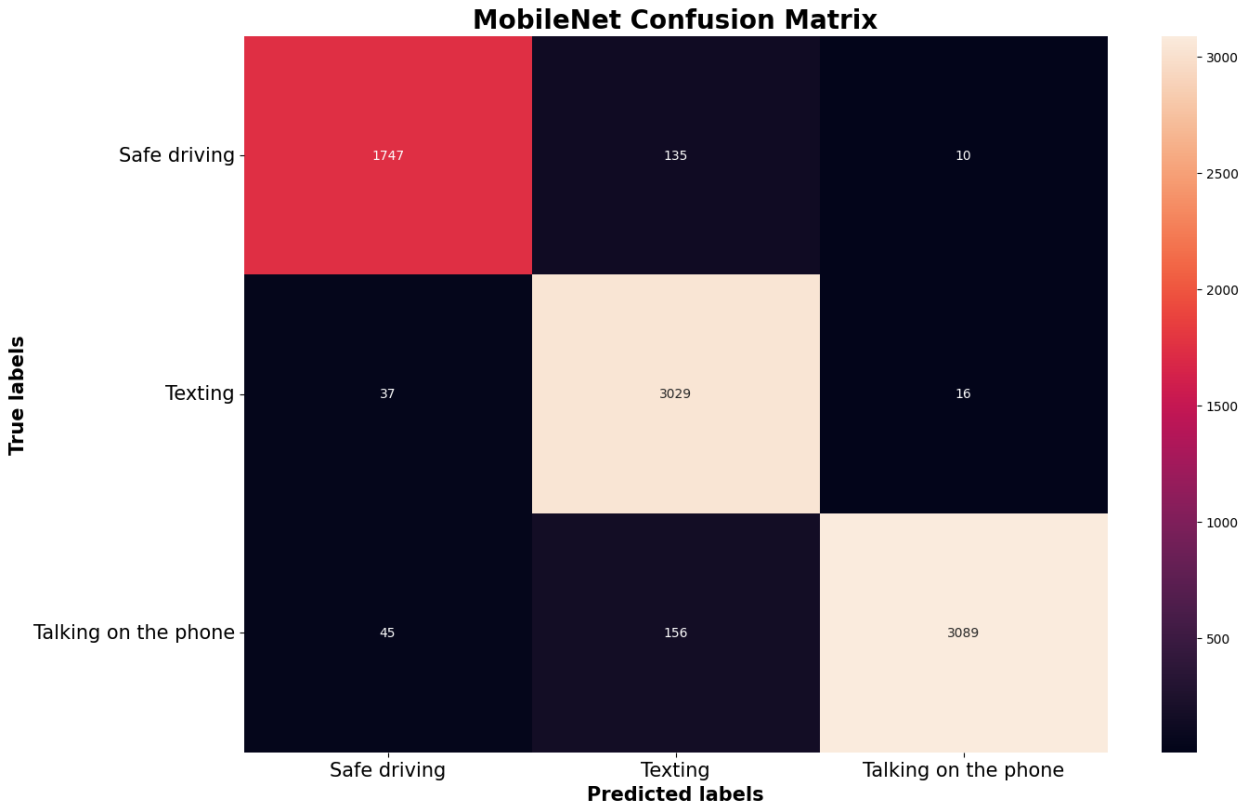


Figure 13: Fine-Tuned MobileNet Model Confusion Matrix

The confusion matrix showed that the model occasionally struggled with classifying the texting category, often confusing it with safe driving and talking on the phone classes. Nevertheless, the overall F1 score remained high at 97%, indicating that the model was not heavily biased toward specific classes.

4.1.1.4 Models Results & Experiments

To gain further understanding of our Distraction Detection custom-trained models, we evaluated their performance and conducted experiments to explore their inner workings. We first examined model predictions on random images from the testing set, observing that the models generally displayed high confidence in their predictions.

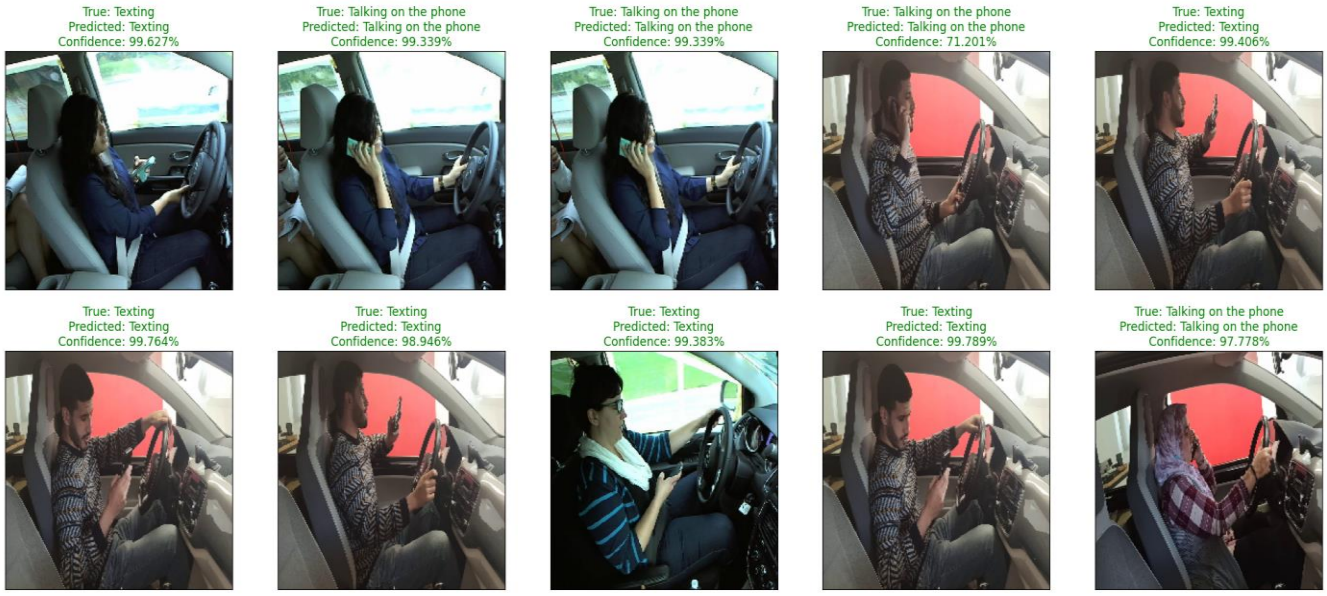


Figure 14: Model Predictions Confidence

However, to gain deeper insights into the models' inner workings, we delved into their black-box nature. By interpreting model errors and identifying confusing classes, we aimed to understand the decision-making process better. For this purpose, we employed techniques such as GRAD-CAM and Saliency Map to explain predictions, shedding light on the models' inference mechanisms and providing valuable insights.

Analyzing the figures, we made several observations. For the Safe Driving class, the model focused on factors such as the driver's arm placement on the wheel and whether they were looking forward. In the

texting class, the model emphasized the presence of a phone in the image and whether the driver's hand was on the wheel. Similarly, in the Talking on the Phone class, the model looked for patterns of an arm carrying a phone, specifically detecting if the phone was in the left hand or the right hand, while also assessing the position of the driver's hand on the wheel.

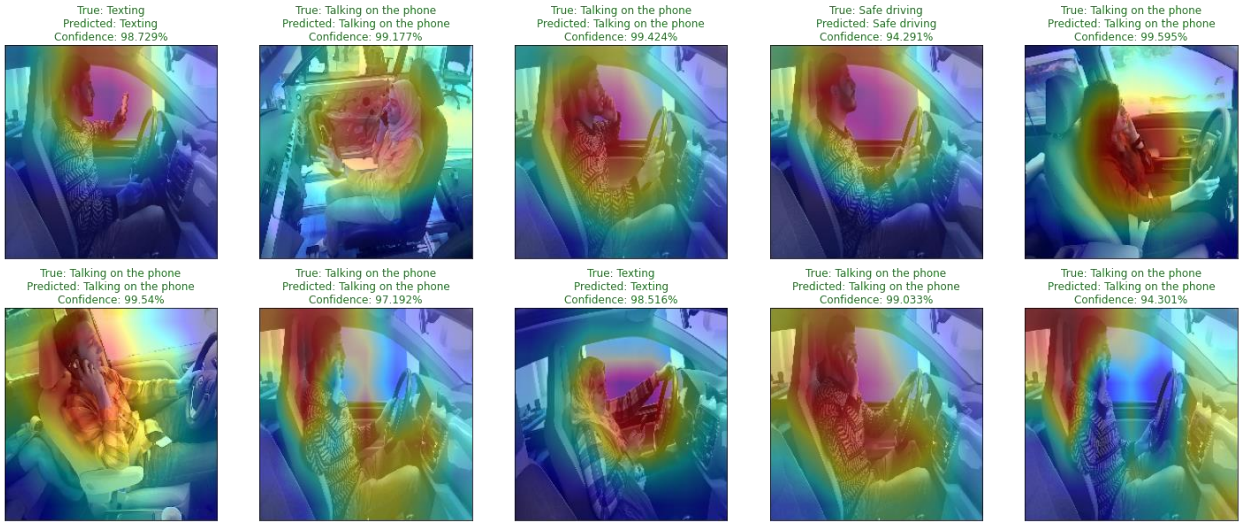


Figure 15: Grad-Cam Heatmap on Model Predictions

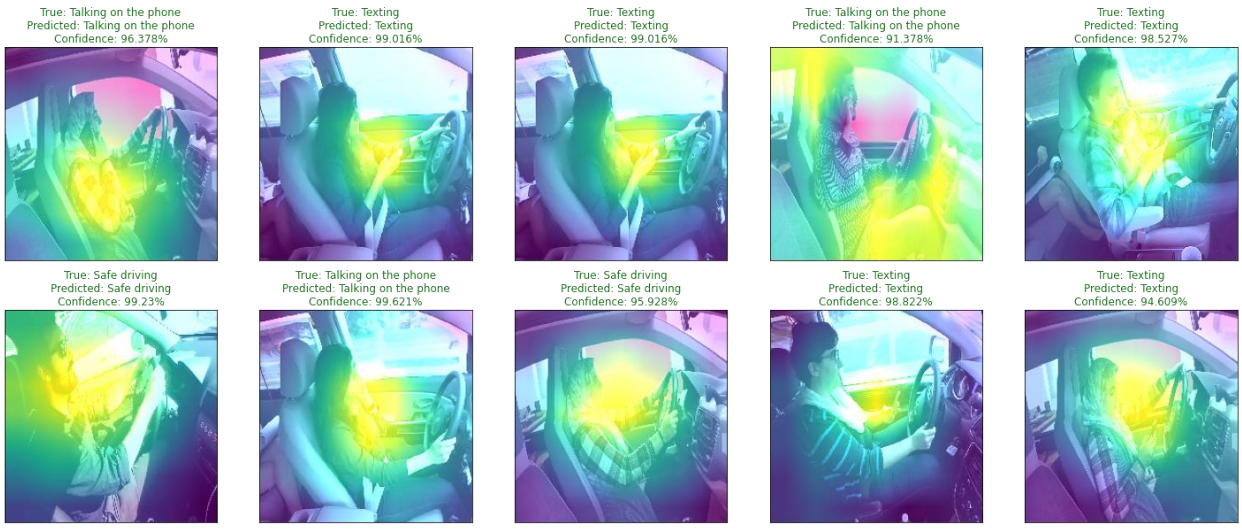


Figure 16: Guided Grad-Cam Interpretation

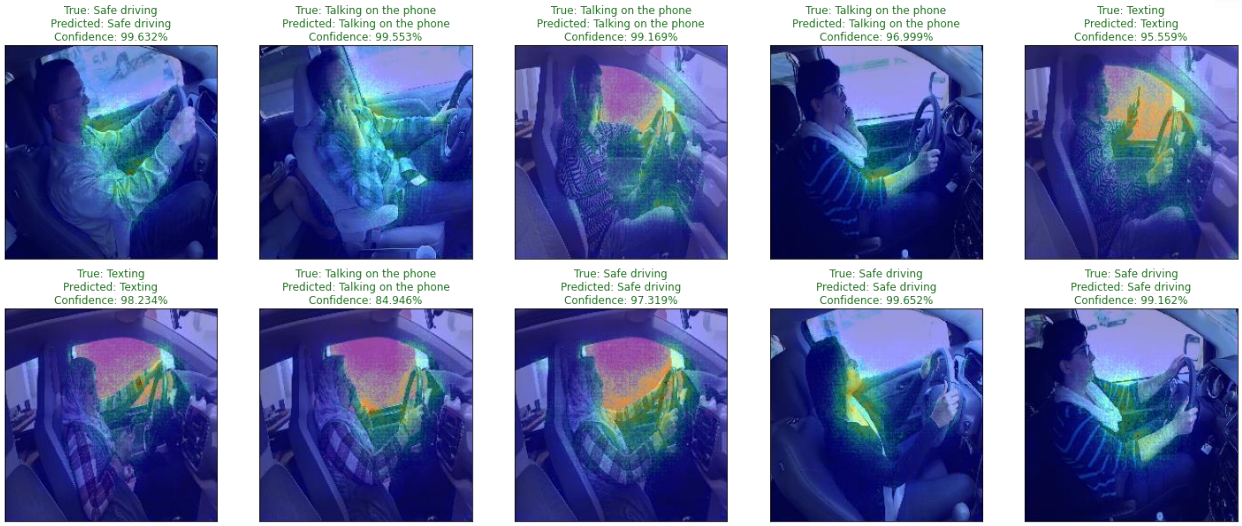


Figure 17: Sailency Map on Model Prediction

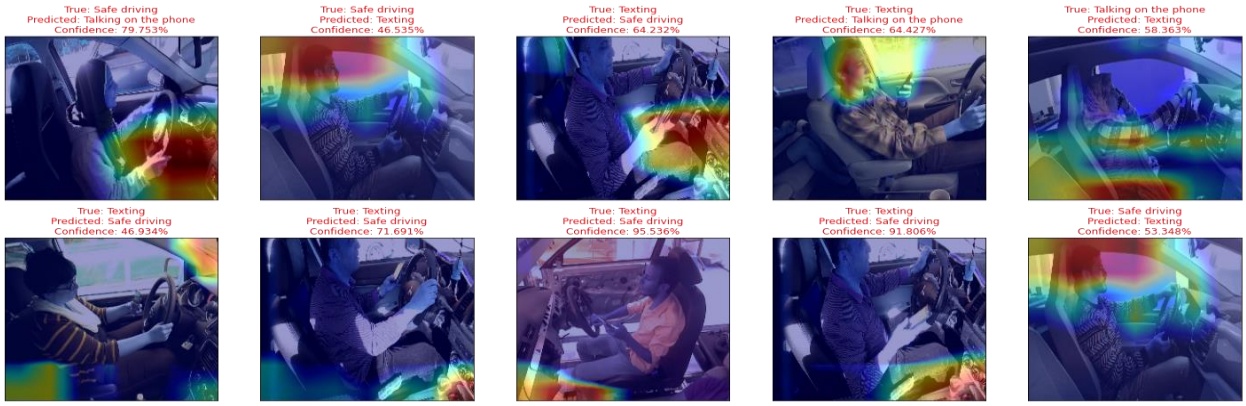


Figure 18: Model Wrong Prediction Interpretation

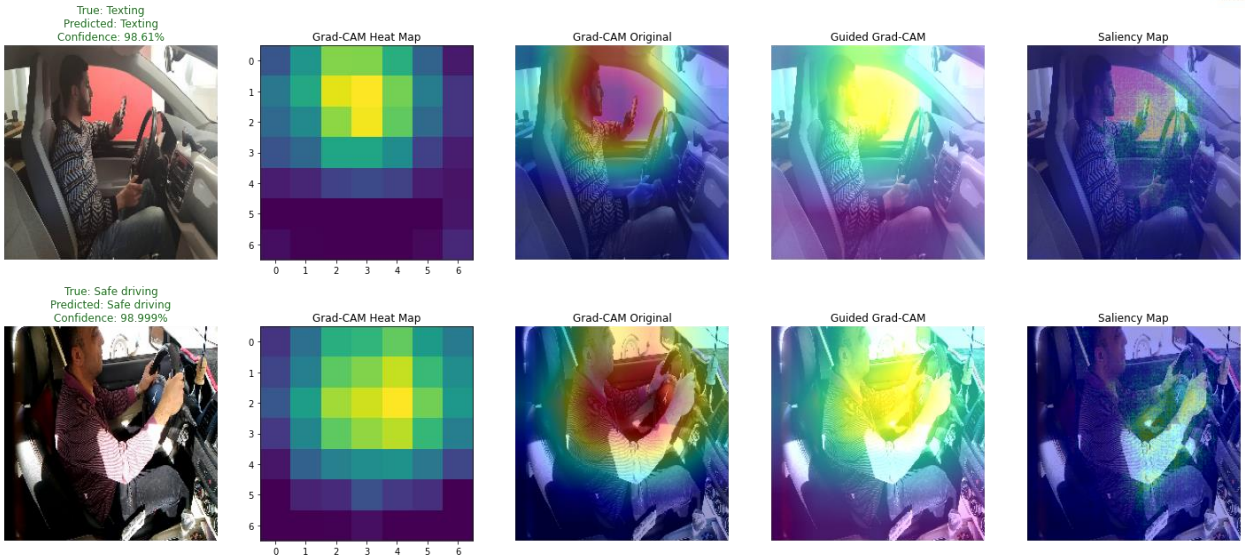


Figure 19: Comparing Interpretation Techniques

These findings allowed us to better understand how the models made predictions, highlighting the important visual cues they relied on for each distraction class. By interpreting and analyzing the models' decision-making processes, we gained valuable insights into their performance and potential areas for improvement.

Finally, we assembled a testing-deployment set comprising approximately 100 images captured from the desired camera angle for our deployment environment. This set aimed to assess the model's performance in real-world scenarios. To our delight, the results showed promising predictions, even though the model had been trained on completely different camera angles. We achieved an accuracy of nearly 80% in correctly predicting the desired outcomes.

Additionally, we explored the use of object detection models to identify and isolate the driver in the image, thereby focusing solely on the driver's behavior rather than other objects or passengers in the scene. For this purpose, we employed the MobileNet Localizer Model provided by Google, which is readily available in the Google ML Kit for mobile

devices. This allowed us to integrate the model seamlessly into our mobile deployment.

Initially, the MobileNet Localizer successfully detected the driver, leading to an increase in model prediction accuracy. However, in many instances, it failed to identify the driver as the prominent object in the image. Moreover, in real-time environments, the performance of the MobileNet Localizer decreased further. There were also cases where useful parts of the image, such as the driver's hand holding a mobile phone, were erroneously cropped out. Due to these limitations and drawbacks, we decided not to pursue this approach further.

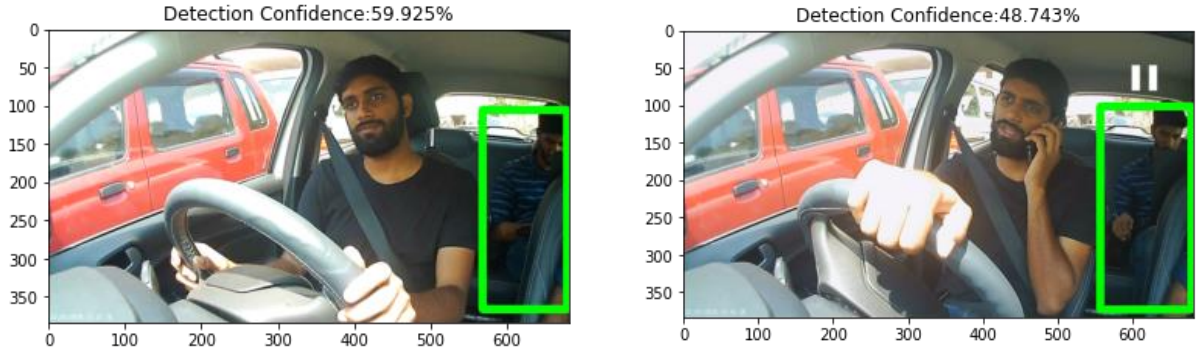


Figure 20: Mobile Localizer Wrong Prominent Object(Driver) Detection

Figure 21: MobileNet Localizer Successful Detection for Driver

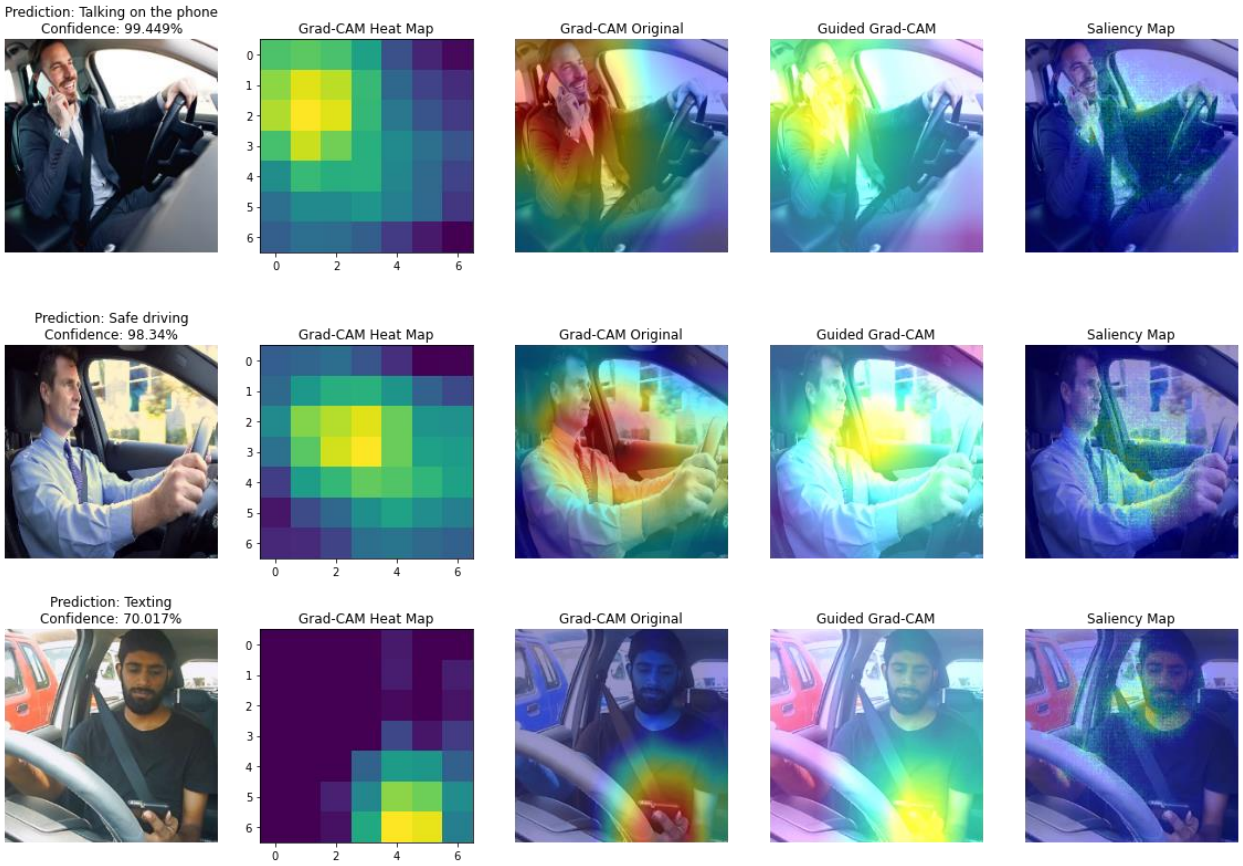


Figure 22: Testing-Deployment Set Prediction Samples

Through extensive testing in real-world scenarios, we rigorously assess and validat the performance of our model in the deployment environment. This comprehensive testing demonstrates our unwavering commitment to ensuring accuracy, efficiency, and real-time driver monitoring.

4.1.2 Gaze Estimation Custom-Trained Model

4.1.2.1 Models Building

We utilized the same MobileNet model architecture used in distraction detection but with a few modifications, specifically regarding model regularization in the Fine-Tuned MobileNet Model.

For the Gaze Estimation model, we made a slight adjustment to the L2 regularization penalty, reducing it to 0.001. This change was implemented to fine-tune the model's performance and optimize its ability to estimate the driver's gaze direction accurately.

By applying L2 regularization, we aimed to prevent overfitting and improve the model's generalization capabilities. This regularization technique helps to reduce the model's reliance on specific training examples and ensures that it can make reliable predictions on unseen data. The adjustment in the L2 regularization penalty was made based on experimentation and fine-tuning to strike a balance between model complexity and generalization ability. The goal was to create a gaze estimation model that performs effectively and accurately in real-world scenarios.

4.1.2.2 Gaze Estimation Dataset

For Gaze Estimation, we employed the LISA V1 Dataset, comprising a total of 30,000 images captured during 11 different long drives with 10 drivers. The dataset encompasses 8 classes known as "driver gaze zones." To ensure diversity, the drives were recorded on various days and at different times, resulting in a wide range of weather and lighting conditions. Additionally, the drivers exhibited diverse appearances, some wearing glasses and others not.

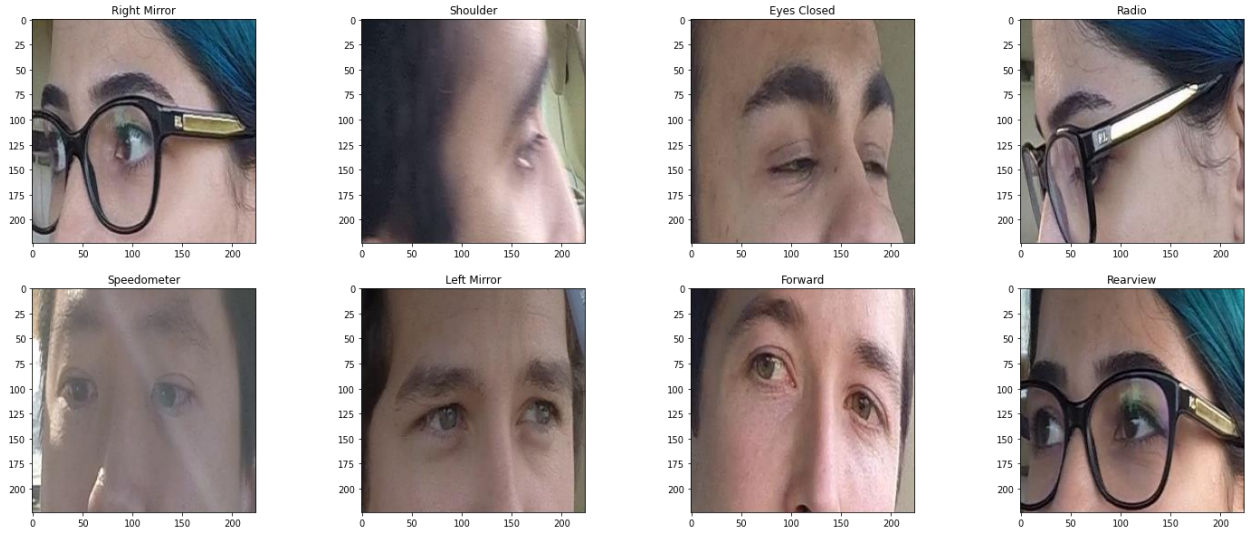


Figure 23: Gaze Zones Classes

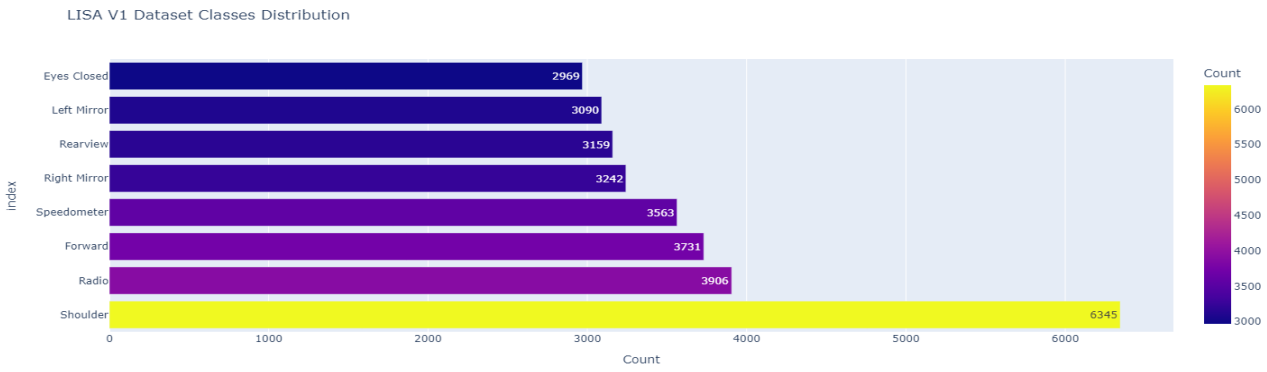


Figure 24: LISA V1 Classes Distribution

To focus specifically on the driver's eye direction, we combined certain classes and excluded others that were not relevant to our gaze estimation

task. For example, we were not interested in whether the driver was looking to the right or at the rearview or side mirrors. Our main focus was on determining whether the driver was looking forward or not.

It's worth noting the class imbalance within the dataset, particularly the shoulder class having double the number of images compared to other classes. To mitigate this imbalance, we decided to delete half of the images from the shoulder class, bringing it closer to the number of images in other classes.

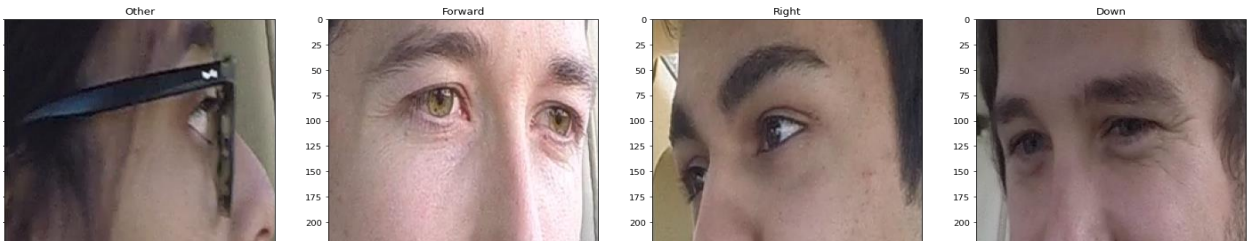


Figure 25;Combined Gaze Zones Classes

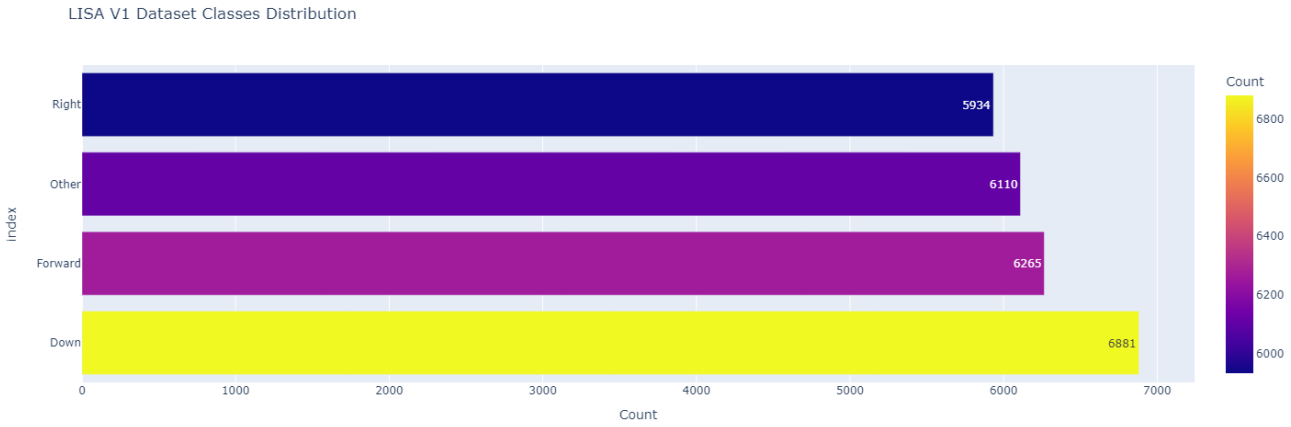


Figure 26: Combined Classes Distribution

To further enrich the LISA Dataset, we applied data augmentation techniques. While we refrained from vertically flipping the images to preserve eye gaze direction, we extensively augmented the dataset by adjusting image brightness, applying zoom, and performing shifts.

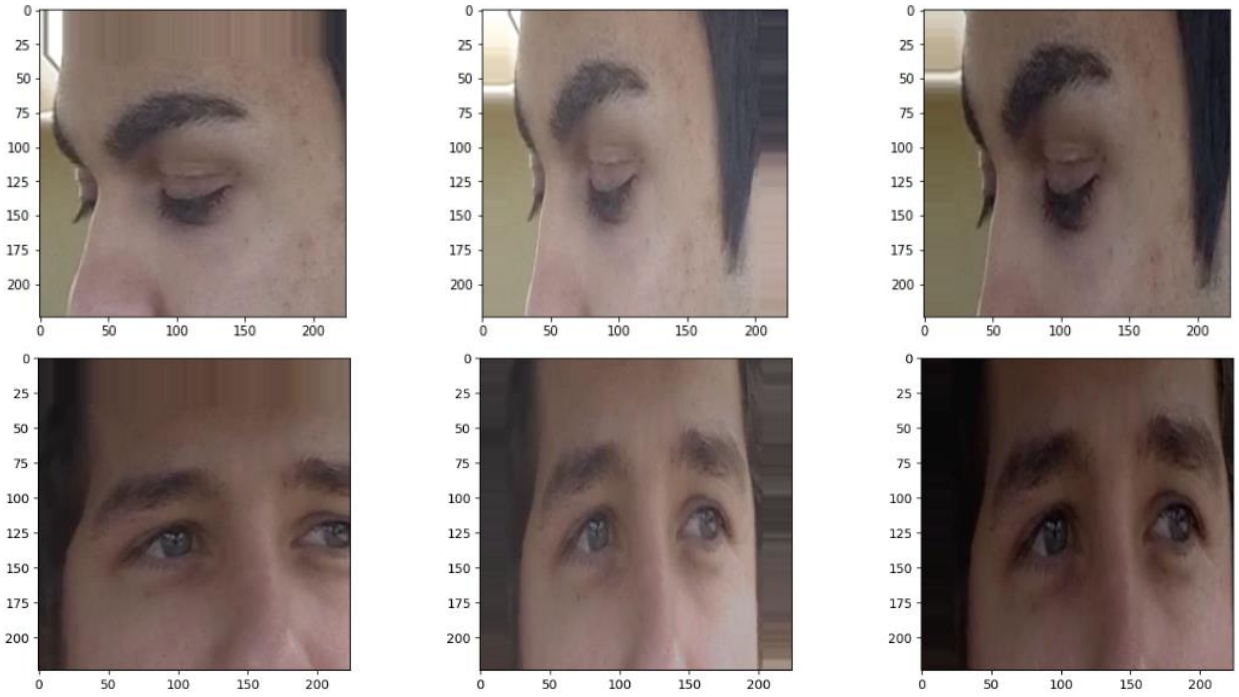


Figure 27: Data Augmentation Samples

By augmenting the dataset, we ensured that our model could better generalize to various real-world scenarios, encompassing different lighting conditions, driver appearances, and driving contexts.

4.1.2.3 Models Training & Evaluation

In line with the training approach for the Distraction Detection model, we employed a similar strategy for the Gaze Estimation model. We divided the LISA dataset into training, validation, and testing sets. Unlike its original split, which only included training and testing sets, we also incorporated a validation set. Furthermore, we based the division on different drivers to prevent the models from overfitting to specific driver features or lighting conditions. Additionally, we ensured that the distribution of classes remained consistent across all sets. This methodology resulted in a diverse and representative dataset for training our models.

Training the Fine-Tuned MobileNet Pre-Trained Model on the LISA dataset proved to be a comparatively easier task than working with the Combined Distraction Dataset. We took advantage of the GPU capabilities offered by Google Colab to expedite the training process. Each epoch took approximately 3 minutes to complete, and we successfully trained the models for 50 epochs using a batch size of 256 on 224x224 RGB images.

The fine-tuned MobileNetV2 model yielded impressive results, achieving an accuracy of 98% on the testing set and 97% on the validation set.

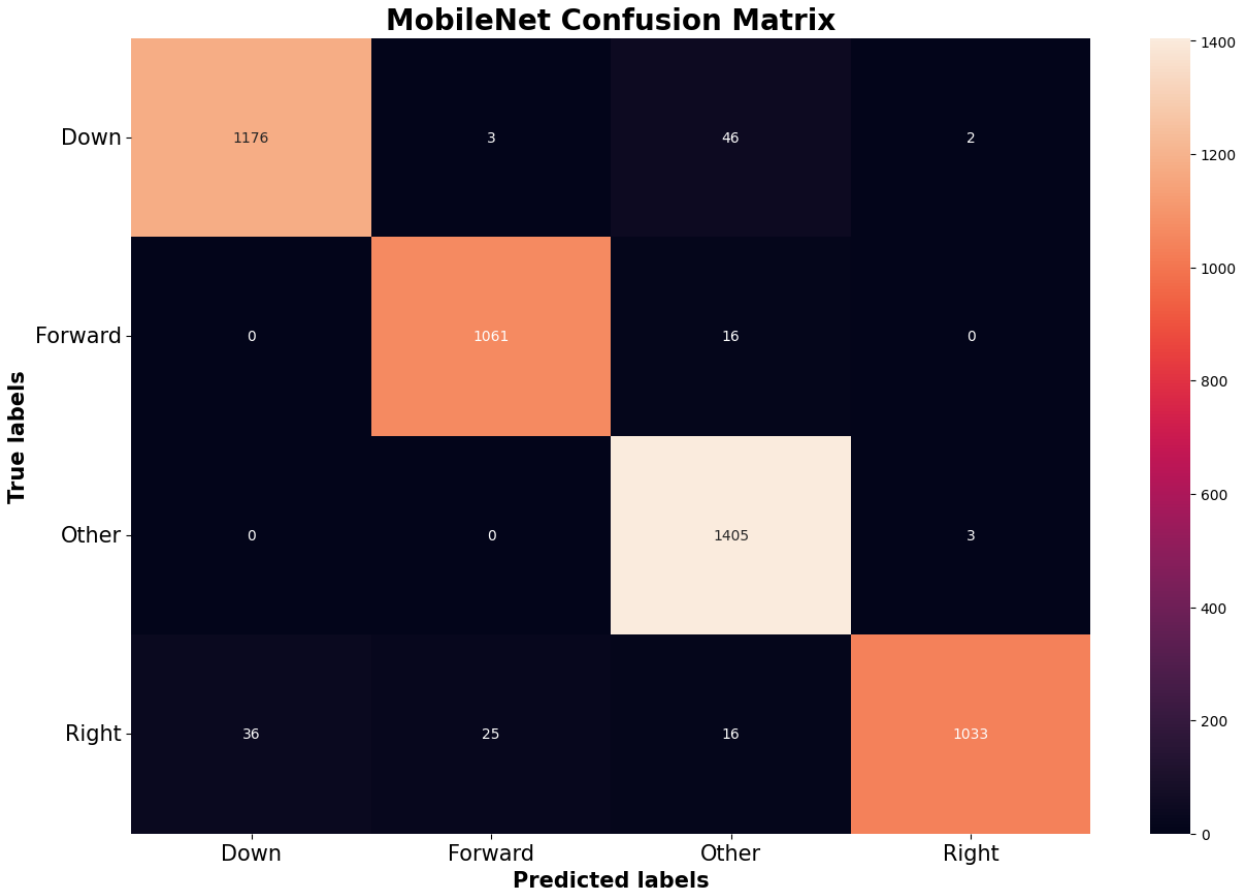


Figure 28: Fine-Tuned MobileNet Model Confusion Matrix

Upon examining the confusion matrix, we noticed that the model didn’t occasionally struggled with specific classes, perhaps the "down" and "other" classes. However, the overall F1 score remained high at 98%, indicating that the model was not heavily biased toward any particular class. These high accuracy and F1 scores demonstrated the model's strong performance and generalization capability for gaze estimation.

4.1.2.4 Models Results & Experiments

Building upon our approach in the Distraction Detection section, we conducted a thorough evaluation of our Gaze Estimation custom-trained model and performed experiments to gain a deeper understanding of its inner workings. Initially, we examined the model's predictions on random images from the testing set and observed that the models consistently displayed high confidence in their predictions.

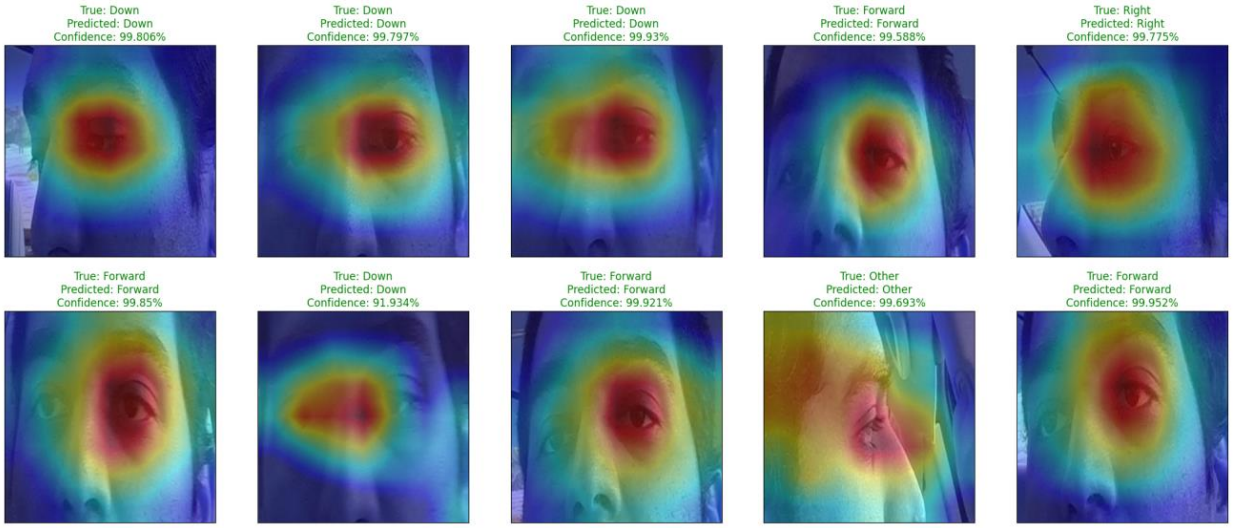


Figure 29: Model Predictions Confidence

To delve further into the black-box nature of the models and uncover their decision-making processes, we employed various techniques. By interpreting model errors and identifying confusing classes, we aimed to unravel the factors influencing the models' predictions. GRAD-CAM and Saliency Map techniques were utilized to provide visual explanations, shedding light on the models' inference mechanisms and offering valuable insights into their performance.

Our analysis revealed several noteworthy observations. The model demonstrated intelligence in focusing on a single eye for determining gaze direction, as shown in the provided figures. This indicated that the

model based its predictions primarily on the direction of one eye, rather than relying on both eyes.

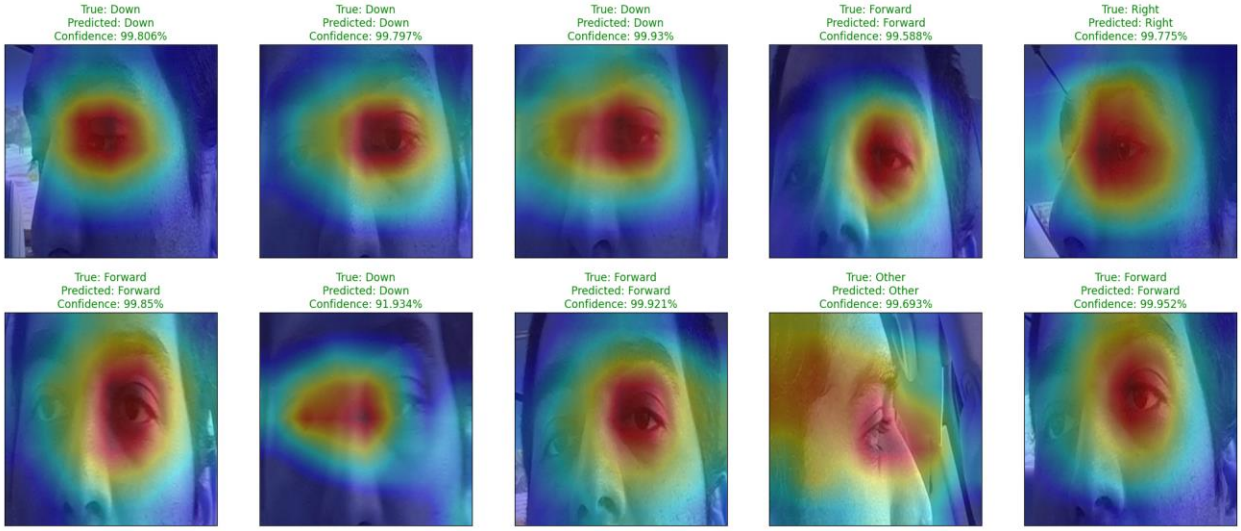


Figure 30: Grad-Cam Heatmap on Model Predictions

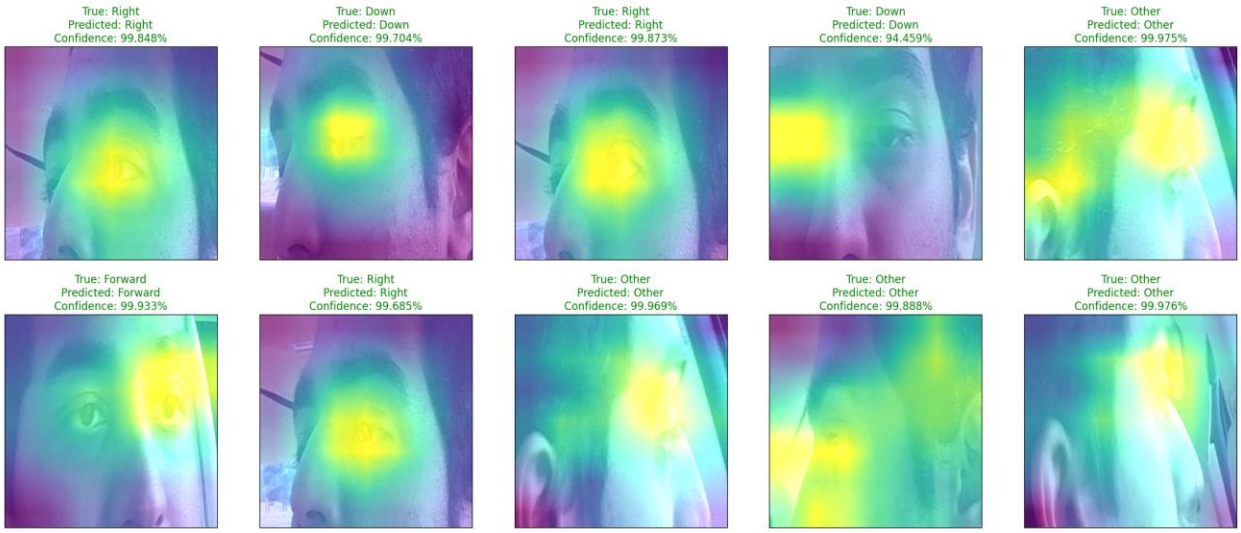


Figure 31: Guided Grad-Cam Interpretation

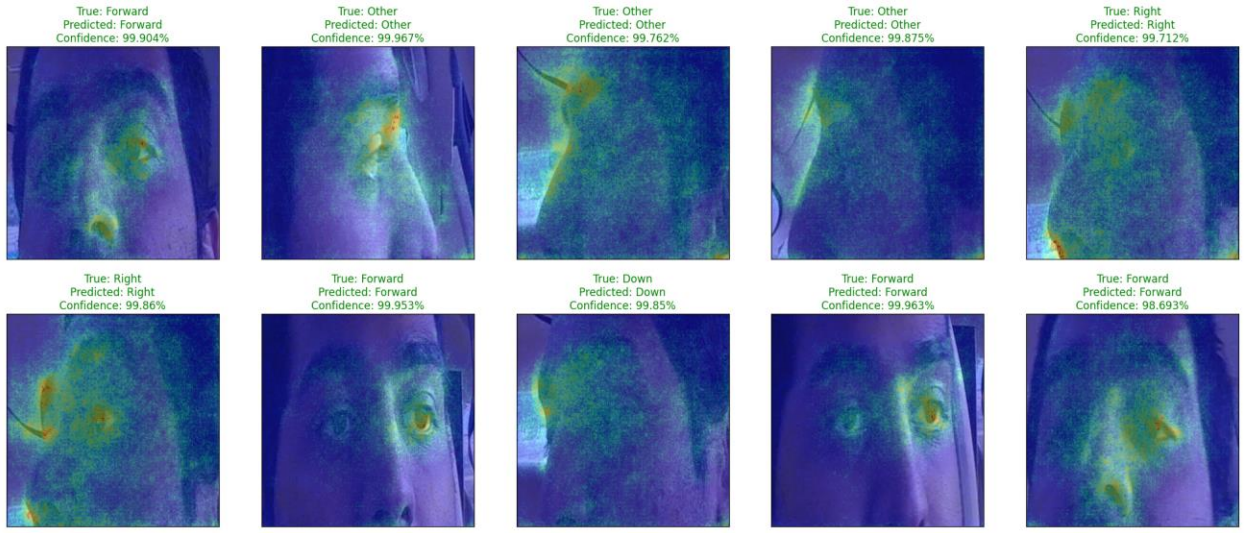


Figure 32: Sailency Map on Model Prediction

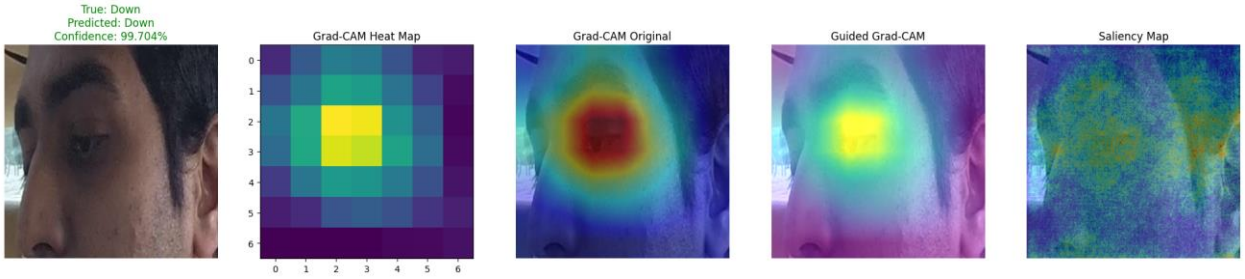


Figure 33: Comparing Interpretation Techniques

By utilizing techniques like Grad-CAM, Guided Grad-CAM, and Saliency Map, we gained a better understanding of the visual cues and regions of interest that influenced the models' predictions. These interpretations provided valuable insights into the models' decision-making processes, enabling us to assess their performance and identify potential areas for improvement.

Additionally, we conducted tests on our model using new images from real-world scenarios, and the results were highly impressive. The majority of the new images yielded accurate predictions, highlighting the robustness of our model. It is important to note that in order to

achieve accurate predictions, the new images must undergo the same preprocessing steps as the images used for training. This includes detecting the driver's face, cropping the upper half of the face, and passing the region of interest to the model for analysis. By adhering to this standardized approach, we ensured consistent and reliable performance in the real-world deployment environment.

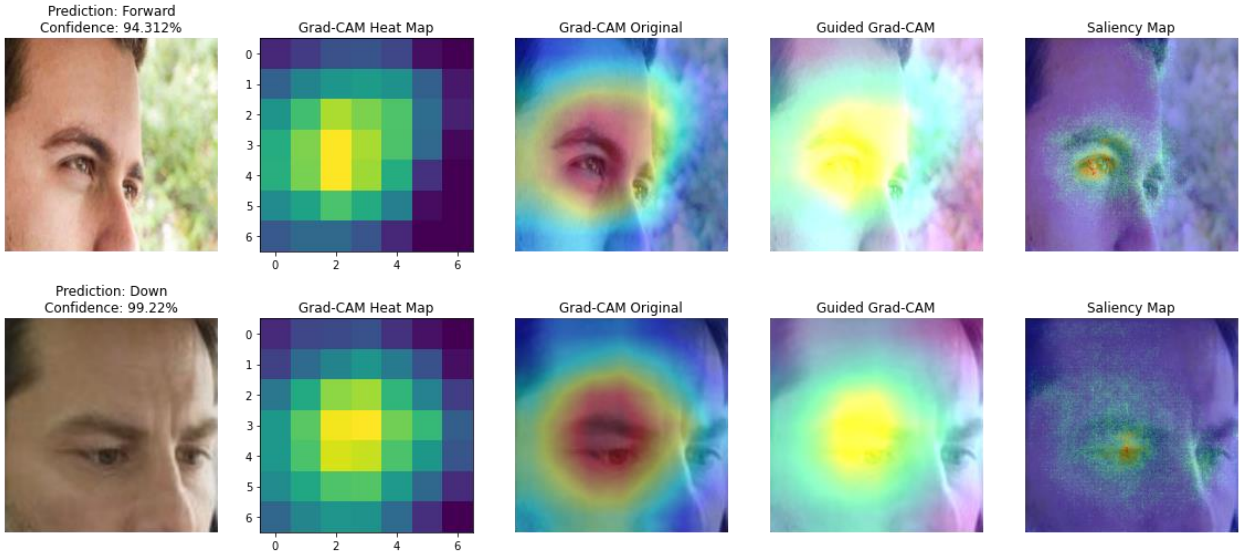


Figure 34: Model Prediction on New Images

4.1.3 Google ML Kit Face and Drowsiness Detection

Google ML Kit provides a comprehensive set of features that enable us to leverage on-device machine learning for enhanced driver monitoring in our app. With its low latency and real-time video processing capabilities, ML Kit allows us to analyze video frames instantly without relying on cloud services. This ensures a seamless and efficient user experience, even in offline environments.

One of the key functionalities we utilize from ML Kit is the face detection API. It enables us to accurately detect and locate drivers' faces in images, providing essential information about facial features and

contours. By leveraging this capability, we can crop the upper portion of the driver's face, which serves as the region of interest for our gaze estimation model.

Furthermore, we used ML Kit for drowsiness detection, it enabled us to determine whether a person has their eyes closed. This feature is crucial for monitoring driver fatigue and ensuring the safety of both the driver and others on the road.

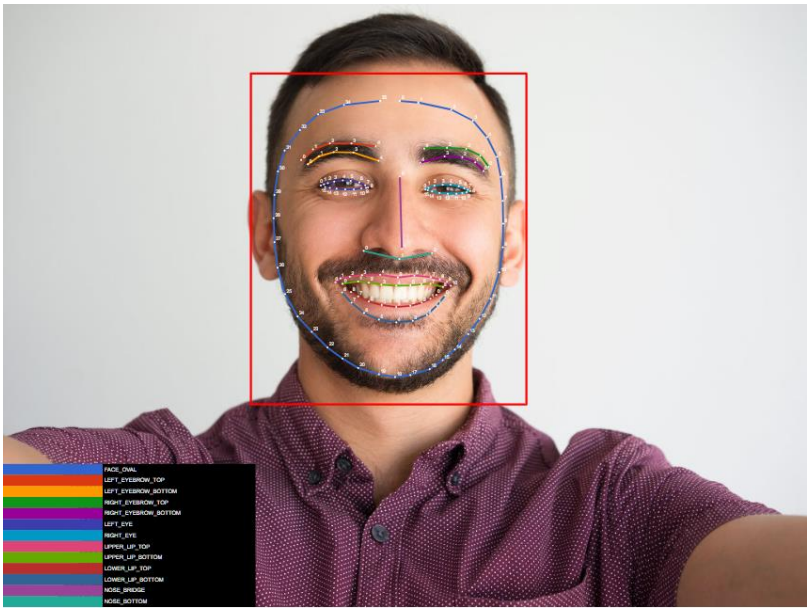


Figure 35: Google ML Kit Face Detection

Additionally, ML Kit allows us to track faces across consecutive video frames by providing a unique identifier for each detected face. This functionality enables us to monitor the driver's face throughout their ride, ensuring continuous and accurate predictions.

By integrating these capabilities from Google ML Kit, we enhance our app's driver monitoring system, enabling real-time analysis of facial features, drowsiness detection, and face tracking for a comprehensive and reliable solution.

4.1.4 Deep Learning Constraints

While we explored and discussed three deep learning models in detail, it's essential to acknowledge that we pursued additional approaches and models to address our driver monitoring problem. In this section, we will highlight the models we attempted to build but ultimately faced challenges or decided not to proceed with in our system.

4.1.4.1 Distraction Detection with DAD Dataset

As previously mentioned, the primary challenge in distraction detection was the limited availability of datasets containing the desired camera angles. To overcome this limitation, we explored using the DAD Dataset, which captured driver images from a more focused angle on the driver, rather than from the side. However, the DAD Dataset presented its own set of challenges.

The DAD Dataset consisted of driver images captured from a focused angle, providing a better perspective on the driver's actions. However, the images in this dataset were captured using an infrared (IR) camera, resulting in grayscale images that appeared dimmed even to the human eye.

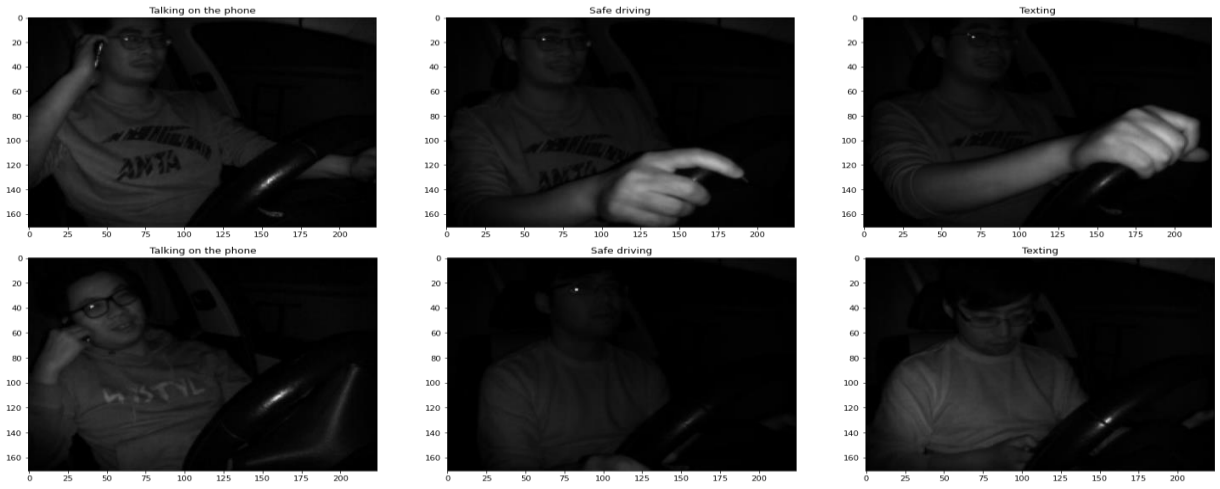


Figure 36: Samples from DAD Dataset

Despite the limitations, we attempted to create a distraction detection model using the DAD Dataset. We employed the same fine-tuned MobileNet Architecture, as introduced earlier, and followed a similar training process, including data balancing, data splitting based on drivers, and data preprocessing. For this purpose, we utilized a small portion of the data, consisting of around 80,000 images, for our model training.

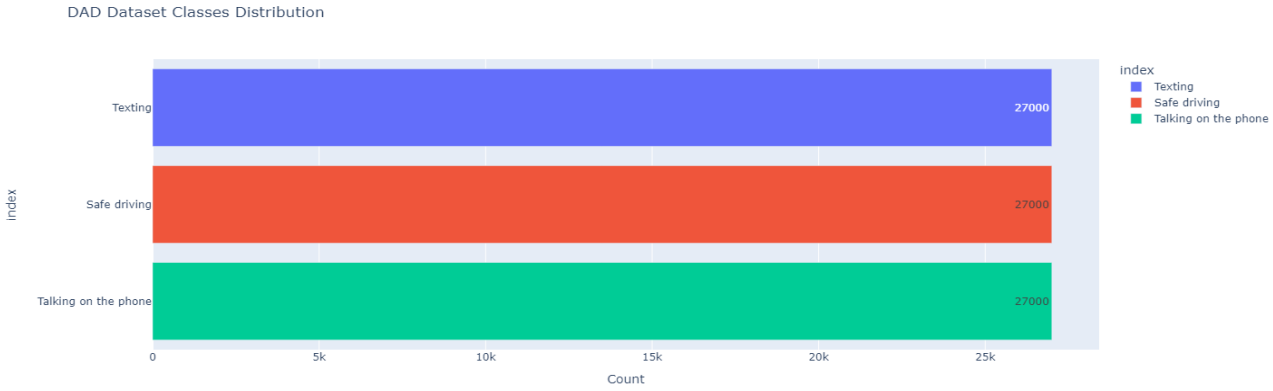


Figure 37: Utilized Portion of DAD Dataset

Regrettably, the performance of this model did not meet our expectations. The best results achieved were an 80% accuracy on the validation set and a 70% accuracy on the testing set.

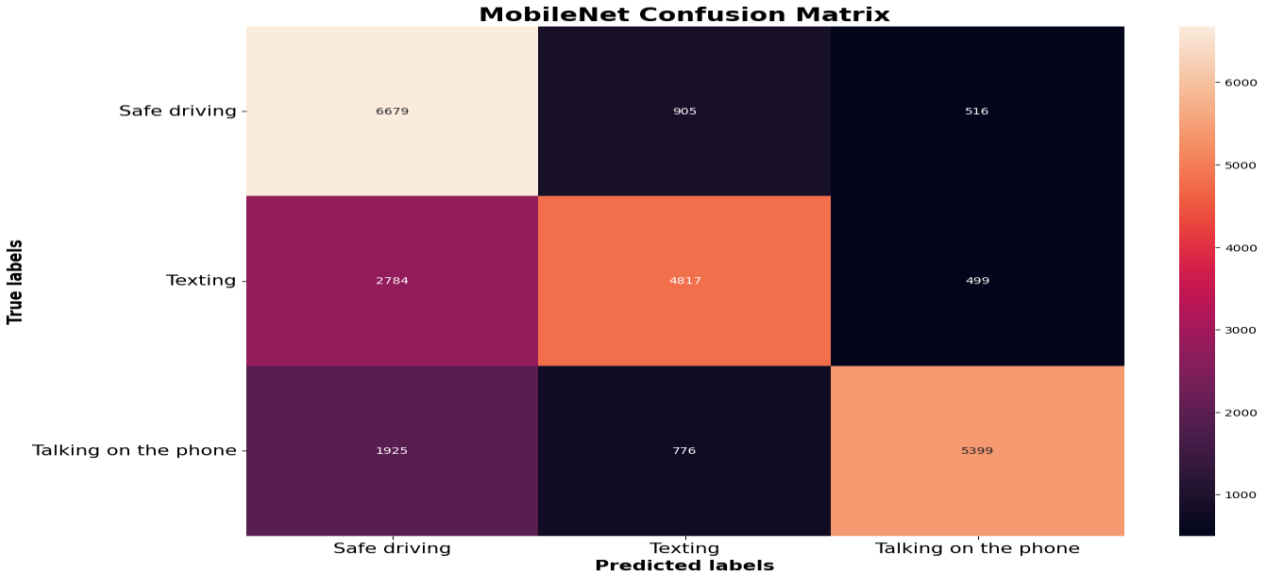


Figure 38: MobileNet Confusion Matrix on DAD Dataset

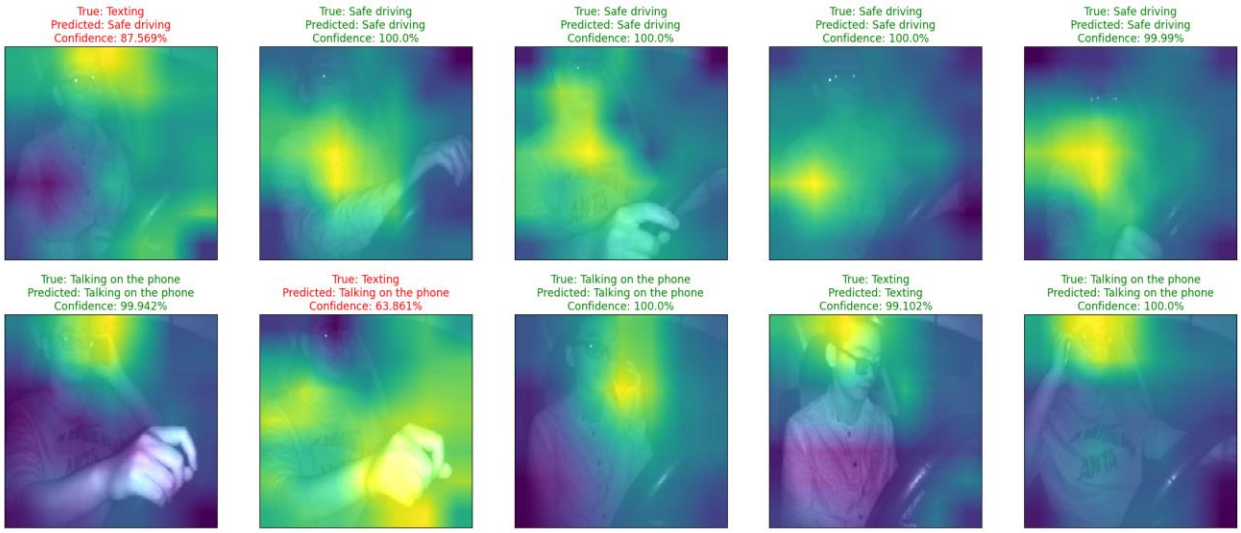


Figure 39: Grad-Cam On DAD Dataset

Upon further examination, it became evident that the model was encountering challenges, particularly in confusing safe driving class images with false predictions. The Grad-Cam interpretation revealed that the model was overfitted to the data and relying on incorrect image features for its predictions in most cases.

Considering the unsatisfactory performance and the challenges with the DAD Dataset, we decided not to continue with this approach. Instead, we focused on the models and techniques that showed more promising results, enabling us to deliver an efficient and accurate driver monitoring system.

4.1.4.2 Custom-Trained Drowsiness Detection

Initially, we attempted to address drowsiness detection using a similar approach to our distraction detection model. The goal was to build a deep learning classifier capable of determining whether the driver's eyes were open or closed. For this task, we experimented with two different models: a Vanilla CNN Model and the Fine-Tuned MobileNet Model, both of which were discussed earlier.

To train these models, we utilized the MRL Eye Dataset, a comprehensive collection of human eye images suitable for classification tasks. The dataset contains infrared images in both low and high resolutions, capturing various lighting conditions and taken with different devices.

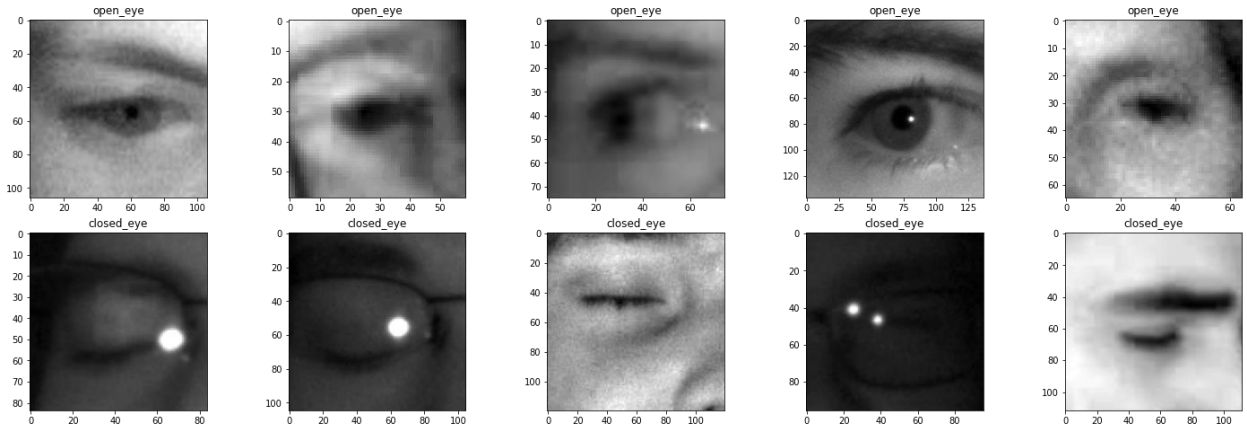


Figure 40: Samples from MRL Dataset

Throughout the training process, we followed a similar approach to what we used for distraction detection, including data balancing, data splitting based on drivers, and data preprocessing. We carefully curated a subset of approximately 50,000 images from the MRL Eye Dataset for training the models.

The results obtained from the trained models were promising, as both achieved an impressive accuracy of about 99% on the testing set. However, despite the high accuracy, we encountered certain limitations that hindered the practicality of deploying these models in real-world scenarios.

One major concern was the complexity and resource-intensive nature of the drowsiness detection model. The process of first detecting the driver's face, then identifying the eyes, and finally passing the eye region to the model posed challenges. This multi-step pipeline led to inaccuracies in face and eye detection, along with slow inference times, making it unsuitable for real-time monitoring on mobile devices.

Considering these deployment constraints and the need for a more convenient and efficient solution, we decided not to proceed with the custom-trained drowsiness detection models. Instead, we shifted our focus to explore alternative approaches like the Google ML Kit discussed earlier, that would deliver accurate and real-time driver monitoring without compromising performance or ease of deployment.

4.1.4.3 Custom-Trained *Fatigue* Detection

In addition to drowsiness detection, we aimed to expand the capabilities of our system by detecting various states of drowsiness, including early drowsiness and fatigue. To achieve this, we utilized the UTA-RLDD (Real-Life Drowsiness Dataset), which consists of approximately 30 hours of video recordings capturing a range of drowsiness signs, from subtle to more obvious cues. However, due to the size of the dataset, we focused on a specific subset called the Driver Drowsiness Dataset (DDD) introduced here, which comprises cropped and extracted facial images of drivers from the RLDD videos.

The DDD Dataset contains around 42,000 images divided into two binary classes: drowsy and non-drowsy. To train our fatigue detection model, we followed a similar approach to our distraction detection model, including data balancing, data splitting based on drivers, and data preprocessing. Additionally, we applied data augmentation techniques to enhance the diversity and robustness of the dataset and as Regularization technique to model training.

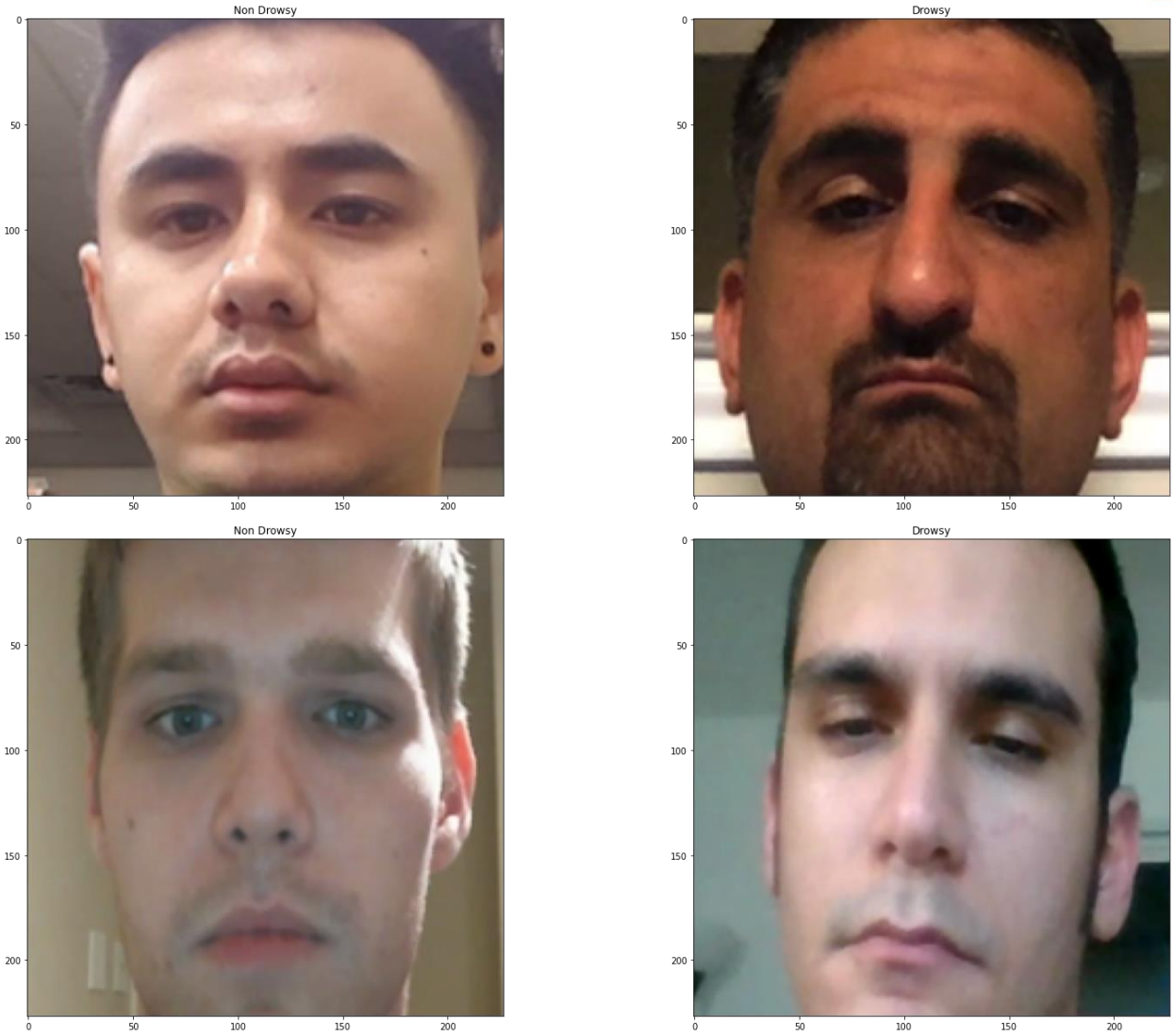


Figure 41: Samples from DDD Dataset

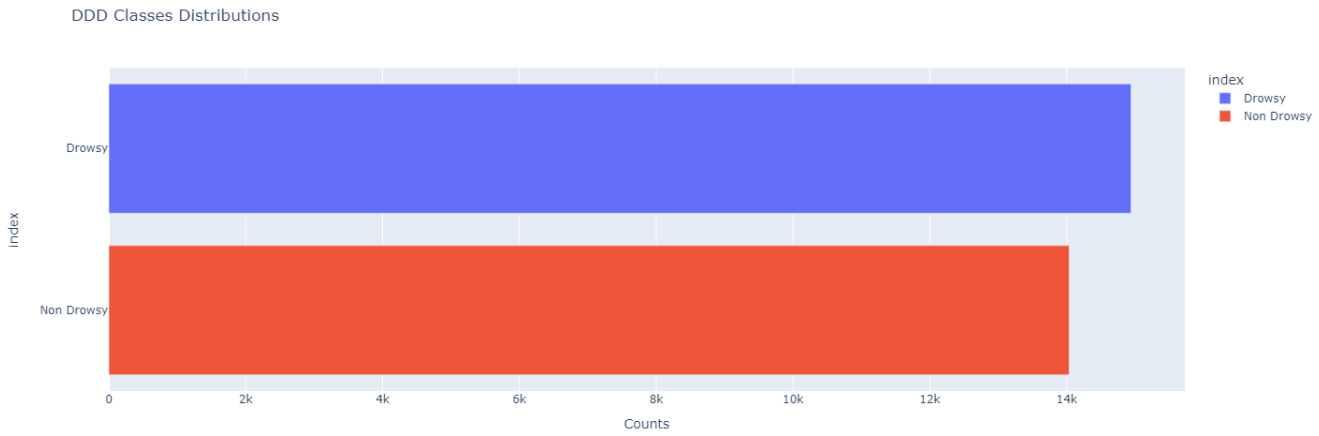


Figure 42: DDD Dataset Classes Distribution

However, for the fatigue detection model, we introduced a change in architecture. Instead of using the Vanilla CNN or MobileNet, we opted for SqueezeNet. SqueezeNet is a specialized deep neural network designed to be smaller than other architectures like AlexNet. It achieves its compact size through various techniques, including using smaller convolution kernels, incorporating fire modules, and employing a global average pooling layer instead of a fully connected layer. This makes SqueezeNet more suitable for deployment on mobile devices with limited computational resources.

However, we encountered challenges with the DDD Dataset, primarily related to overfitting. Despite our efforts, the model's performance did not exceed 80% accuracy on the validation set and 75% accuracy on the testing set. The issue stemmed from the limited variety and distinguishability of the images within the two classes. Even for human observers, discerning the correct class label was challenging due to the subtle differences between drowsy and non-drowsy states captured in the frames. The frames extraction technique used to create the DDD Dataset from the UTA-RLDD proved to be less than ideal for our specific fatigue detection task. The extracted images lacked significant variations

between drowsy and non-drowsy states, leading to difficulties in model generalization.

From the GRAD-CAM interpretation, it is evident that the model is exhibiting signs of overfitting. It seems to rely heavily on specific driver features or make correct predictions by chance, indicating the negative impact of overfitting on the overall performance of the model.

Considering these limitations and the inability to achieve satisfactory performance on the DDD Dataset, we decided not to pursue the custom-trained fatigue detection model further. Instead, our focus shifted towards developing robust methods for accurate and practical detection of drowsiness in real-world mobile device enviroment.

4.2 Deep Learning Testing Application

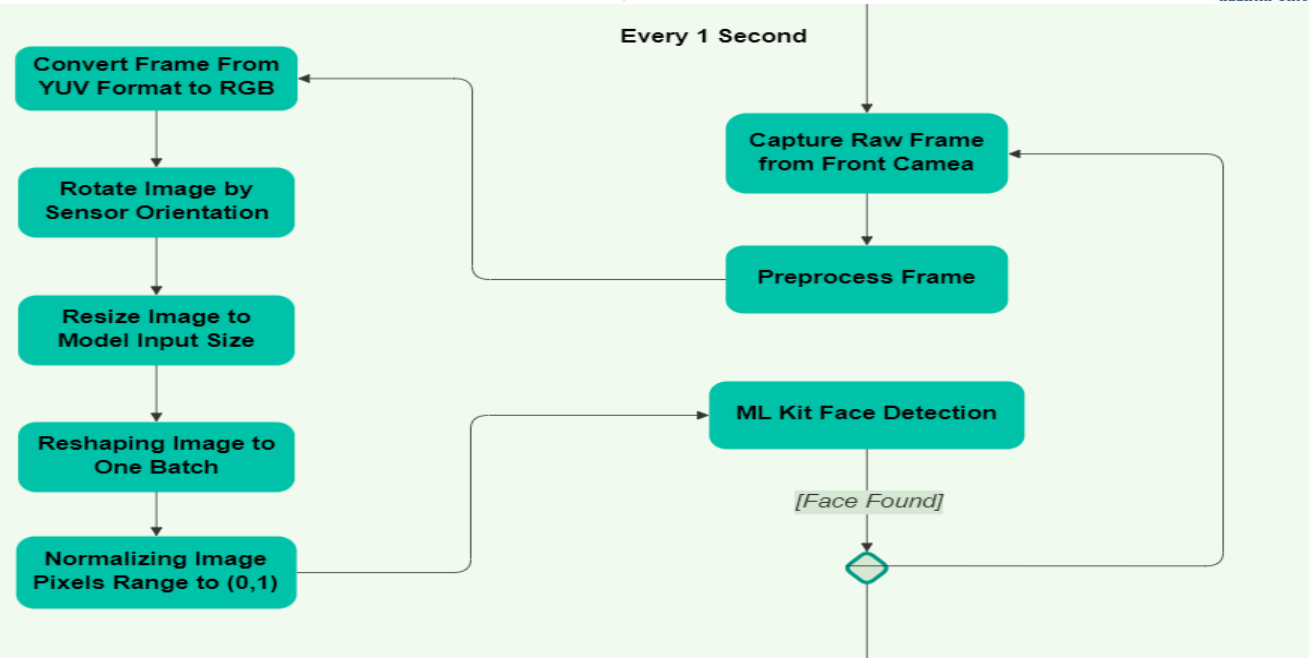
As mentioned before, we developed a dedicated testing application to validate the accuracy of the deep learning monitoring system and evaluate each model's performance in providing accurate predictions.

4.2.1 Deep Learning Deployment

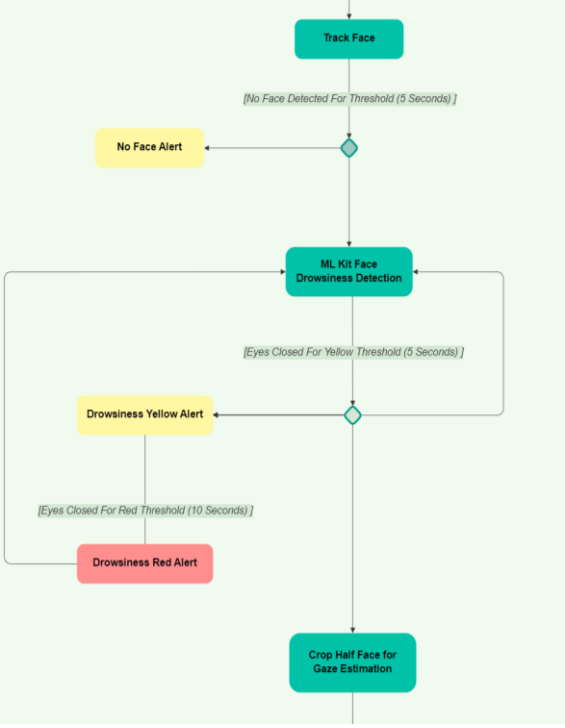
After training and saving our custom-trained deep learning models in the H5 format, the next step was to deploy them in our mobile testing application to assess their performance. To make them compatible with mobile devices, we utilized TensorFlow Lite. By converting our models to TensorFlow Lite models using the default post-training quantization method, we successfully obtained interpreter formats that could be used on mobile devices. Subsequently, we re-evaluated the performance of the interpreters and observed a slight drop of 0.1 to 0.2, which was not a significant concern.

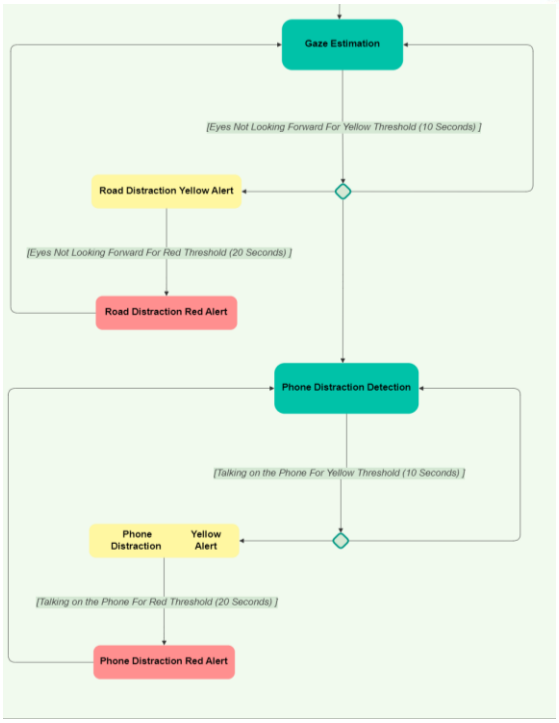
Next, we needed to find a suitable package or plugin for integrating the TensorFlow Lite models into our Flutter application. This proved to be a challenging task as we needed a plugin that not only supported the tf lite format but also offered the required functionality for accurate predictions.

Let's examine the sequence of the deep learning inference pipeline on mobile devices:



1. The image is captured from the front camera of the phone and undergoes several preprocessing steps before being fed into the models. Initially, the image is converted from the raw frame YUV format to RGB format to ensure compatibility with Flutter plugins. Additionally, the image is rotated based on the mobile phone's sensor orientation and cropped to match the expected input size of our models. It is then reshaped into batches and normalized, similar to the training data, with pixel values ranging from 0 to 1.



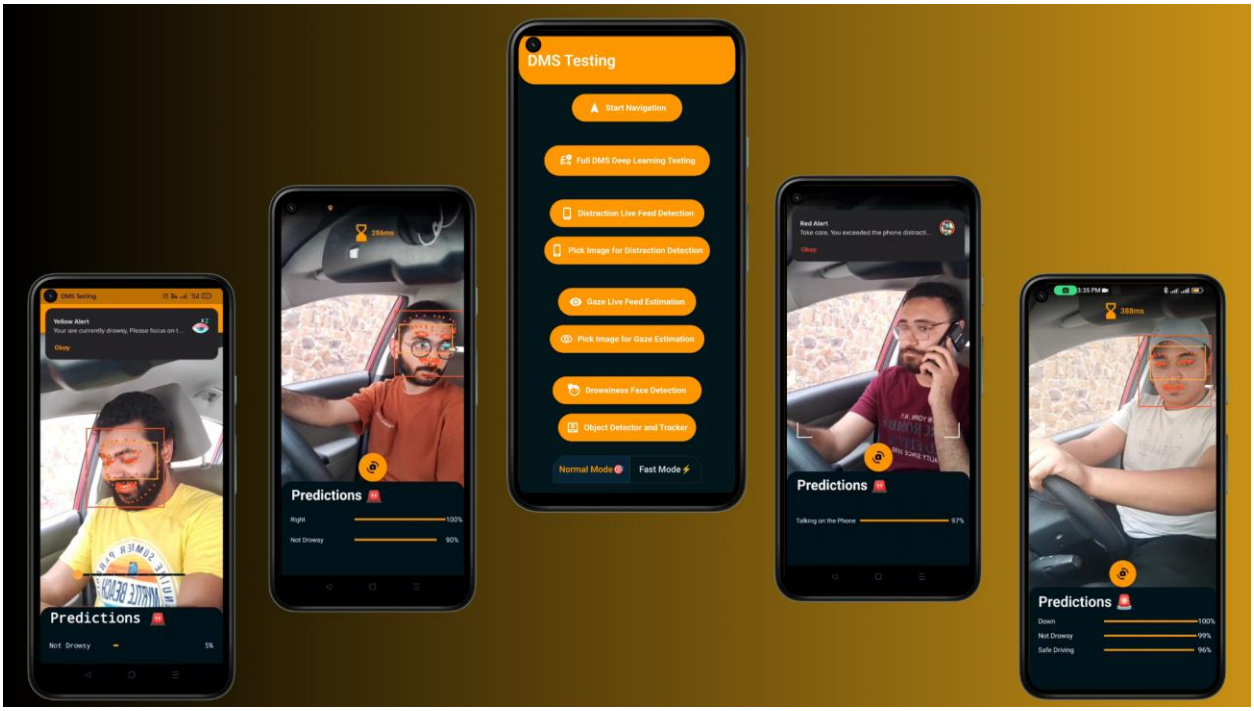


2. The Google ML Kit face detection model is employed to determine if the driver's face is present in the image. If the face is not detected, the deep learning monitoring ceases, and the driver is alerted to adjust the front camera angle. On the other hand, if the driver's face is detected, the same face detection model is used to detect drowsiness by monitoring the driver's eye state.
3. Additionally, the gaze estimation model is utilized to determine the driver's focus on the road. Leveraging the face landmarks detected by the Google ML Kit model, we accurately select the relevant region of the face for gaze estimation.
4. Lastly, we employ the distraction detection model to identify any phone distractions by analyzing the entire processed frame from the front camera, such as texting or talking activities.

The deep learning models are executed sequentially, rather than in parallel, every second. On most mobile devices, the average inference time for our deep learning monitoring is approximately 300 milliseconds, utilizing only one processor core. If the inference time is less than one second, we wait for the remaining time to ensure inferences are made every second. However, if the hardware of the mobile device cannot perform inferences within one second, we proceed with the next inference cycle as soon as possible.

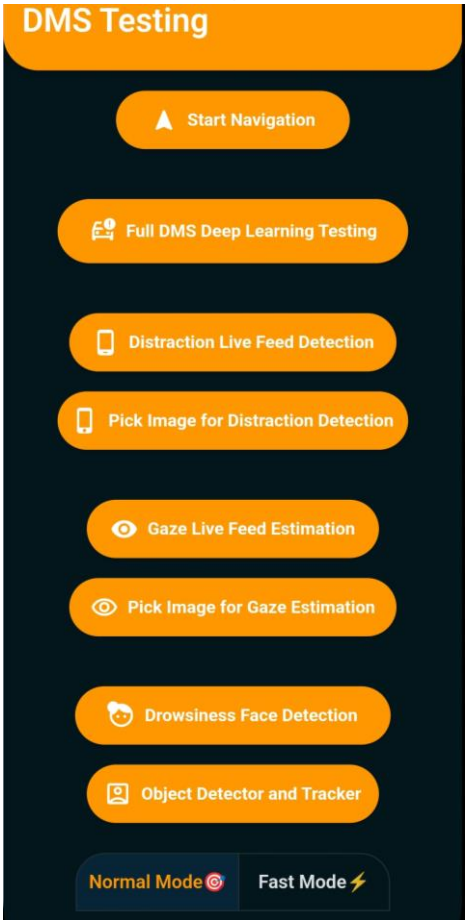
4.2.2 Testing Application Showcase

4.2.2.1 Testing Application UI



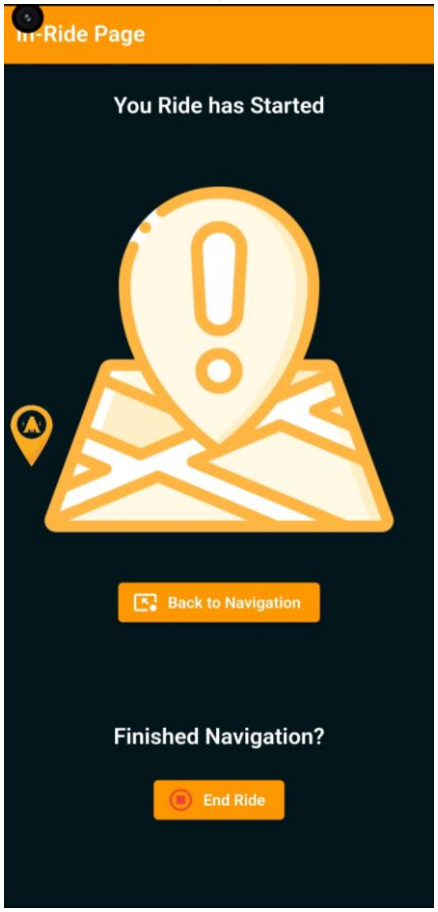
4.2.2.2 Home Page

When Normal Mode is activated the deep learning models will run on a single processor, on the other hand the fast mode will run the deep learning models on all available processors in the mobile device. However, we find that for most phones normal phones makes faster prediction than fast mode, so we continued to make our prediction only on a single processor.



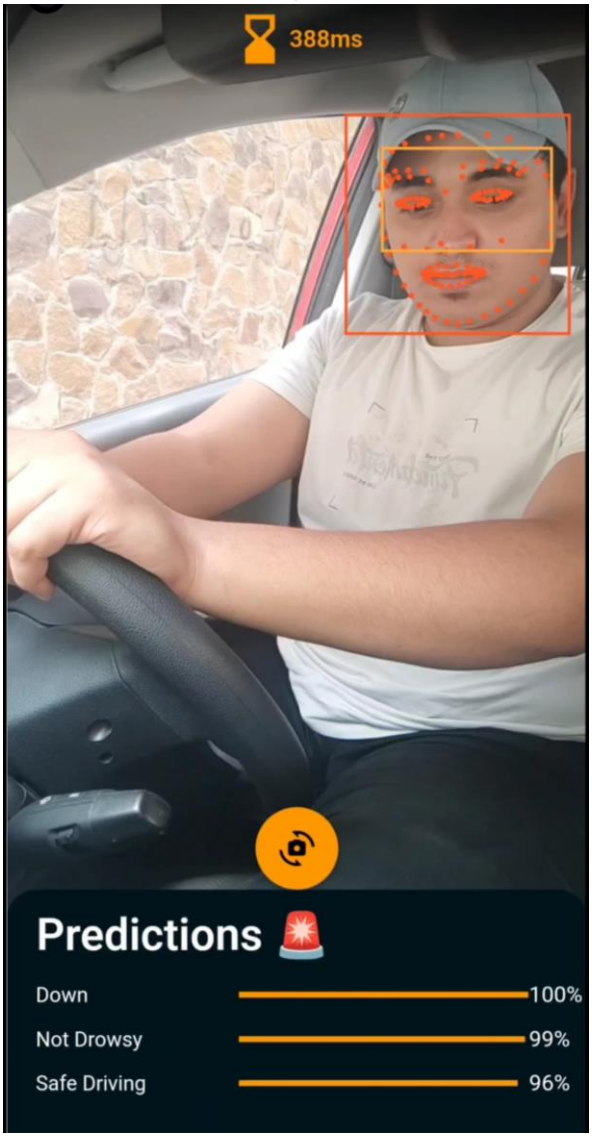
Start Navigation Page

In here we simulate and test how our app deep learning monitoring will work on the background as required for the Mobile Application.



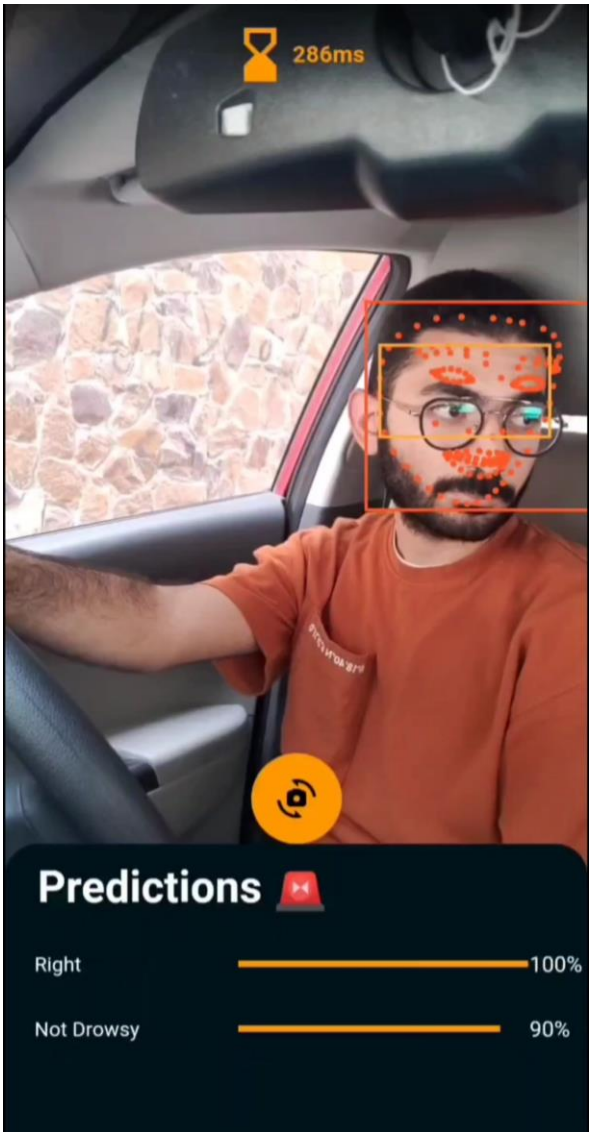
4.2.2.4 Full Deep Learning Testing Page

Here we test all of our deep learning and visualize each model prediction in real-time.



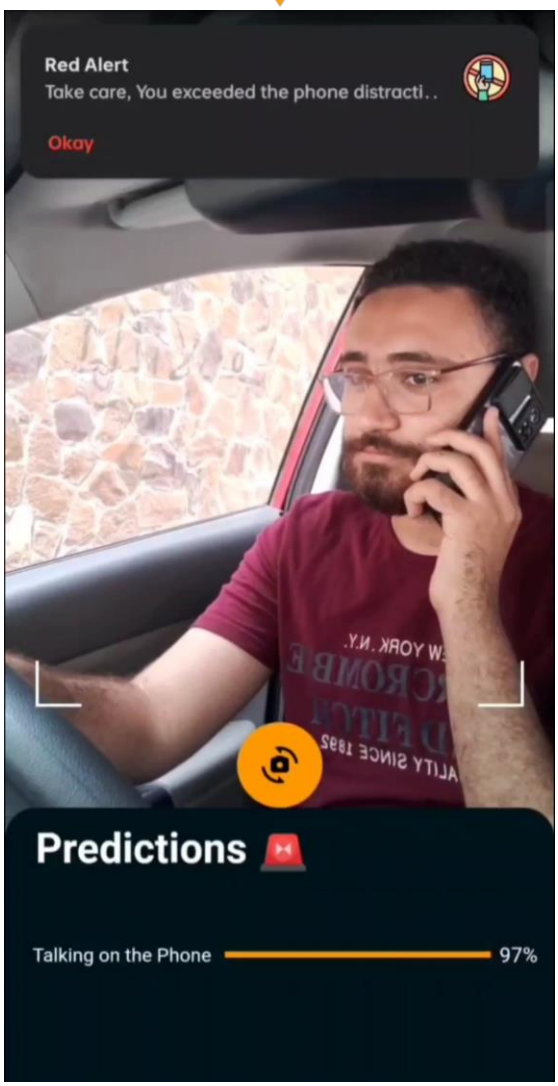
Gaze Estimation Testing Page

Here we test the performance of our custom-trained gaze estimation model along with google ml kit face detection, tracking, cropping gaze region of interest in real-time.



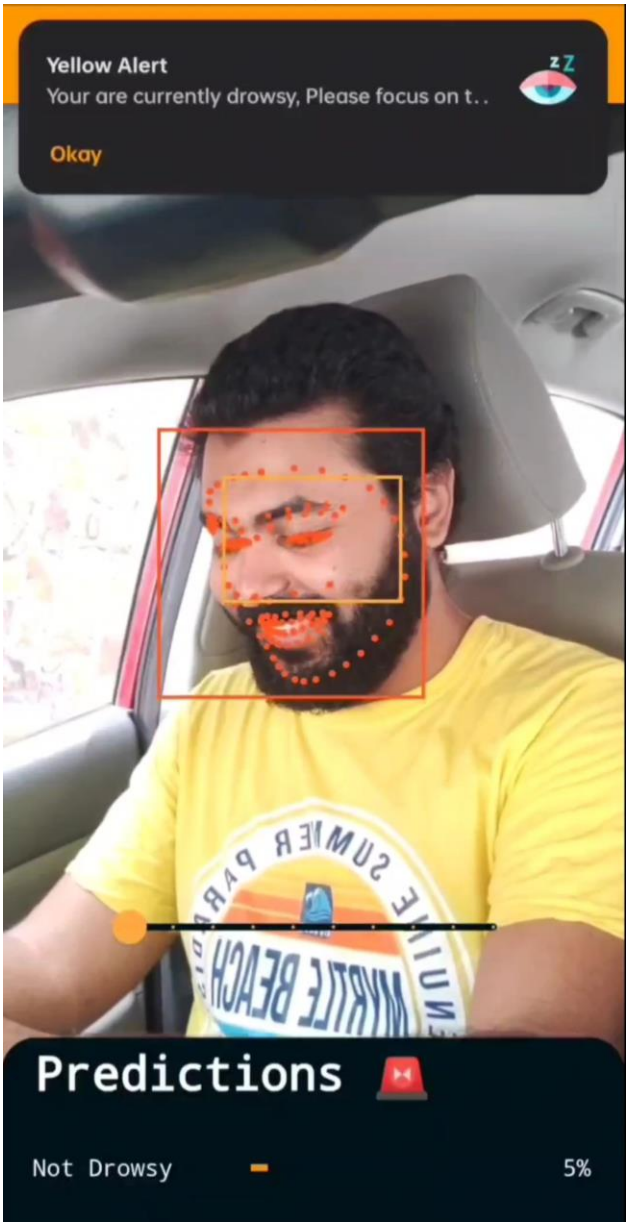
Distraction Detection Testing Page

Here we test the performance of our custom-trained distraction detection model in real-time.



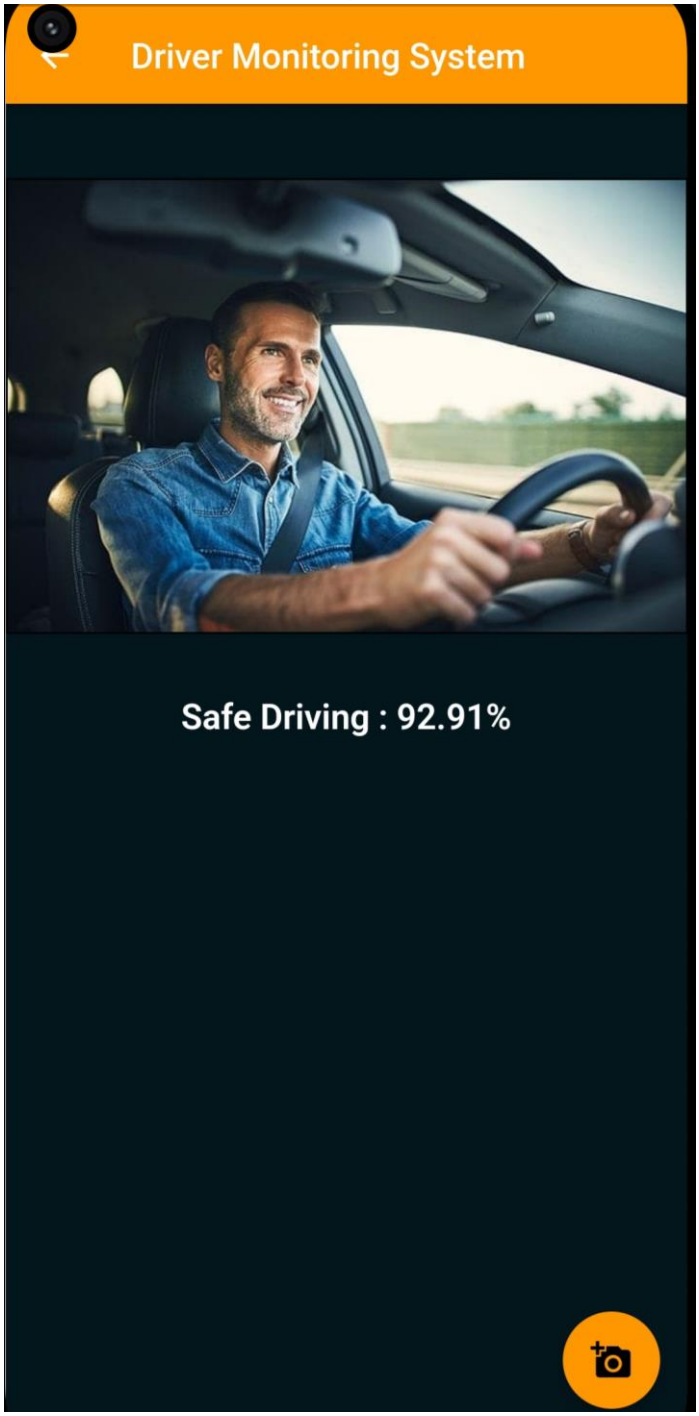
4.2.2.4 Drowsiness Detection Testing Page

Here we test the performance of the Google ML Kit Face Detection and Tracking along with drowsiness detection.



4.2.2.4 Gaze & Distraction Single Image Prediction Testing Page

In these Pages we test the performance of our two custom-trained models on single image prediction.


Driver Monitoring System
Safe Driving : 92.91%

4.2.2.4 Object Detector Testing Page

In this page we test the performance of the Google ML Kit Object Detector and Localizer for the driver, but we decided to not continue with it due to insufficient prediction to the driver in most cases as mentioned before.

4.3 Drivers Mobile Application

Our mobile application is developed using the Flutter framework, which offers numerous features and advantages for building clean code architectures following design patterns. We have been meticulous in adhering to these principles, including SOLID Principles, as well as implementing other design patterns such as Singleton Design Pattern for our Monitoring Services and utilizing the BLoC state management design pattern to handle real-time data displayed in the maps UI. Additionally, we have adopted the MVC pattern, which is used for working with Cloud Firestore and handling most operations within our app.

4.3.1 Cloud Firestore

We have chosen to leverage Cloud Firestore due to its inherent advantages. The document-based design of Firestore makes it convenient for organizing our collections and accessing the required information seamlessly. When implementing operations related to Firestore, we have ensured that they follow the MVC design pattern. In this pattern, there is a model responsible for mapping data from Firestore's JSON documents. The controller of each collection handles operations such as adding, editing, deleting, and reading data, while some collections include query functions to retrieve specific information. In the views, the appropriate function from the collection controller is called to retrieve the desired data. By following this design pattern, we maintain a structured and efficient approach to interacting with Firestore within our application.

4.3.2 Google Maps API

We will explain the use-cases we implemented in the maps section of our mobile application and how we employed BLoC state management to ensure consistent and efficient code. Furthermore, we will describe how we utilized the Google Cloud Console to access the Google Maps API services.

4.2.2.1 BLoC State Management

In order to manage the state of our mobile application and ensure code efficiency and consistency, we employed BloC state management. BLoC separates the presentation layer (widgets) from the business logic layer (BLoCs), which makes the code easier to read, understand, and maintain. Additionally, BLoC allows for reusability across different widgets and screens, reducing code duplication and making the app more efficient. BLoCs are also easy to test because they don't depend on the UI. You can write unit tests for the BLoC logic to ensure that it works as expected. Finally, BLoC provides a way to manage the state of the app by using streams to emit new states as the app changes, making it easier to update the UI based on changes in the app state.

4.2.2.2 Map Cubit Class

The Map Cubit class is a crucial component of our mobile application's state management. It contains most of the logic of the maps section and uses the BLoC pattern to ensure that the state remains consistent throughout the application. We've designed the MapCubit class to ensure that the state of the map remains consistent throughout the application. When we call one of the methods to interact with the map, the class updates the state of the map accordingly. If an error occurs at any point, the class updates the state to reflect the error, ensuring that the application remains stable and responsive.

4.2.2.3 Google Cloud Console

To use the Google Maps API service in our mobile application, we used the Google Cloud Console to manage our API key and access the various services provided by Google Maps. The Google Cloud Console provides a user-friendly interface that allows us to manage our API key and monitor usage. One of the advantages of using the Google Cloud Console is that it offers a free trial of up to $200 per month. This

allowed us to use the Google Maps API service during development and testing. With the Google Maps API service, we were able to integrate maps and location-based services into our application, enabling us to provide users with accurate and up-to-date information about their location and nearby places of interest.

4.2.2.4 Google Maps API Use-cases in Driver Role

4.2.2.4.1 View Current Location on Maps

Drivers should be able to view their current location on maps within the system.

To enable this feature, we used the Geocoding API provided by the Google Maps API service in our mobile application. This API is used to convert geographic coordinates (latitude and longitude) into human-readable addresses and vice versa.

To implement this feature, we first obtained the current latitude and longitude coordinates of the driver's location using the device's location services. We then made a request to the Geocoding API with the coordinates of the location. The API provided us with a response containing information about the location, such as the address, postal code, and other relevant information. We were then able to display this information on the map, allowing drivers to view their current location in a human-readable format.

The Geocoding API was crucial in enabling us to provide accurate location-based services to our users. By converting geographic coordinates into human-readable addresses, we were able to provide users with relevant information about their location and nearby places of interest. This feature was essential in enabling drivers to navigate to their destination and find nearby locations of interest, enhancing their overall experience with our application.

4.2.2.4.2 Search for Destination

Drivers should be able to search for a destination on maps and set it as the destination for their ride.

To enable this feature, we used the Places API provided by the Google Maps API service in our mobile application. This API is used to search for places and retrieve information about them.

We used two features of the Places API to implement this feature.

The first feature is autocomplete place names, which provides an autocomplete feature that allows users to enter the name of a place and get suggestions as they type. To use this feature, we made requests to the API with the user's input, and the API provided us with a response containing a list of suggested places. This feature was essential in enabling drivers to quickly and easily find the location they were looking for.

The second feature of the Places API that we used was retrieving detailed place information. This feature provides detailed information about a place, such as its address, phone number, website, photos, reviews, and more. To use this feature, we made requests to the API with the place ID of a location, and the API provided us with a response containing detailed information about the place. This feature was crucial in enabling us to provide users with all the information they needed about a place, enabling them to make informed decisions about their destination.

The Places API was crucial in enabling us to provide users with accurate and up-to-date information about places of interest. By using the autocomplete feature, we were able to help users find the location they were looking for quickly and easily. By retrieving detailed place information, we were able to provide users with all the information they

needed about a place, enabling them to make informed decisions about their destination. This feature was essential in enhancing the overall experience of our application for drivers.

4.2.2.4.3 View Best Route on Maps

Drivers should be able to view the best route on maps within the system.

To enable this feature, we used the Directions API provided by the Google Maps API service in our mobile application. This API is used to get directions between two or more locations and display a route on the map.

To implement this feature, we made requests to the Directions API with the starting and ending locations of a route. The API then provided us with a response containing information about the route, such as the distance, estimated travel time, and turn-by-turn directions. By using this information, we were able to display the best route on the map, enabling drivers to navigate to their destination quickly and safely.

The Directions API was crucial in enabling us to provide users with accurate and up-to-date information about the best route between two or more locations. By displaying the route on the map, we were able to help users navigate to their destination quickly and safely. This feature was essential in enhancing the overall experience of our application for drivers.

4.2.2.4.4 View Radars on Maps

Drivers should be able to view the location of radars on maps within the system.

4.2.2.4.5 Redirect to Google Maps for Navigation

Drivers should have the ability to be redirected to Google Maps for navigation to their destination. Along with starting a floating bubble that indicates that the system is still running in the background.

4.2.2.4.6 Receive Alerts before Approaching Radars (Maps Monitoring)

The purpose of this use-case is to enhance road safety for drivers by providing alerts before approaching radars. Drivers should receive alerts when approaching radars, along with specialized alerts when they exceed the speed limit or when they are in an unsafe driving state before approaching radars. To achieve this, we created two services: Maps Ride Monitor and Speed Cam Monitor.

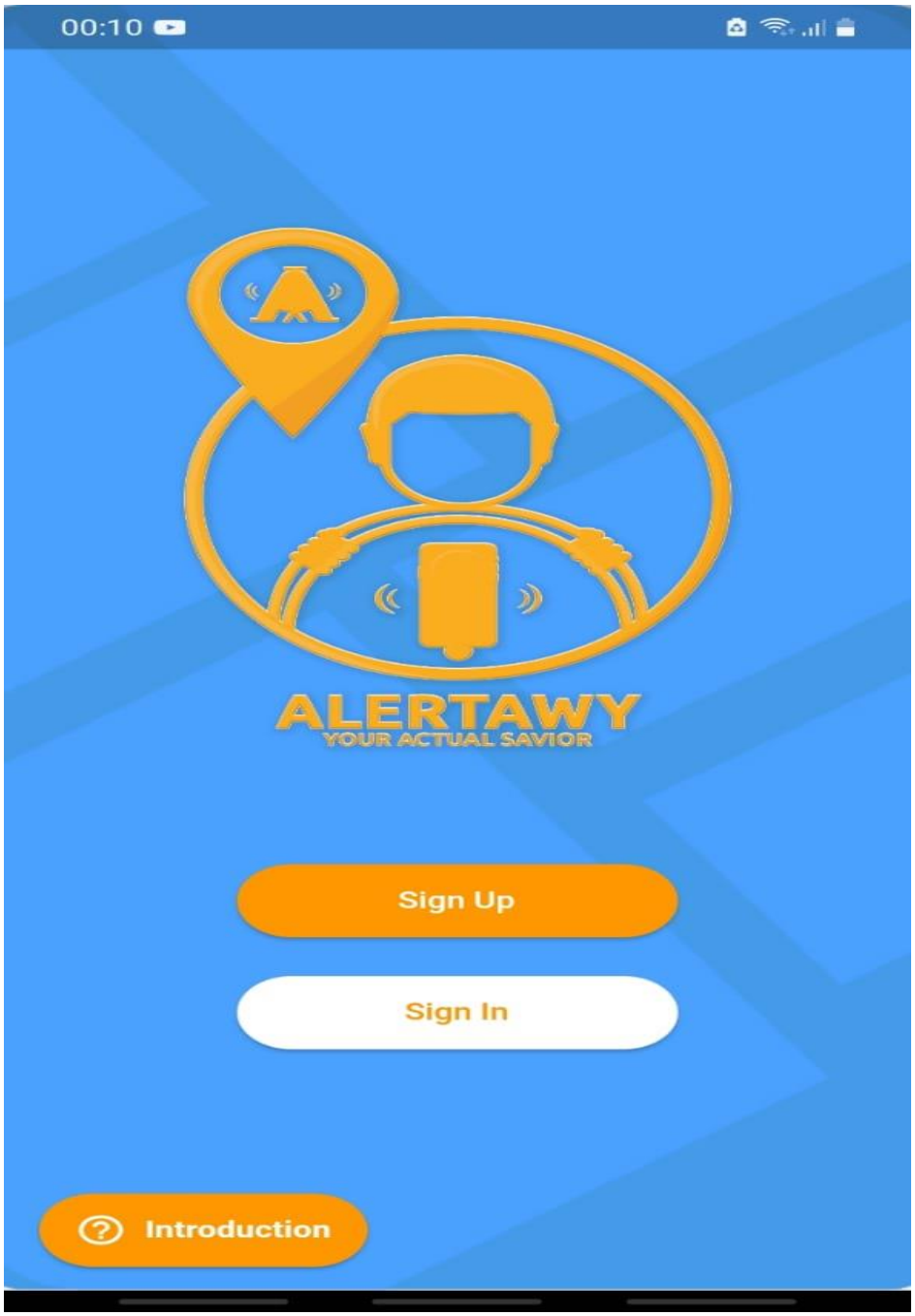
The Maps Ride Monitor service is a singleton class used to monitor the ride of a driver on a map. It has several properties and methods that enable it to track the driver's position, distance traveled, and speed, as well as alert the driver of any speed cameras ahead. The class monitors the ride of a driver every 5 seconds to keep track of the distance traveled and current speed. By doing this, the class can provide timely alerts to the driver if they approach a speed camera ahead. The class guarantees that the driver is notified at least 200 meters before a speed camera, as long as their speed is 144km/h or lower. This helps to ensure the safety of the driver and other road users by preventing accidents due to speeding.

The Speed Cam Monitor class is also a singleton class used to monitor a driver's speed and distance from speed cameras on a ride. It calculates the driver's distance from each speed camera in the ride and provides timely alerts to the driver if they approach a speed camera. The class depends on the Maps Ride Monitor service to provide updates on the driver's current location and speed. It also uses the Alerts Service to create speed camera alerts for the driver based on this information. The

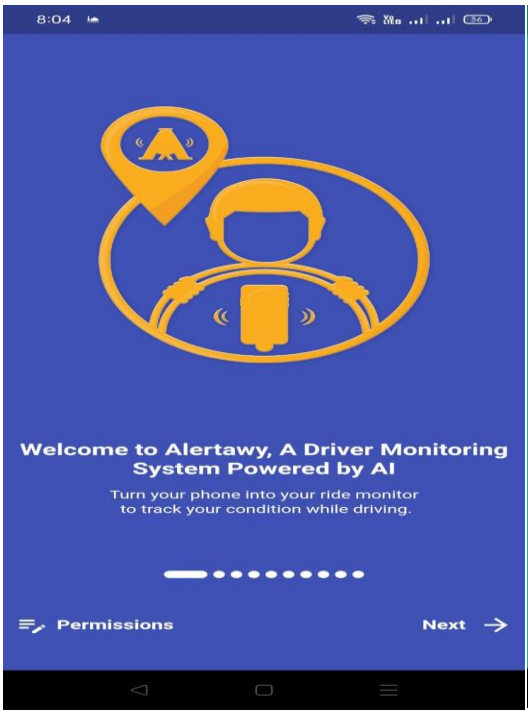
class uses the Deep Learning Monitoring Services to detect driver distractions and trigger red alerts if necessary.

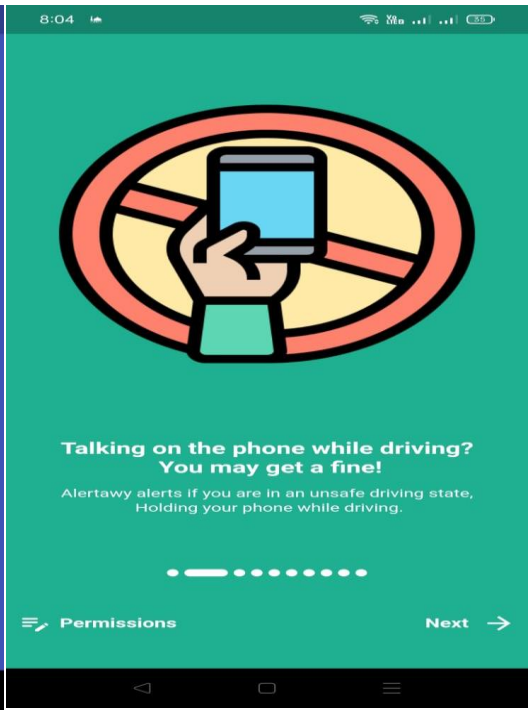
4.3.1 Start Pages

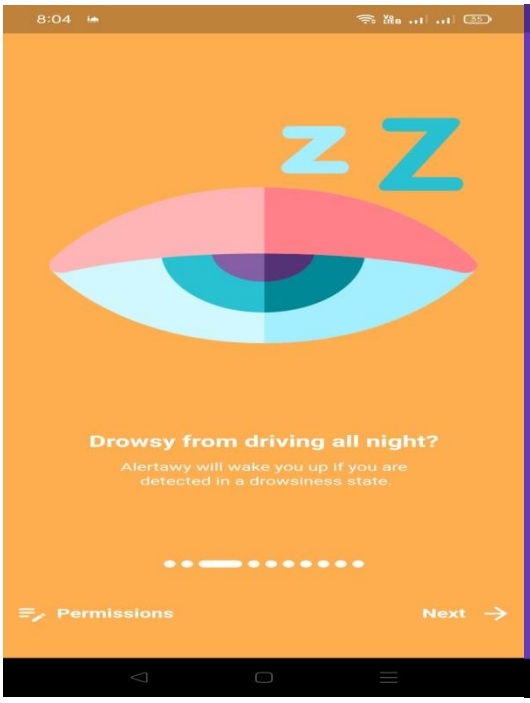
Firstly, when we open the application, we have three options sign up, sign in & introduction.

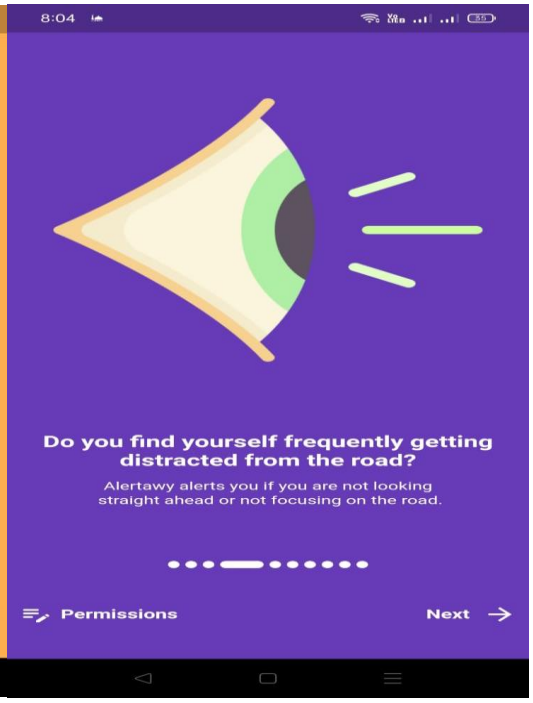


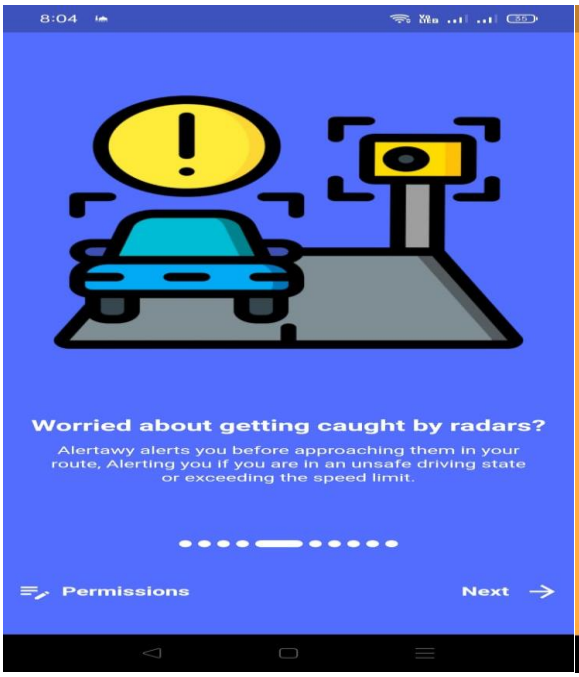
4.2.1.1) Introduction

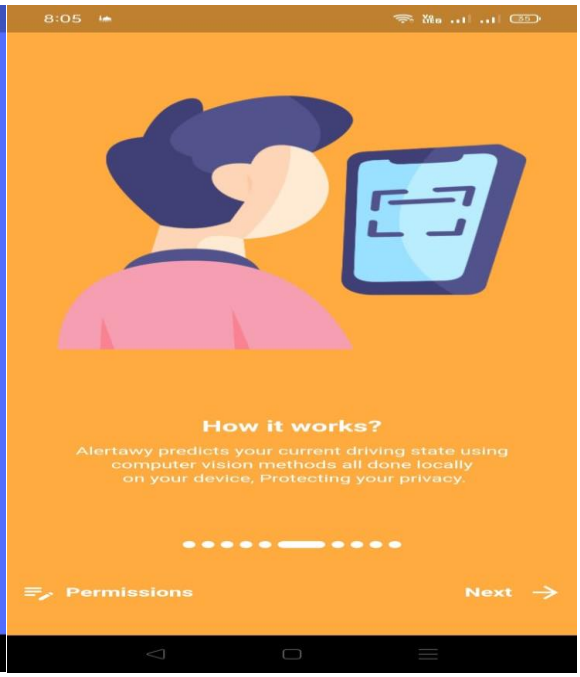


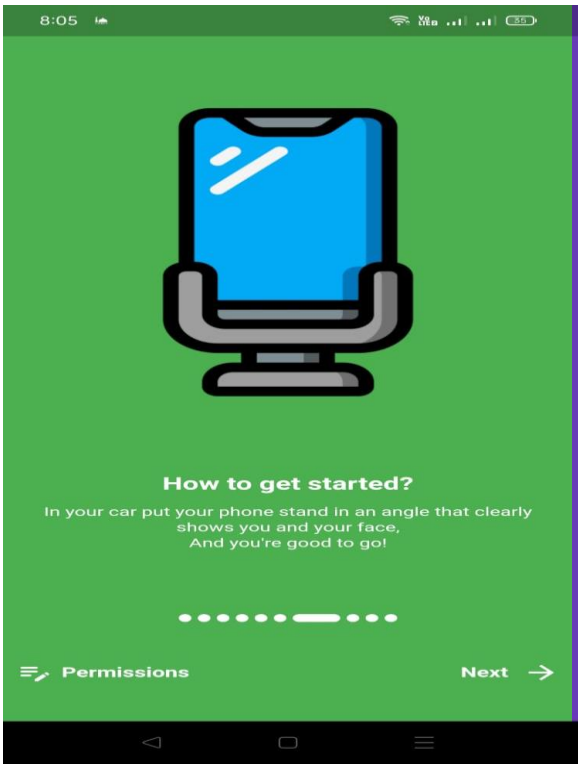


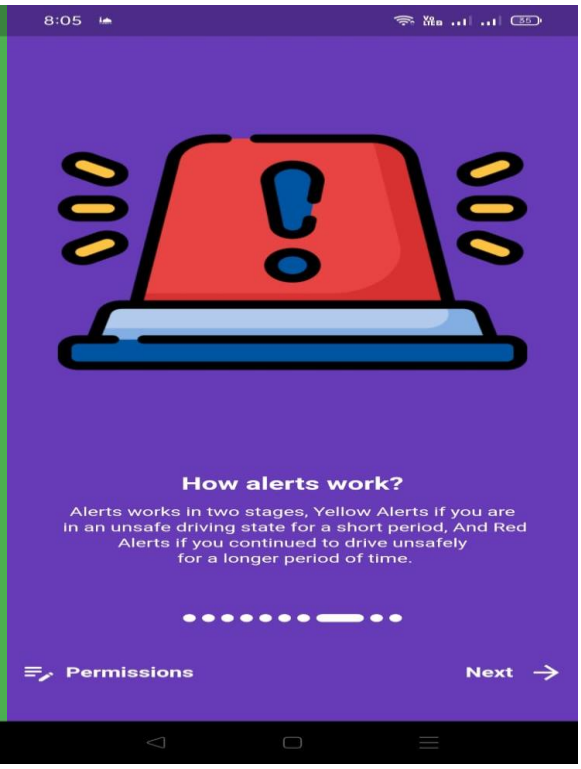


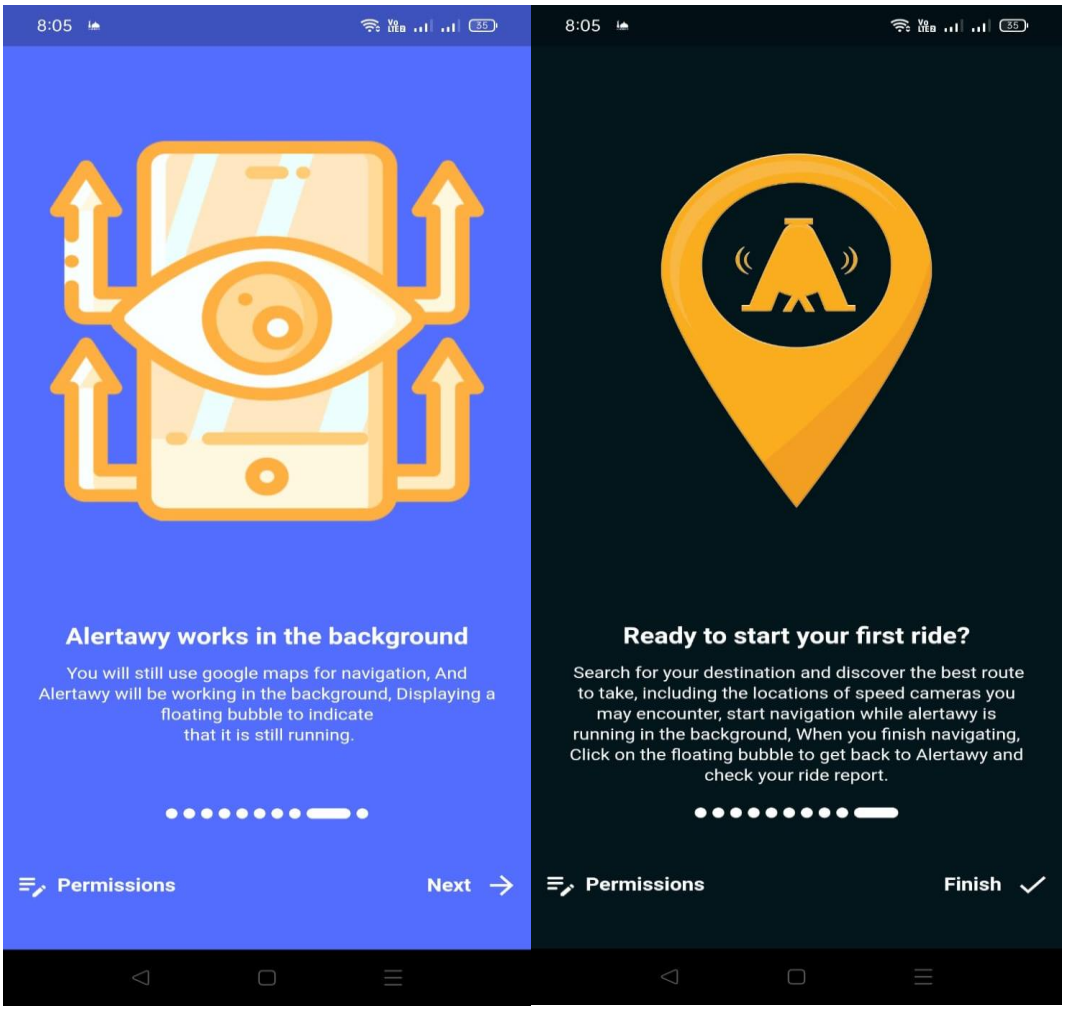



8:04
Worried about getting caught by radars?
Alertawy alerts you before approaching them in your route, Alerting you if you are in an unsafe driving state or exceeding the speed limit.
Permissions
Next


8:05
How it works?
Alertawy predicts your current driving state using computer vision methods all done locally on your device, Protecting your privacy.
Permissions
Next


8:05
How to get started?
In your car put your phone stand in an angle that clearly shows you and your face,
And you're good to go!
Permissions
Next

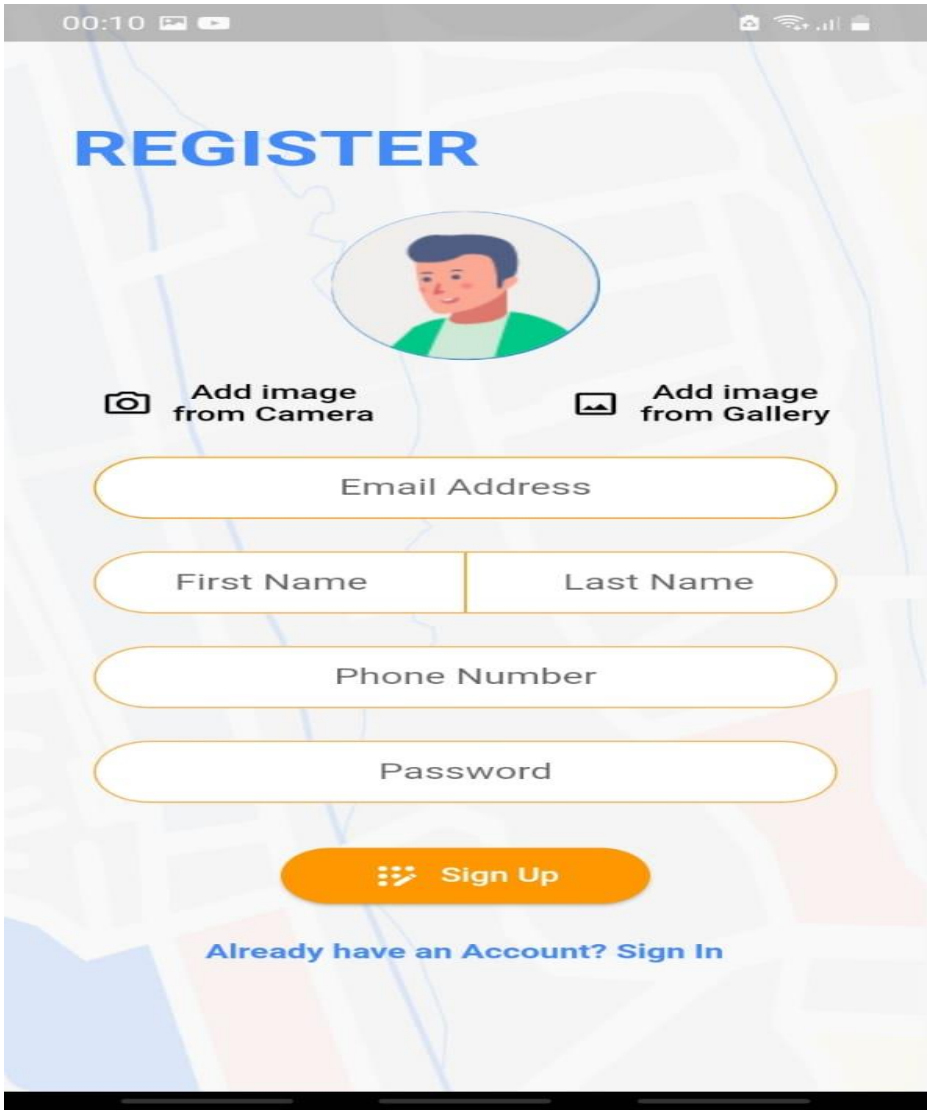

8:05
How alerts work?
Alerts works in two stages, Yellow Alerts if you are in an unsafe driving state for a short period, And Red Alerts if you continued to drive unsafely for a longer period of time.
Permissions
Next


8:05
Alertawy works in the background
You will still use google maps for navigation, And Alertawy will be working in the background, Displaying a floating bubble to indicate that it is still running.
Permissions
Next
8:05
Ready to start your first ride?
Search for your destination and discover the best route to take, including the locations of speed cameras you may encounter, start navigation while alertawy is running in the background, When you finish navigating, Click on the floating bubble to get back to Alertawy and check your ride report.
Permissions
Finish

4.2.1.2 Register

On this page we enter email, first name, last name, EGY phone number, password & image picked from gallery or picked it from camera (optional).

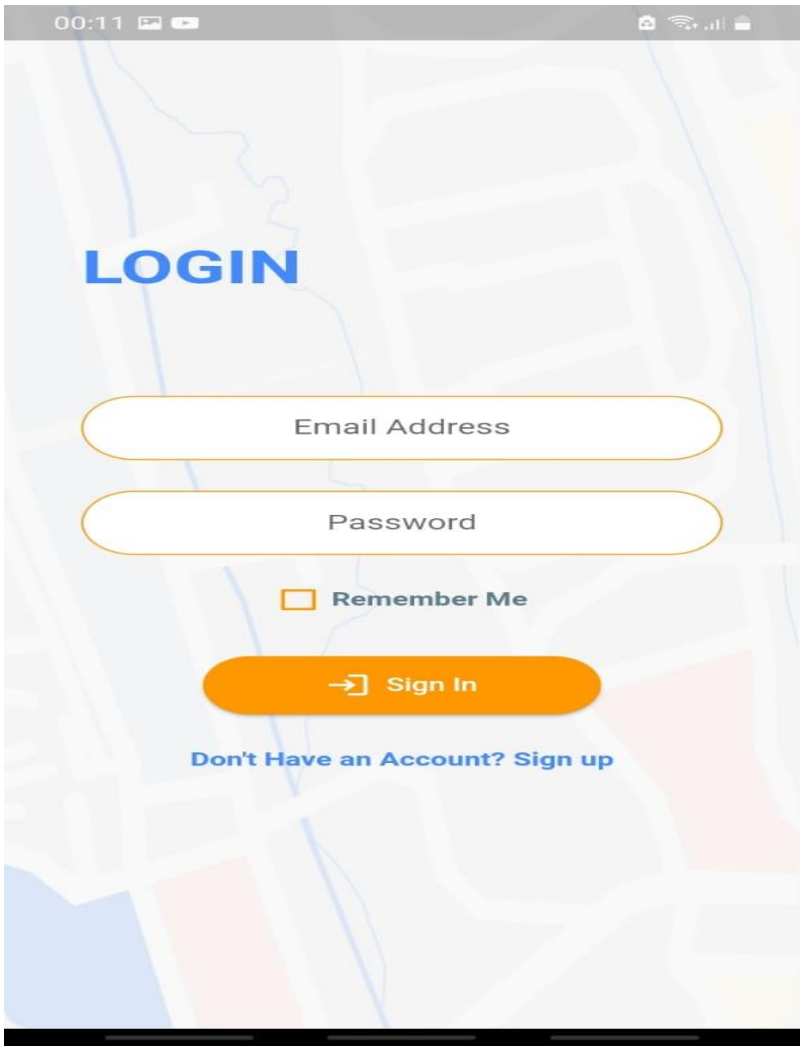
then validate all this information from some safety functions if something doesn't satisfy the application rules the application will ask for correct information, else these information & image link saved in firebase firestore also the image is stored in firebase storage, save email & password in firebase authentication.



4.2.1.3 Login

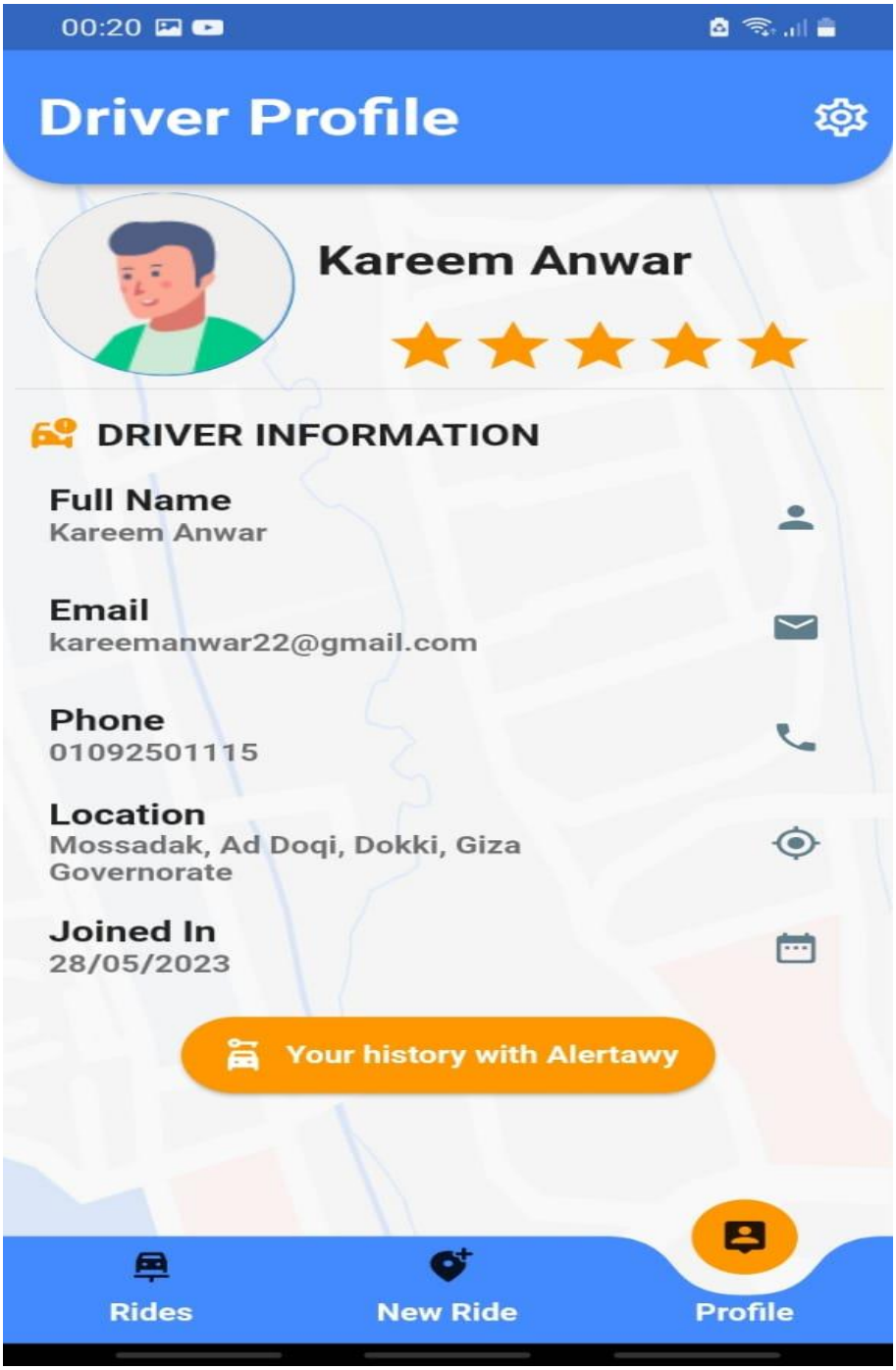
The user enters email & password and has an option (Remember me) to save email & password locally at mobile database, so the user doesn't need to enter email & password every time login to the application.

Then the user clicks on sign in the email & password verified by firebase authentication.



4.3.2 View Profile

Show the driver's profile information, including their full name, email, phone number, last location, and date of joining. (At the first enter to the application the rating default with 2.5)



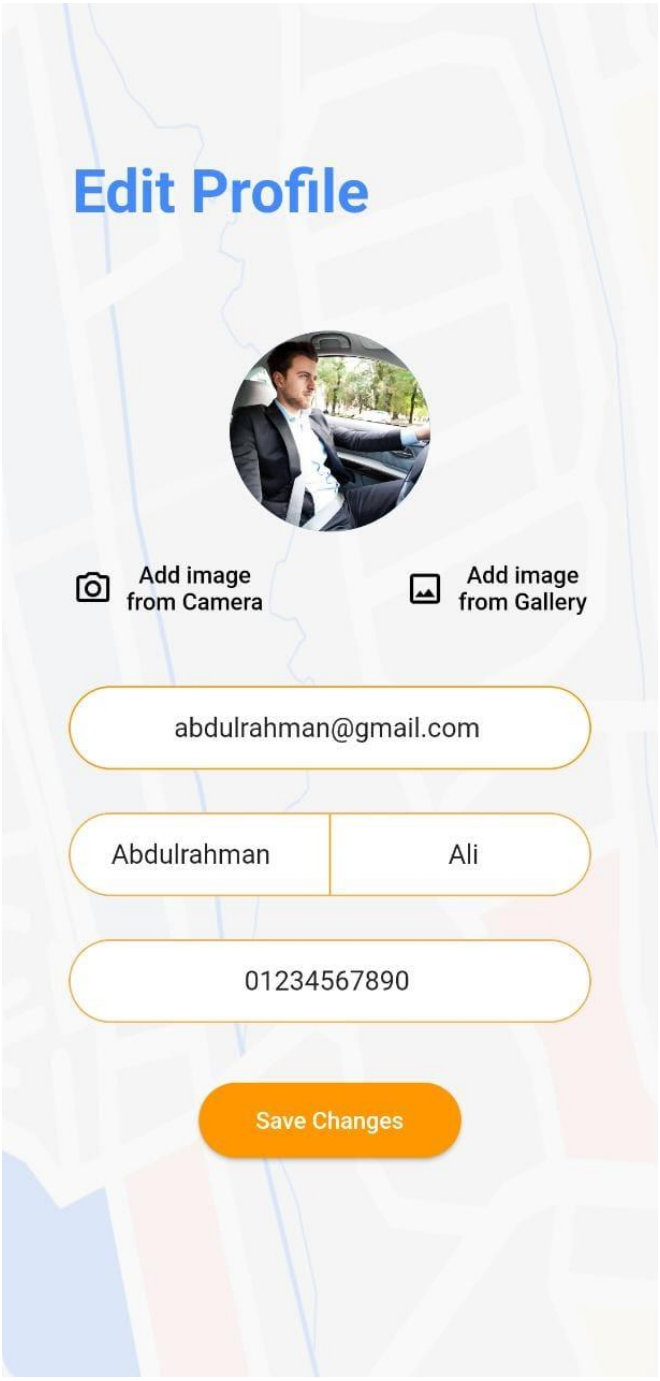
4.2.2.1 View History Report for Overall Rides Performance

At the profile page we have a button that called “Your history with Alertawy” that transfer to another page that showing the user history report of all rides that consists of phone distraction, drowsiness, road distraction, safe driving, estimated / total time, estimated / total distance, maximum / average speed & overall rating.

4.3.3 Settings

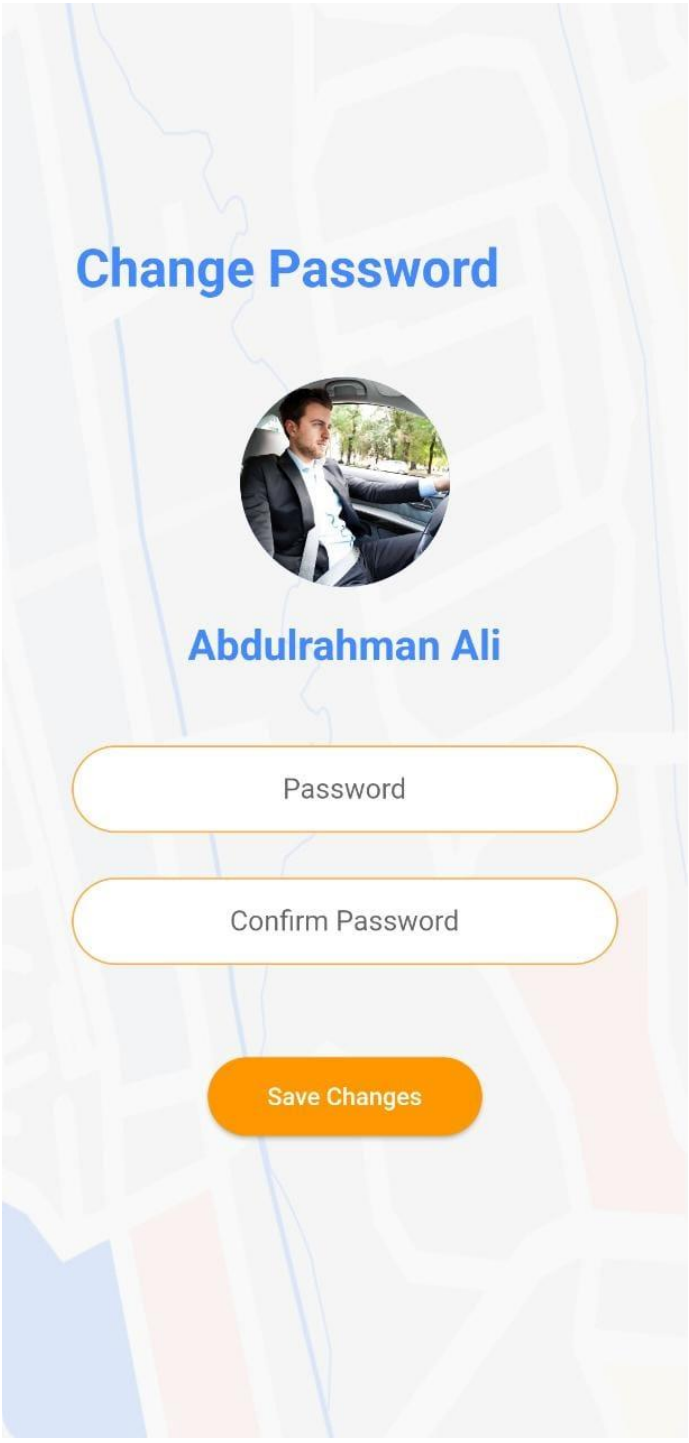
4.2.3.1 Edit Information

- Enter the new information like email, image, first name, last name & EGY phone number, then these new information change in firebase firestore also email change in firebase authentication.



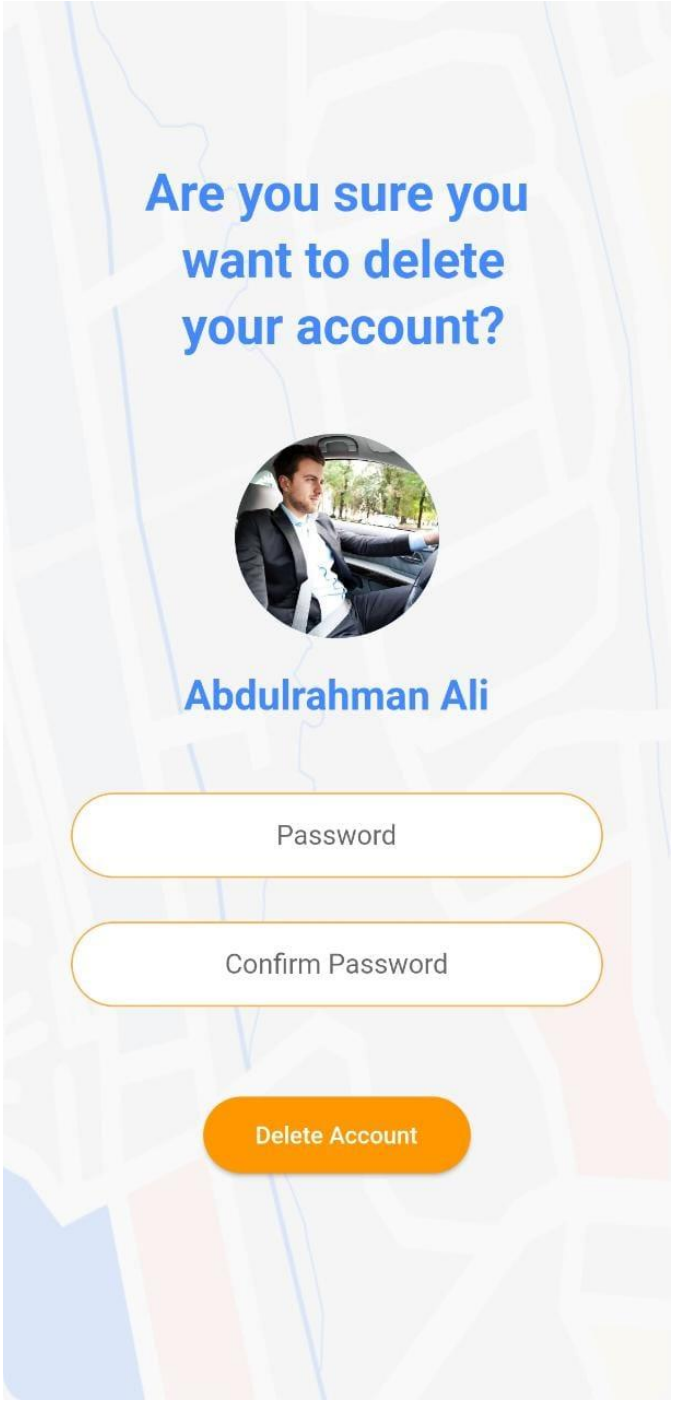
4.2.3.2 Change Password

Enter the new password and confirm password. Then, a validation operation will occur. If the password is true, it will be changed to Firebase Authentication and Firebase Firestore.



4.2.3.3 Delete Account

Enter the new password and confirm password. Then, a validation operation will occur. If the password is true, this account information will be deleted from Firebase Authentication, Firebase Firestore & Firebase Storage.



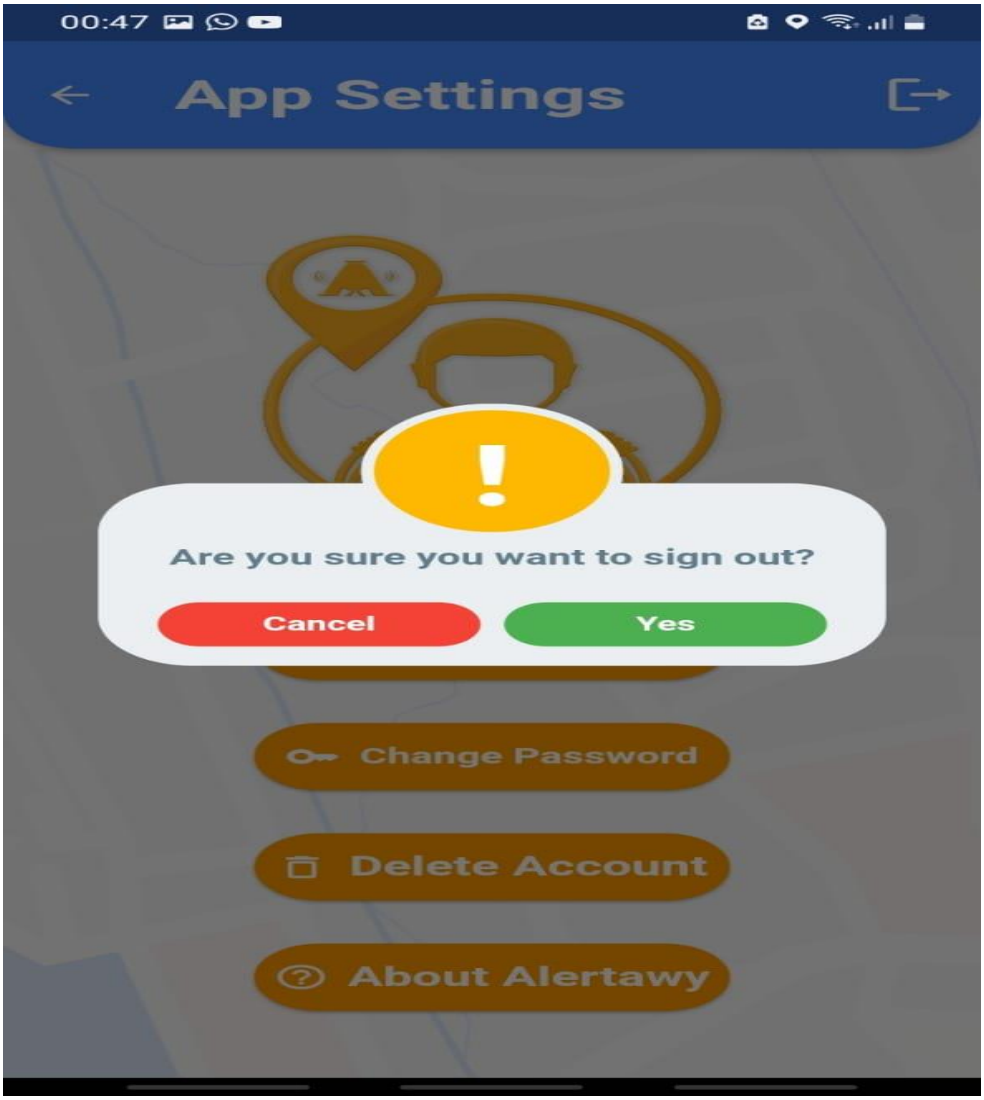
4.2.3.4 About Alertawy

Look at introduction section 4.4.1.1) Introduction.

4.2.3.5 Logout

When the user clicks on the sign-out icon, a pop-up will appear. If the user clicks 'Yes', the application will sign out the user, transfer the user to the start page and save his last time he logged out in Firebase Firestore.

Note: If the user checks the 'Remember Me' checkbox on the sign-in page, their data will be deleted from the local database on the mobile device.

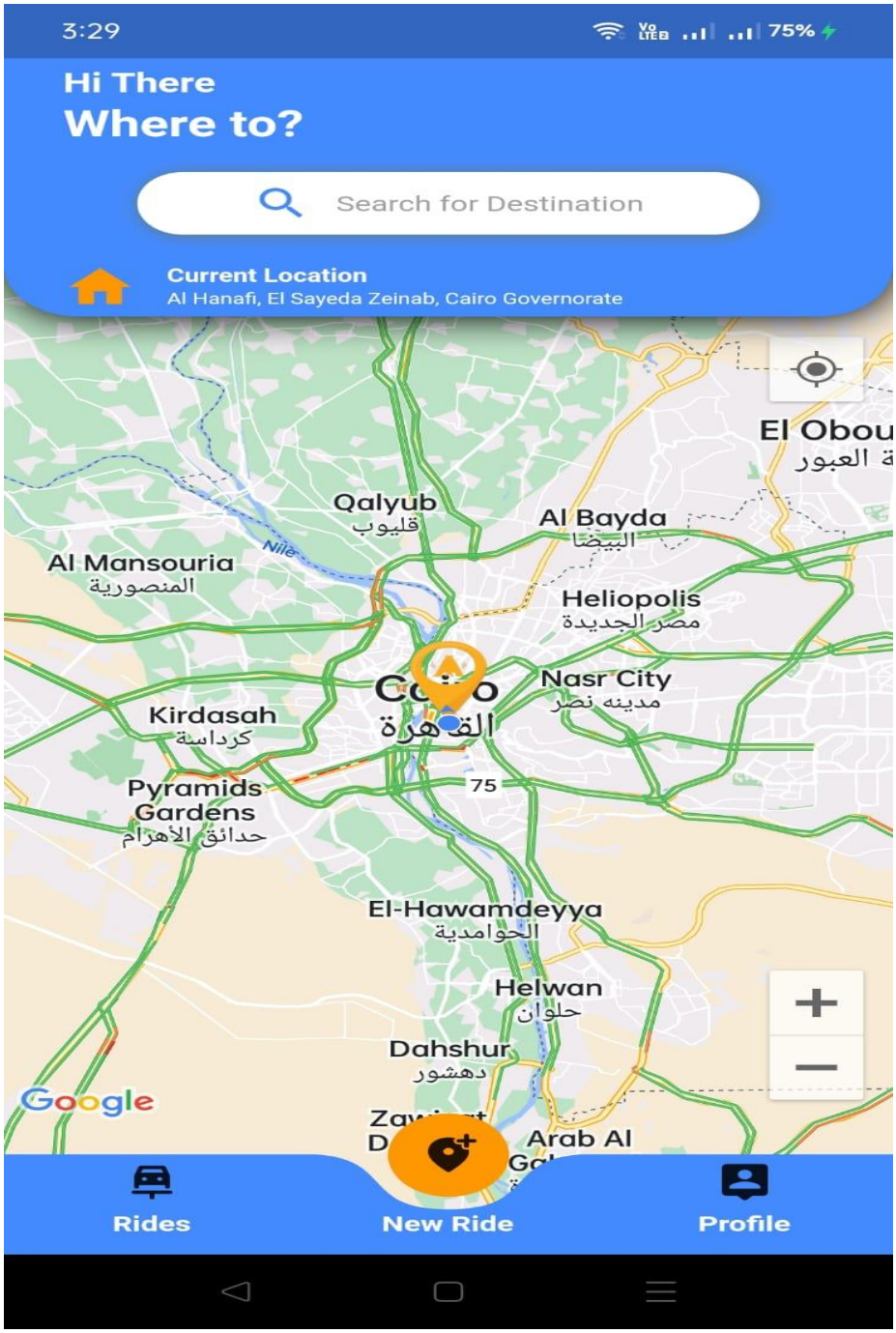


4.3.2.2) Start New Rides Page

Drivers should have the ability to start a new ride in the system.

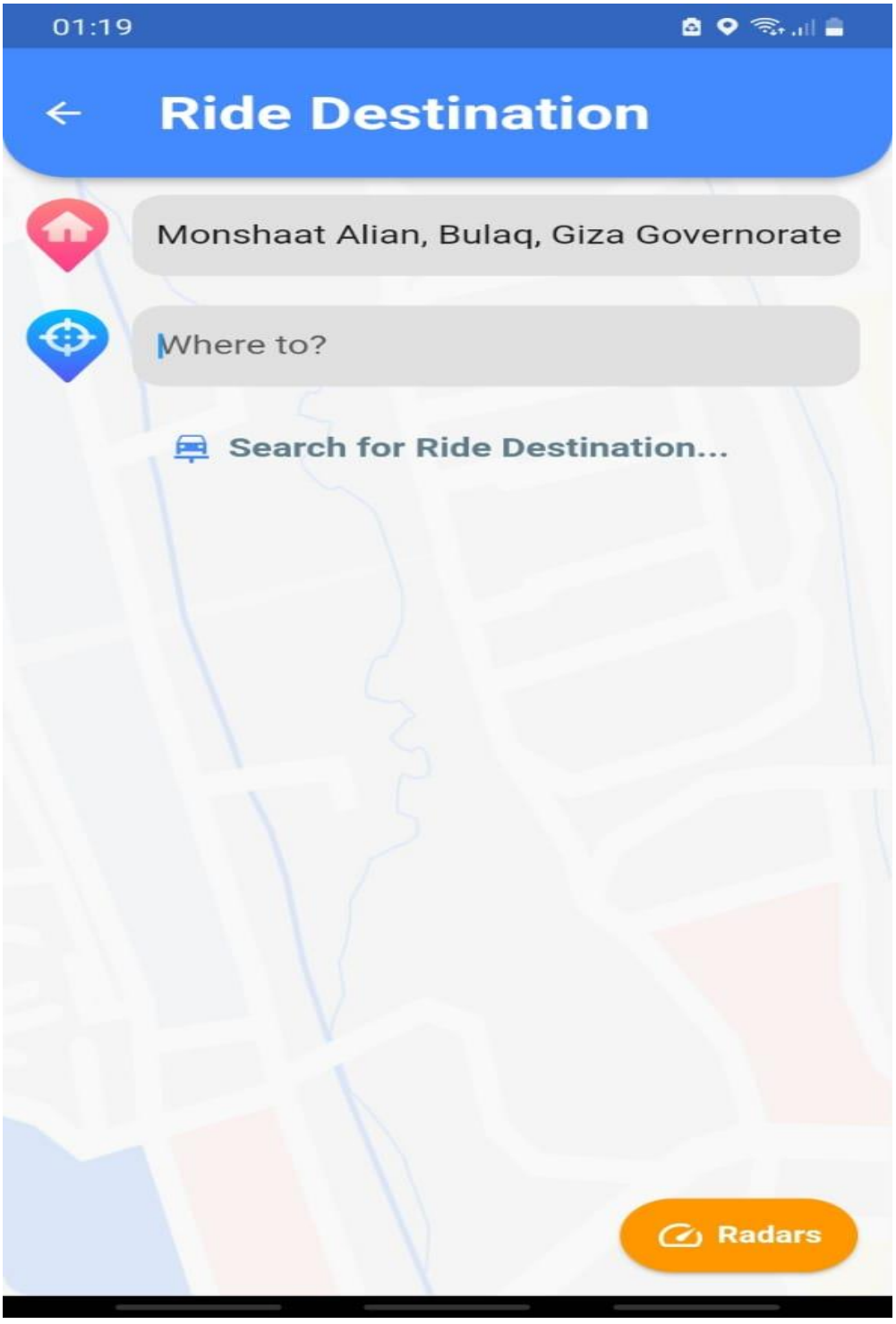
4.3.4 View Current Location on Maps

Drivers should be able to view their current location on maps within the system.



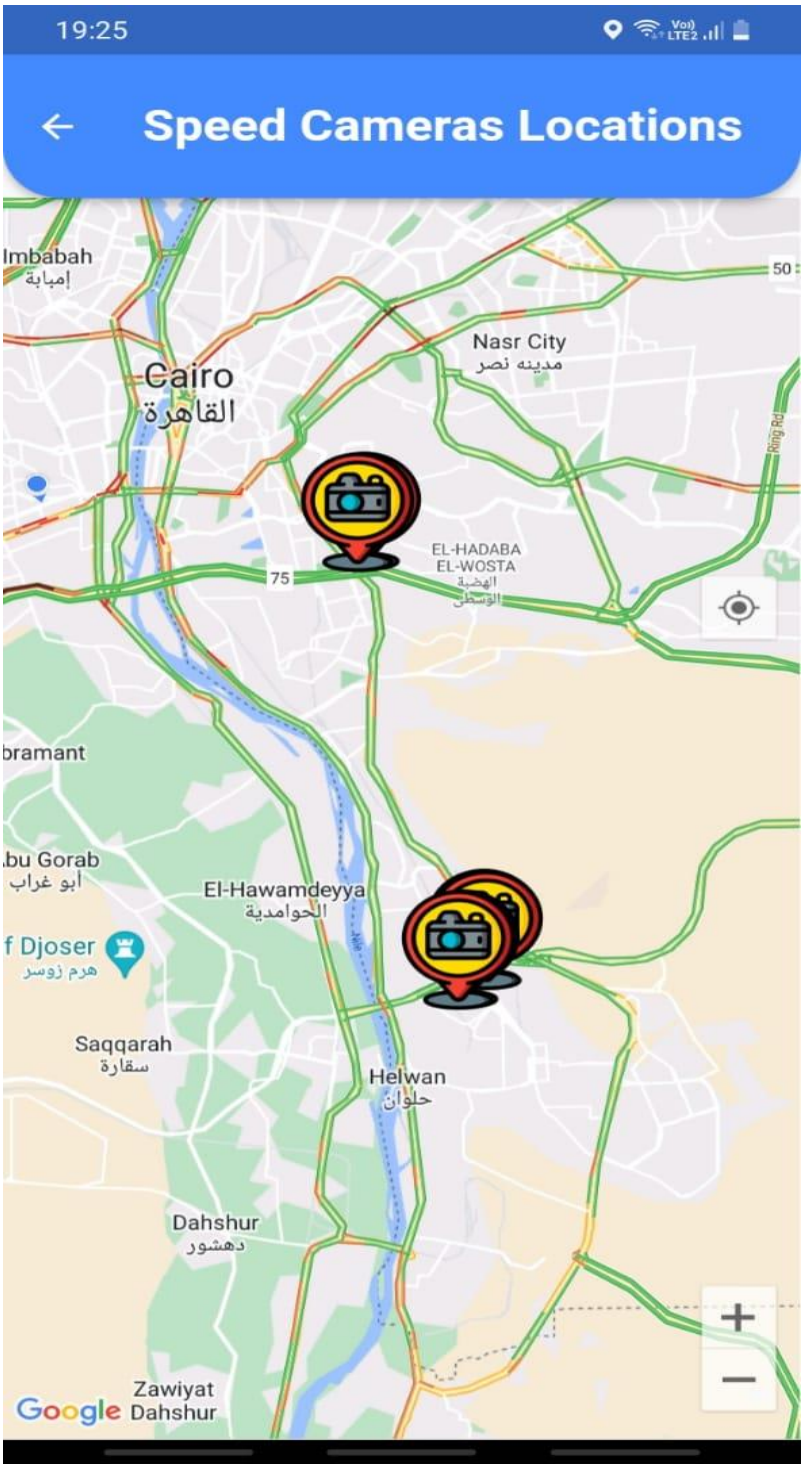
4.3.5 Search for Destination

Drivers should be able to search for a destination on maps and set it as the destination for their ride.



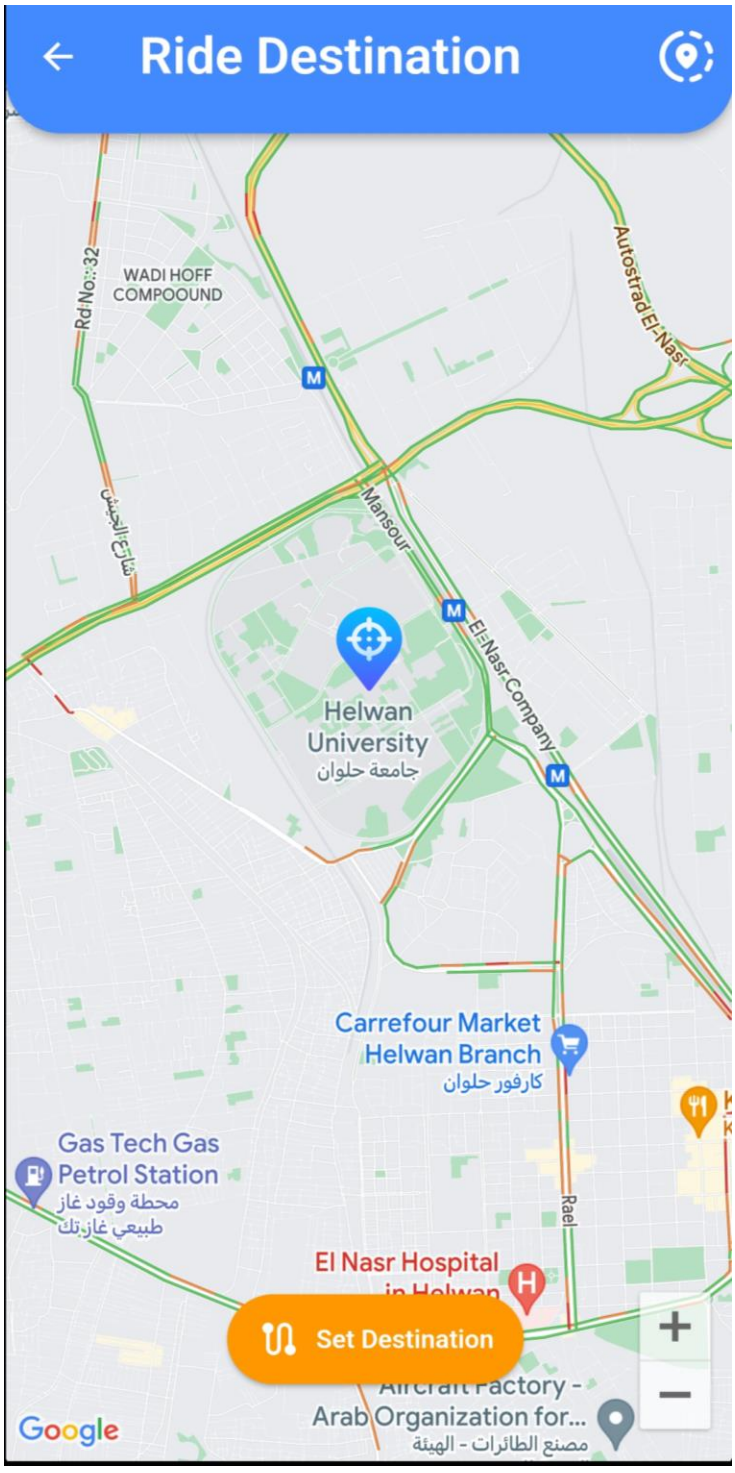
4.2.6 View All radars saved on the System

Showing all the radars that exist on the system through the map.



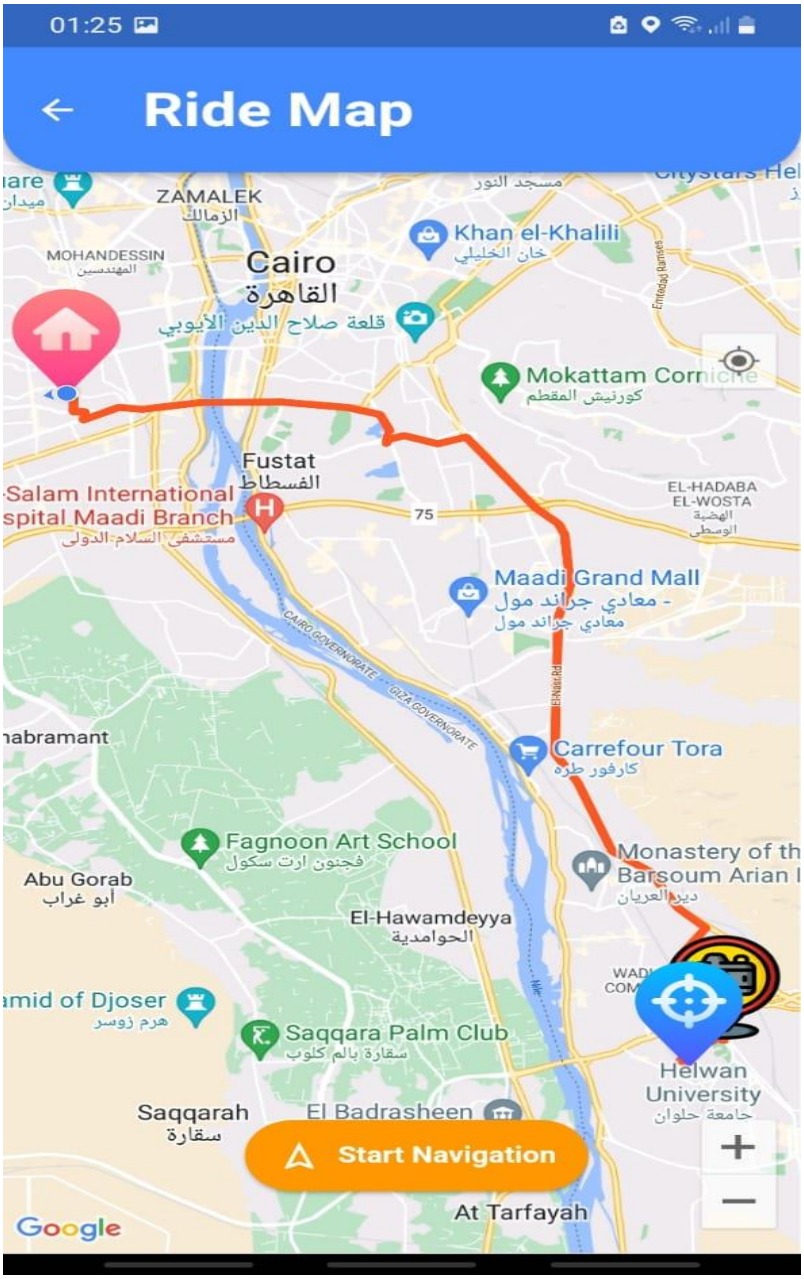
4.2.6 Set Destination on Maps

Select Specific Location on Maps or Select Searched Place Location.



4.3.7 Display Optimal Route and Radar Locations on Maps

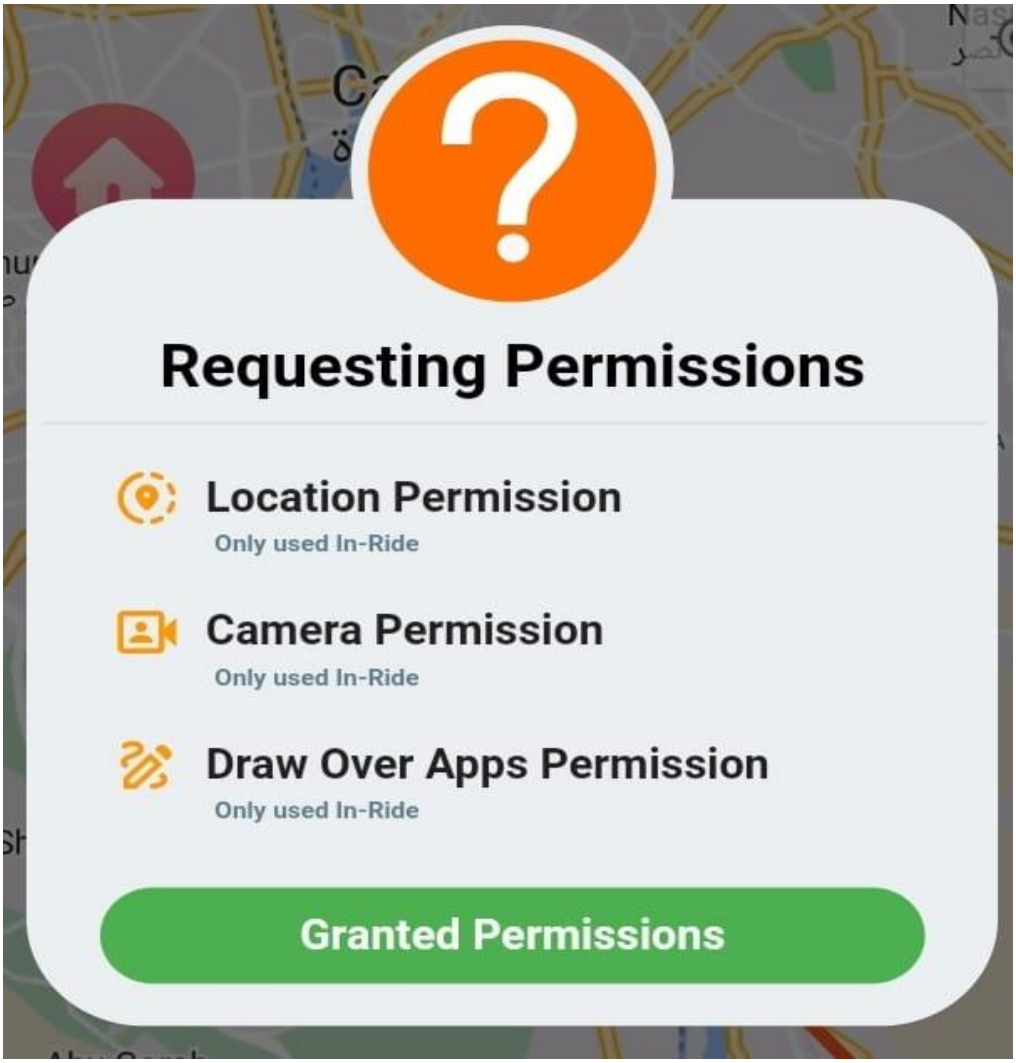
Drivers should have the capability to view the optimal route on maps within the system, along with the locations of radars they will encounter along the route.



4.3.8 Start Navigation

4.2.8.1 Permissions Request

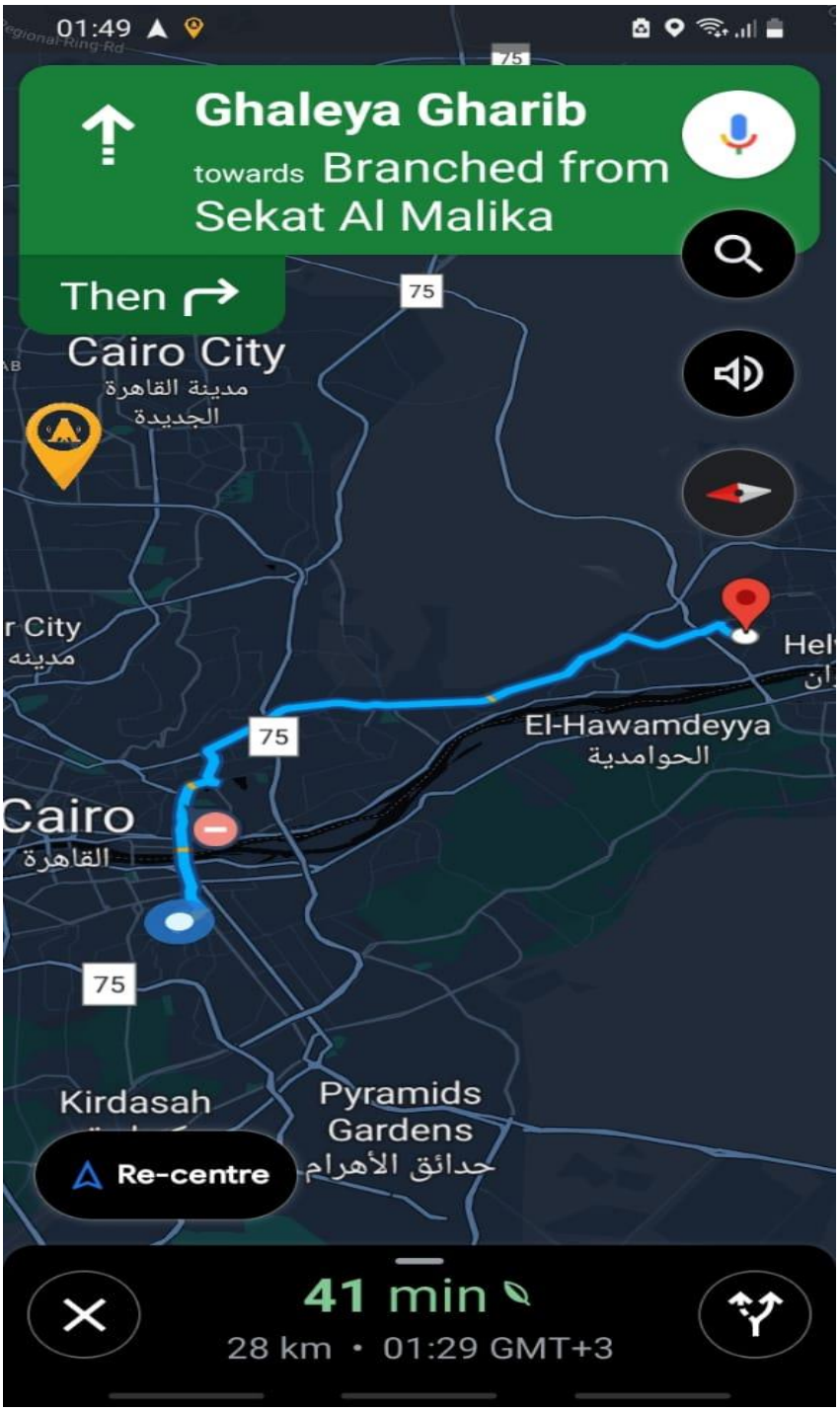
Request ride permissions if not granted.



4.2.8.2 Redirect to Google Maps for Navigation

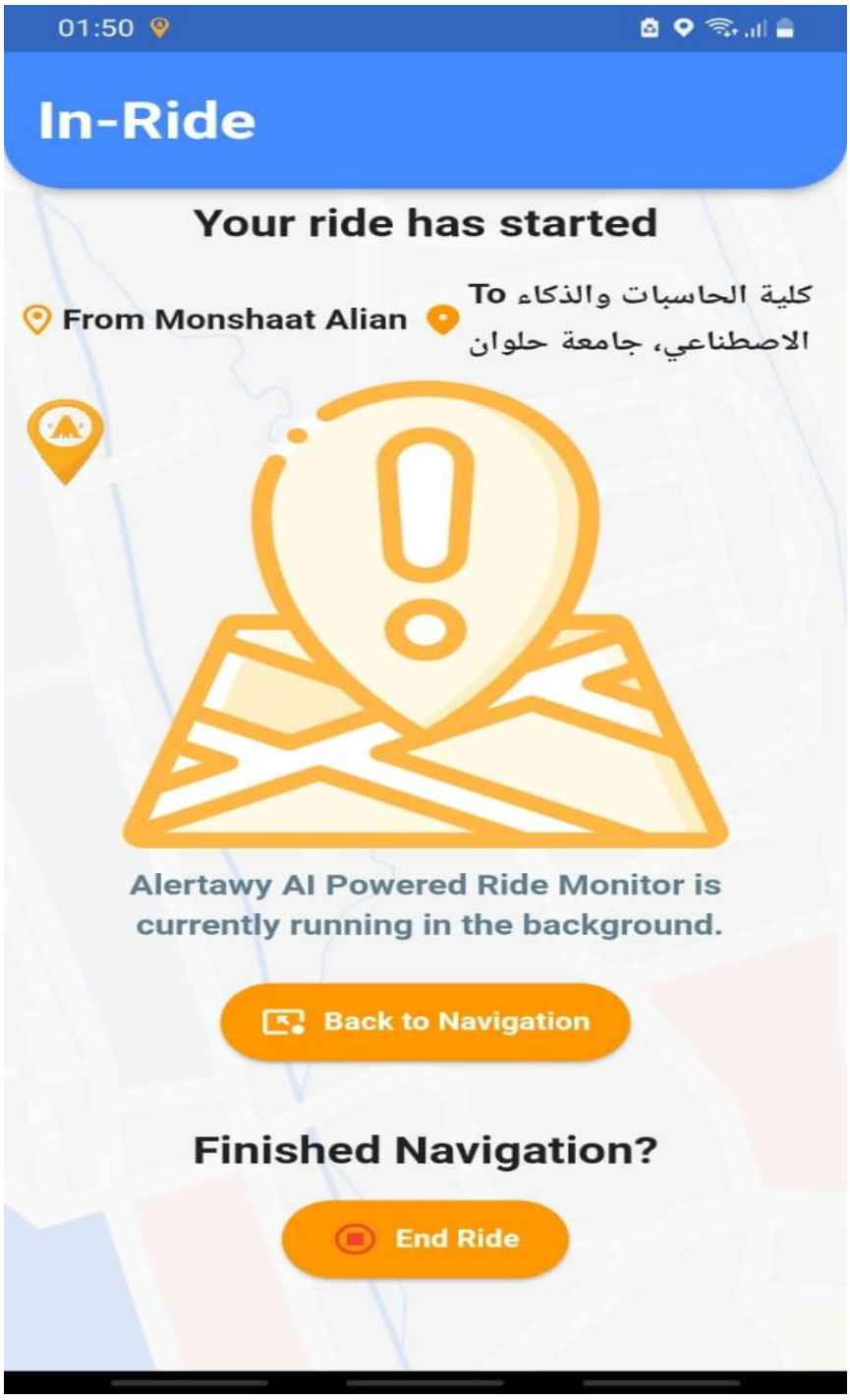
After the application obtains the necessary permissions, it will transfer the user to navigate through google maps.

The monitoring system will run at the background of the application.



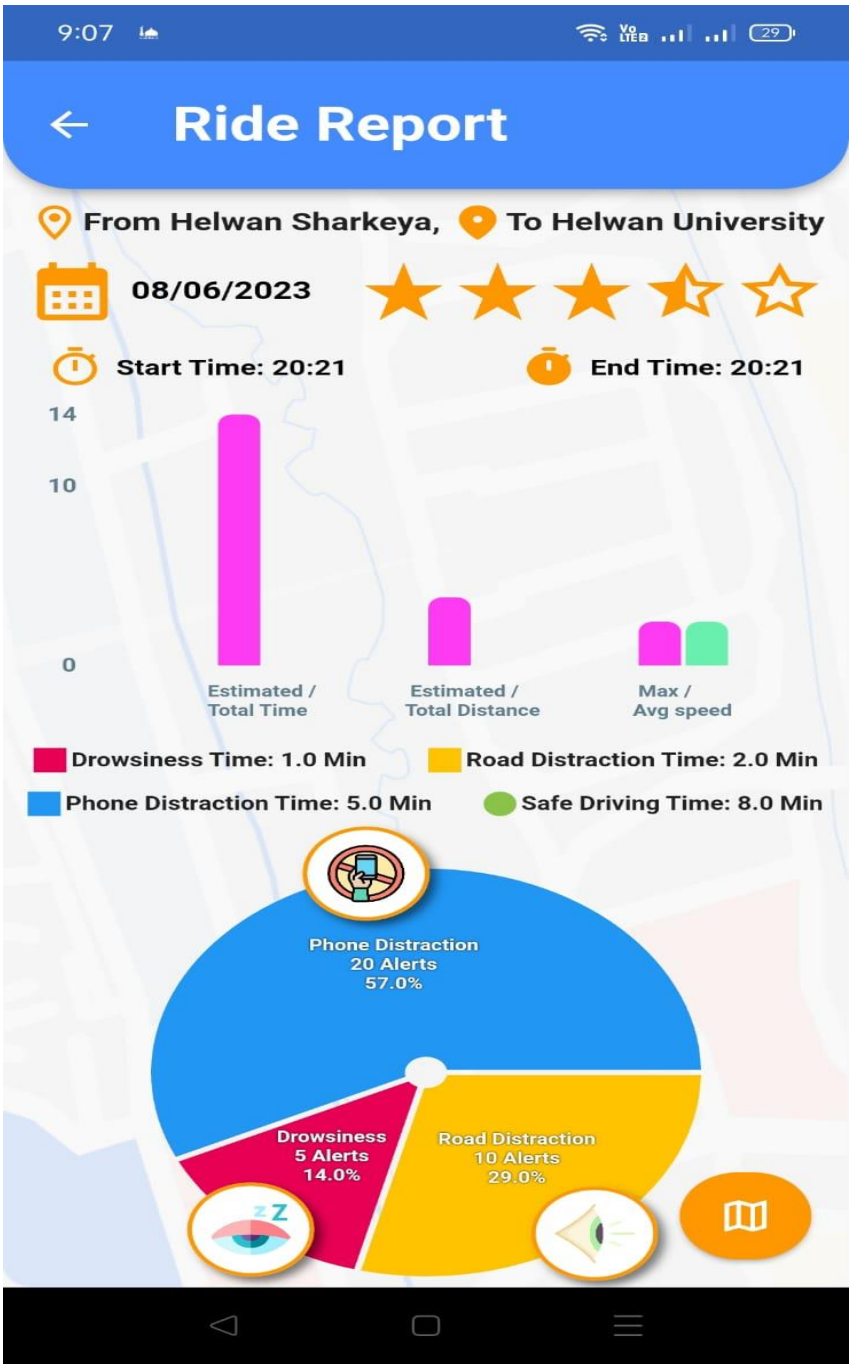
4.2.8.3 End ride

When the user ends the ride the information of the ride will be saved in Firebase Firestore.



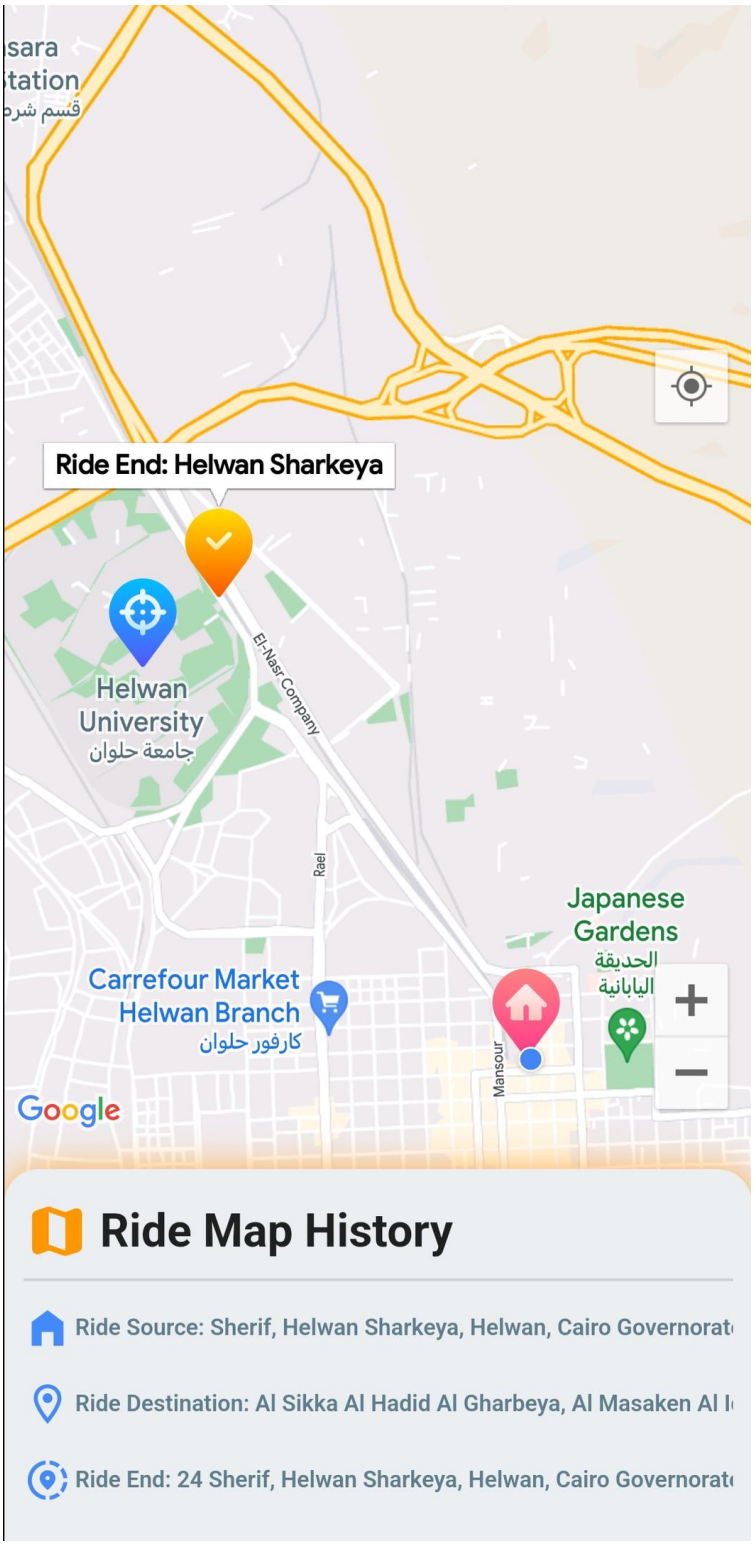
4.2.8.4 Ended Ride Report

The report of the ended ride includes estimated / total time, estimated / total distance, maximum and average speed, drowsiness time, road distraction time, phone distraction time, safe time, start and end time.



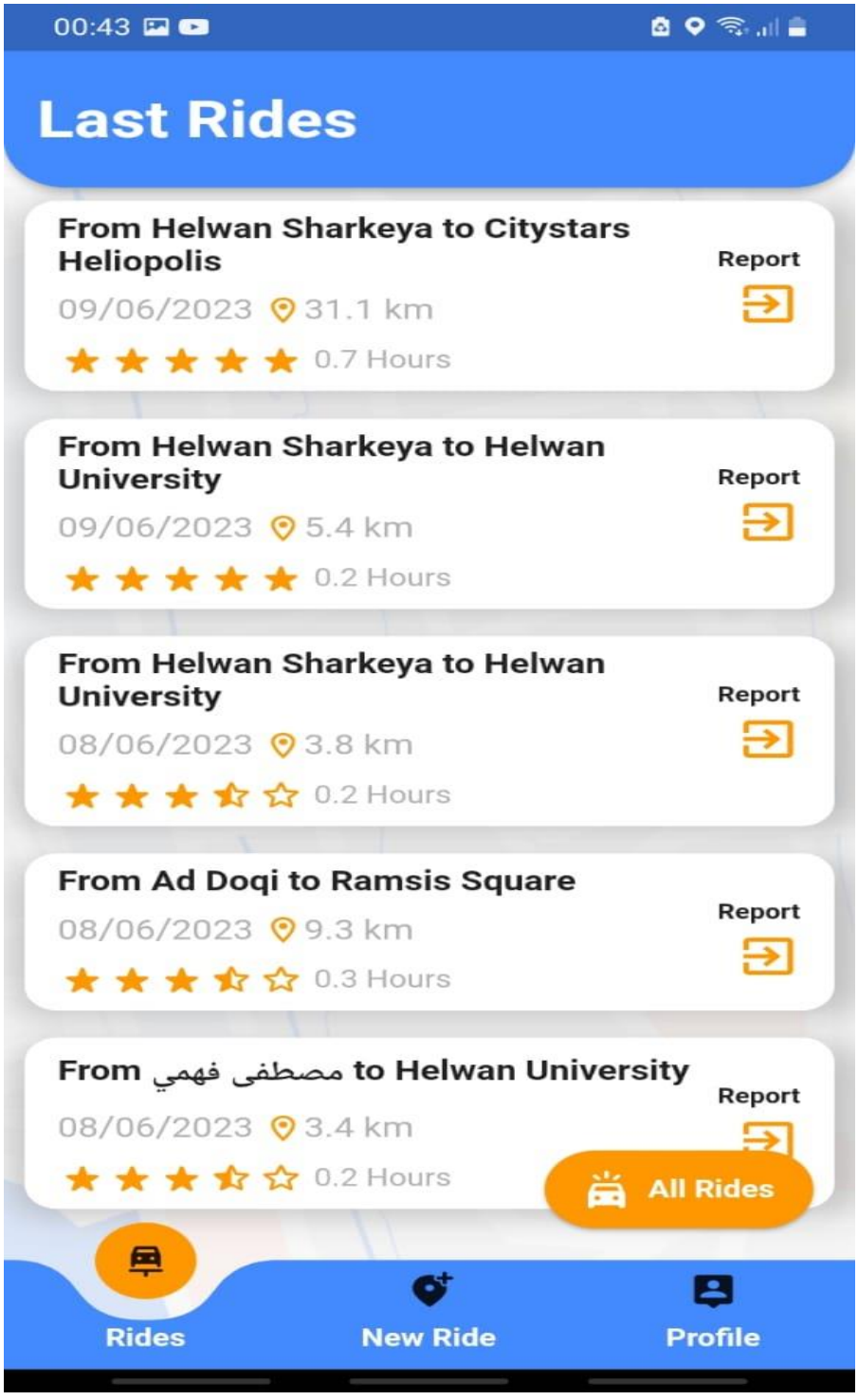
4.2.8.4.1 Ride Map History

Showing the ride source, ride destination & ride end places on the map .



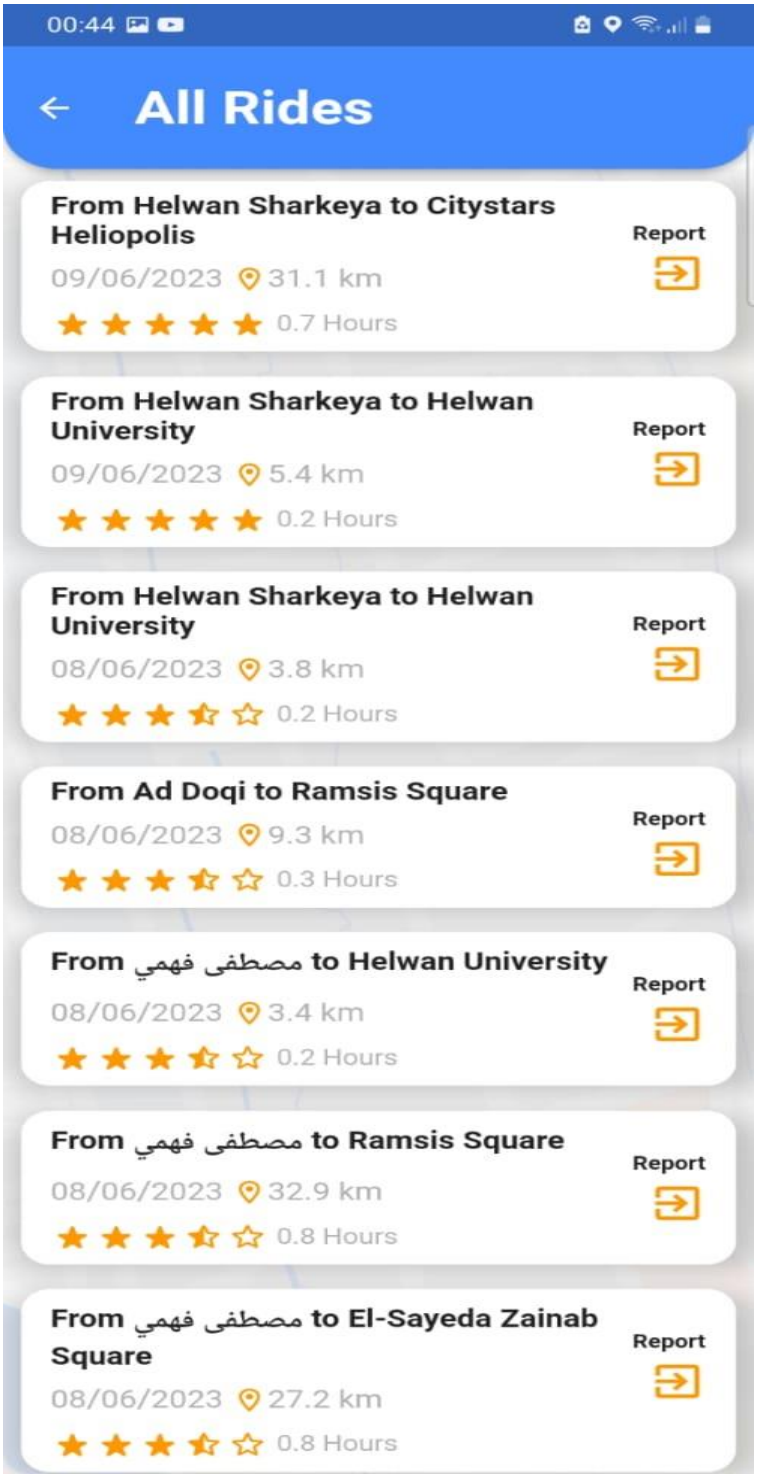
4.3.9 View Past Rides

Showing the last five rides performed by the user from newest to oldest.



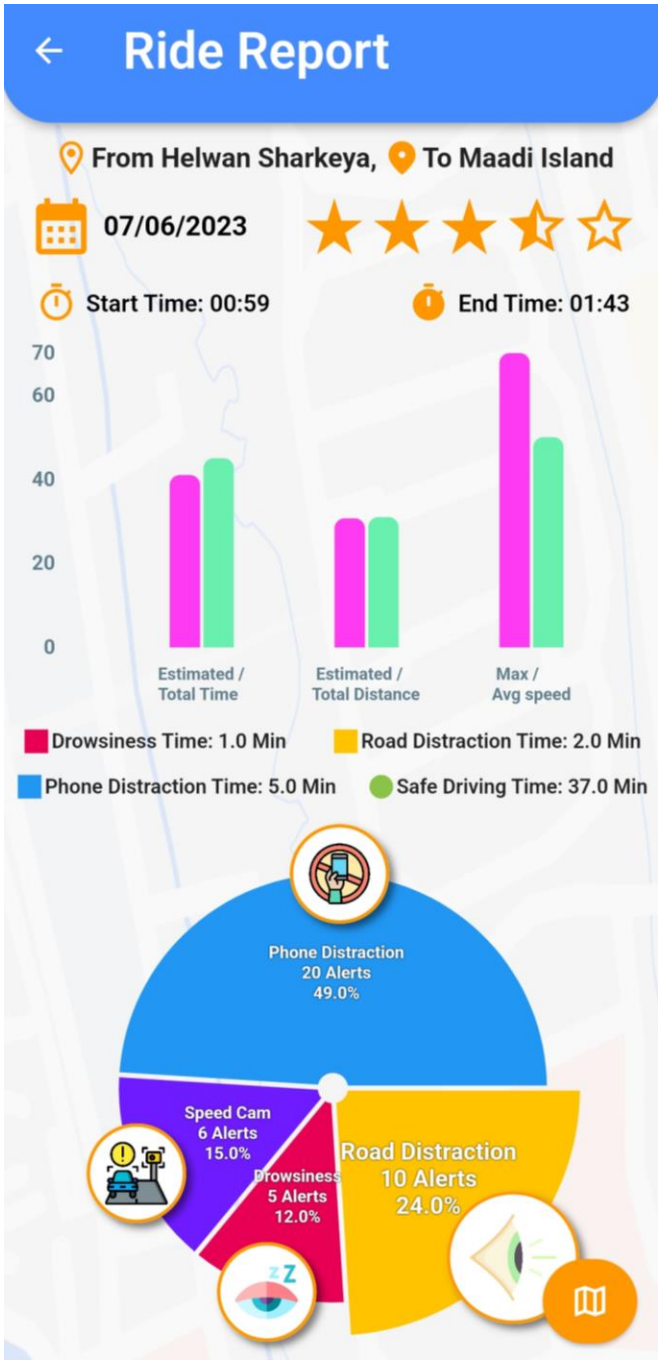
4.3.10 View All Rides

There is a button at the rides page that shows all rides performed by the user from newest to oldest.

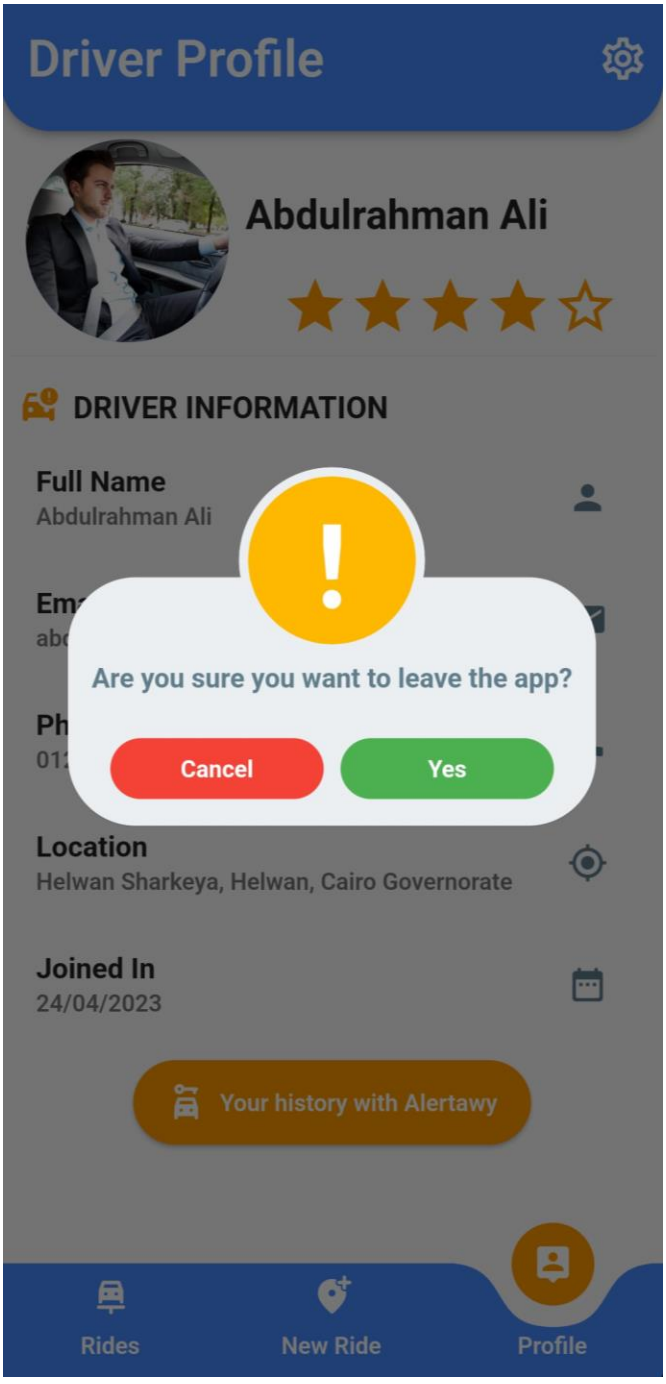


4.3.11 View Past Rides Reports

There is a report button at every ride when click on it showing you the report information of this ride.



4.3.11 Exit Application



4.4 Administration Web application

Our administration web application follows the same principles and technologies as our mobile application. Built using the Flutter framework for web development, it benefits from the platform's extensive features and advantages in creating clean and well-organized code architectures while adhering to design patterns.

Just like in our mobile app, we have placed significant emphasis on adhering to SOLID Principles, ensuring that our code is maintainable, scalable, and follows best practices. Additionally, we have implemented the Singleton Design Pattern for our Authentication Service, which helps in managing admin and company roles authentication efficiently.

Moreover, we have chosen to adopt the MVC pattern for handling interactions with Cloud Firestore and managing various operations within our web app. This pattern provides a clear separation of concerns, with models responsible for mapping data from Firestore's JSON documents, controllers handling data manipulation and retrieval, and views rendering the user interface with data obtained through controllers.

By employing these design patterns and leveraging Flutter's capabilities for web development, our administration web application achieves a well-structured and robust foundation, ensuring a smooth user experience and facilitating easy maintenance and future enhancements.

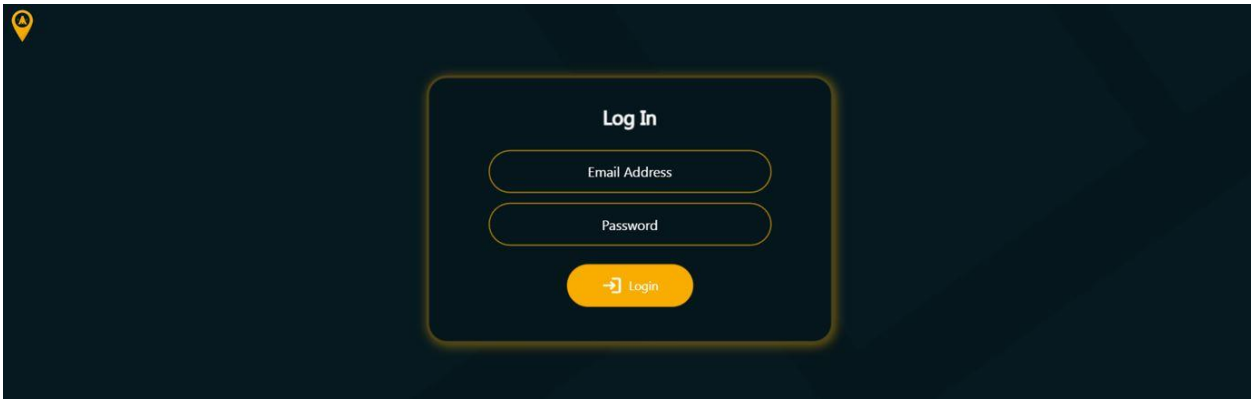
In addition to the design principles and technologies, we have also taken the necessary steps to host our administration web application. Leveraging the Firebase platform, specifically Firebase Hosting, we have deployed our web application to ensure its availability and accessibility to users. By utilizing Firebase Hosting, we can take advantage of its scalability, security, and ease of deployment, allowing us to focus on delivering a seamless and reliable web experience to our administrators.

4.4.1 Company Representatives Role

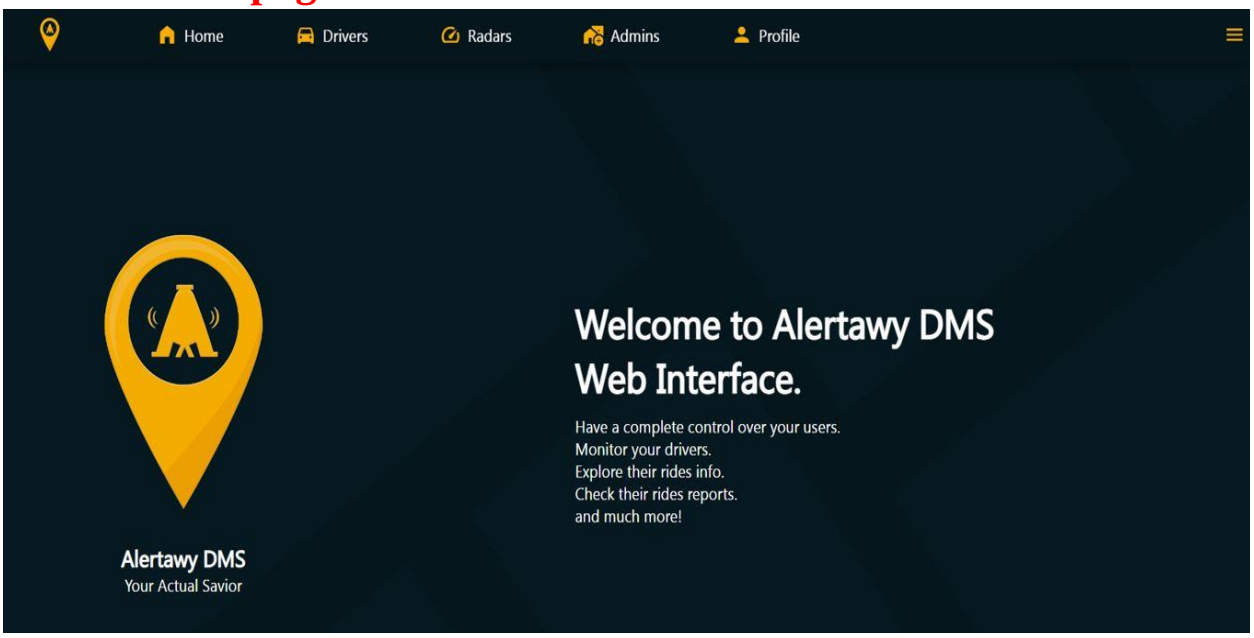
4.4.1.1 Start Page

4.4.1.1.1 Login

The company signs in with Company email & password and verify them through Firebase Firestore.



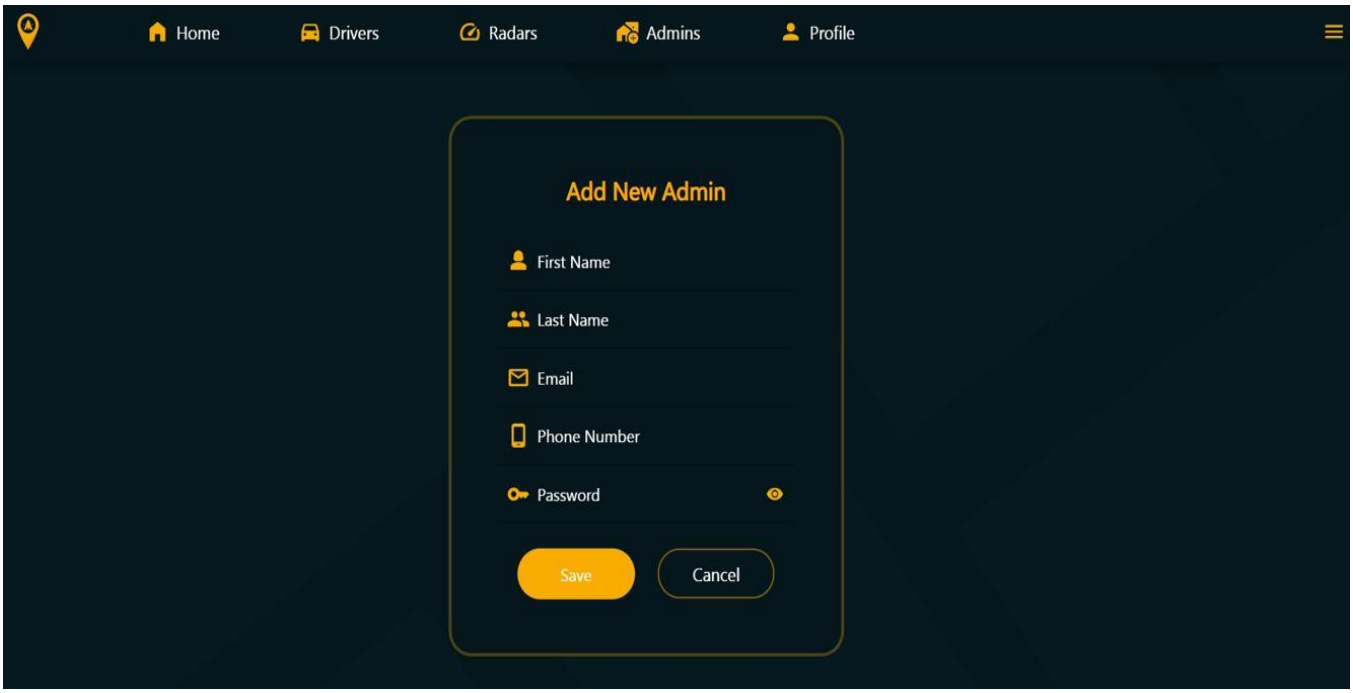
4.4.1.2 Home page



4.4.1.2.1) Add Company Admins

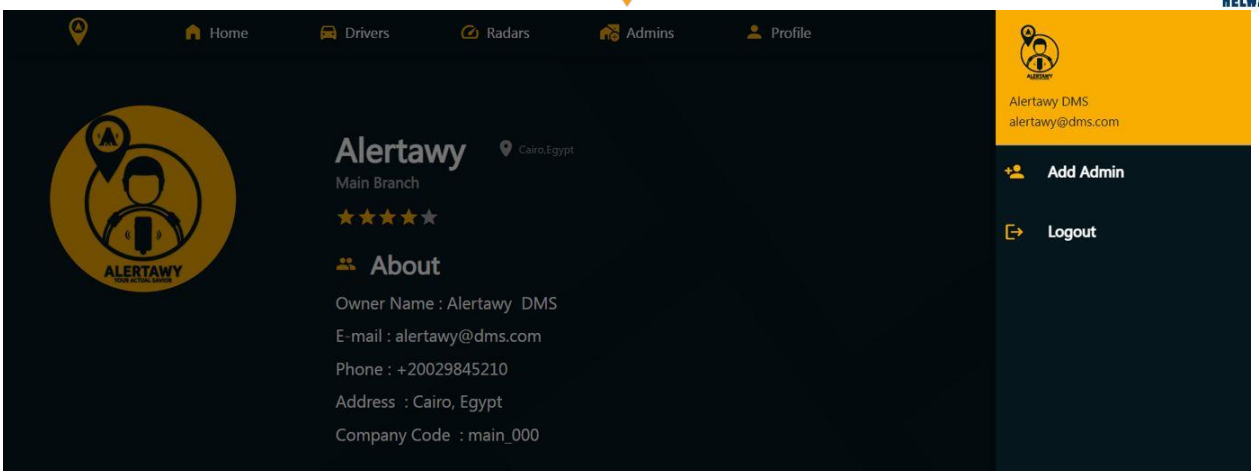
Add Admin information like First name, Last name, Email, EGY phone number & password, then validate this information from some safety functions if something doesn't satisfy the application rules the

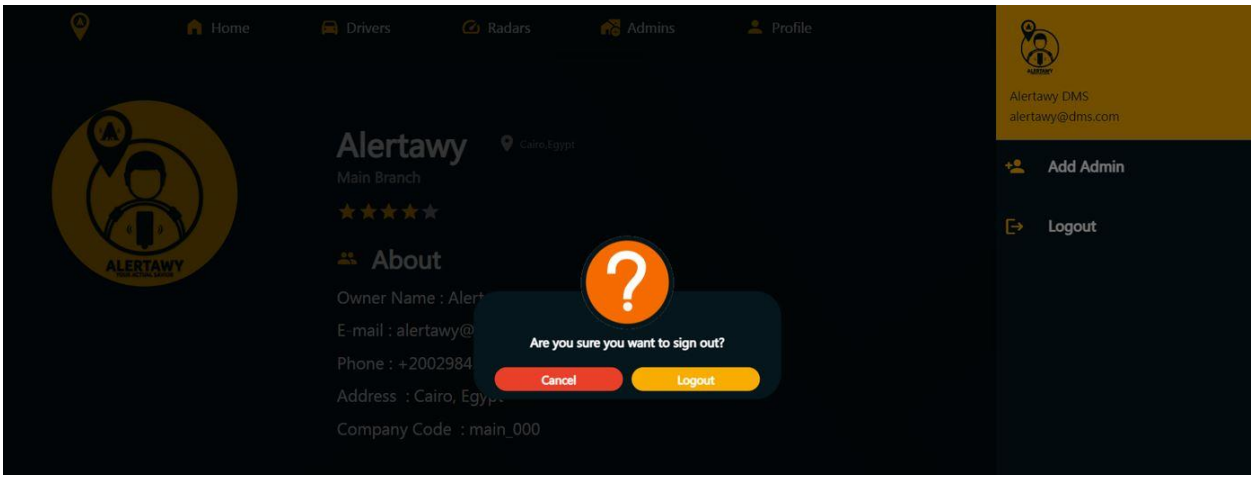
application will ask for correct information, else this information saved in firebase firestore.



4.4.1.2.2) Logout

sign out the company from the system with saving his last time he logged out in Firebase Firestore.

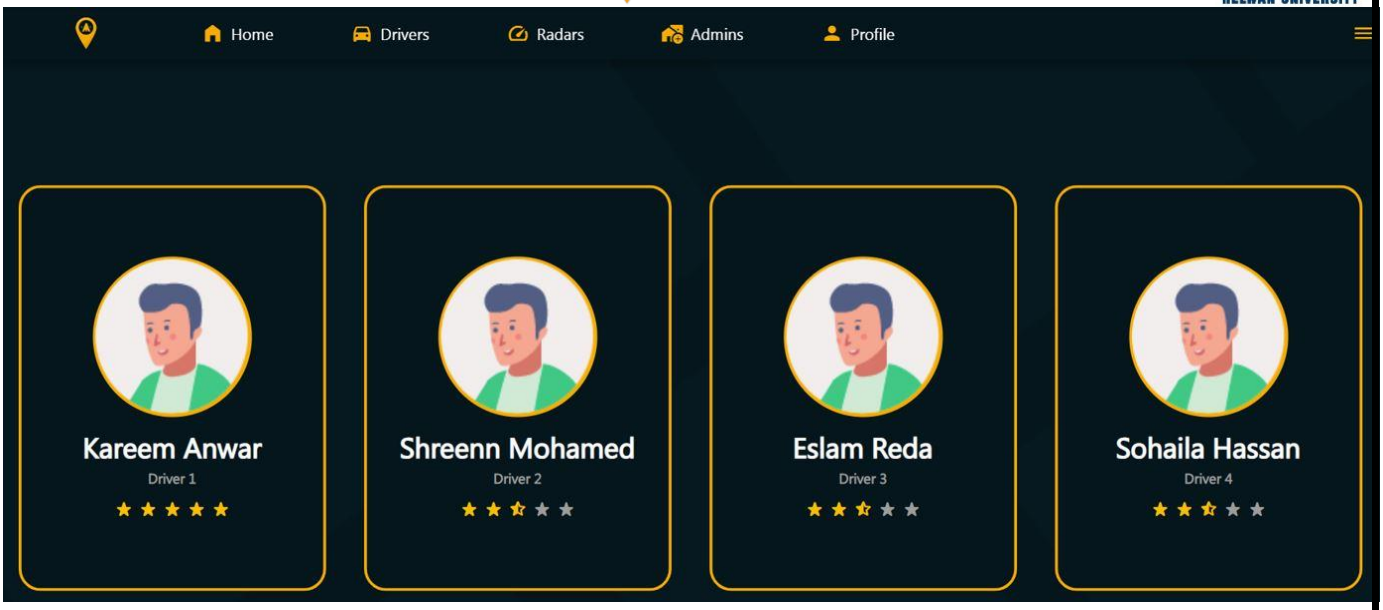




4.4.1.3 Company Driver's Page

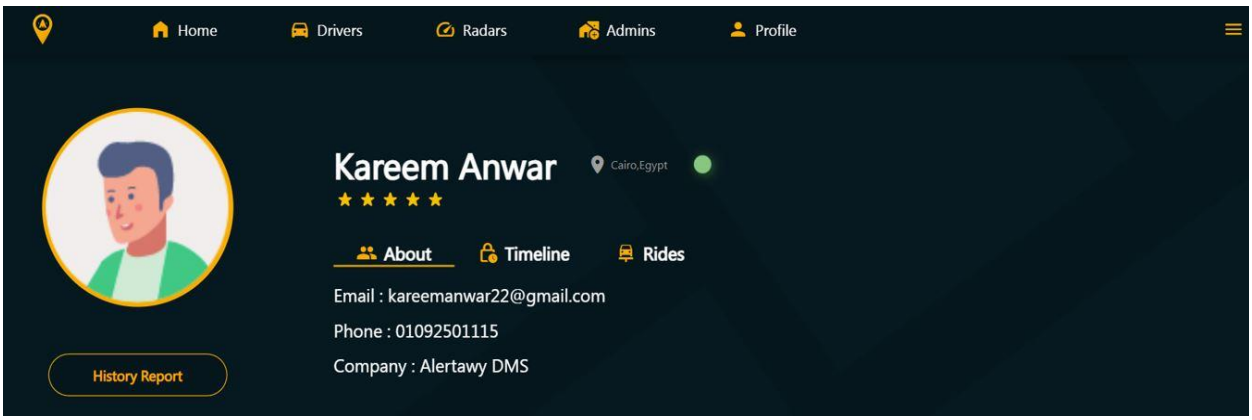
4.4.1.3.1 View Company Drivers

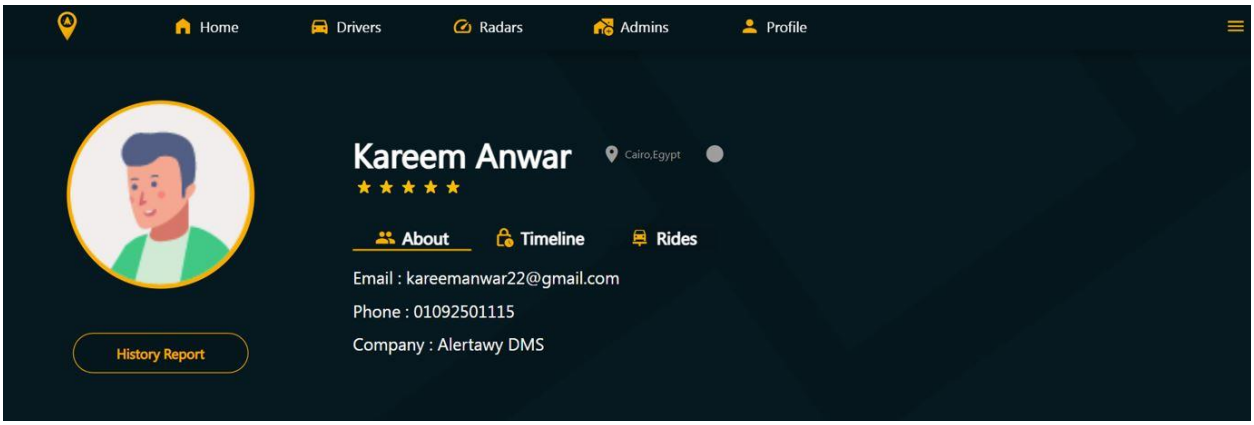
Showing list of all drivers in the system.



4.4.1.3.2) Driver profile

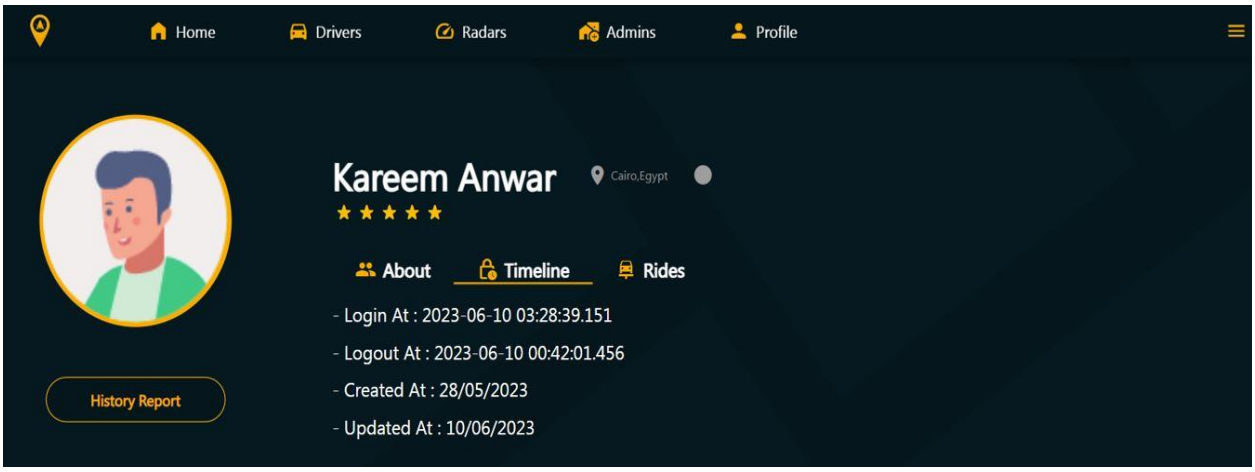
Showing driver profile in the ride or not in the ride with his information.





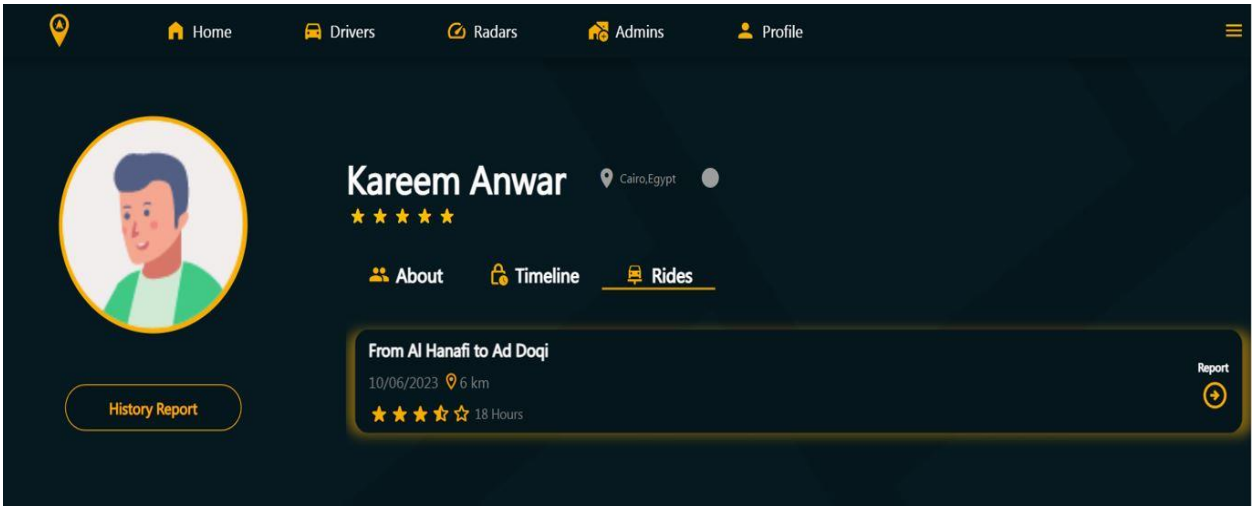
4.4.1.3.3) Driver Timeline

Showing the driver timeline since he has created his account on the system.



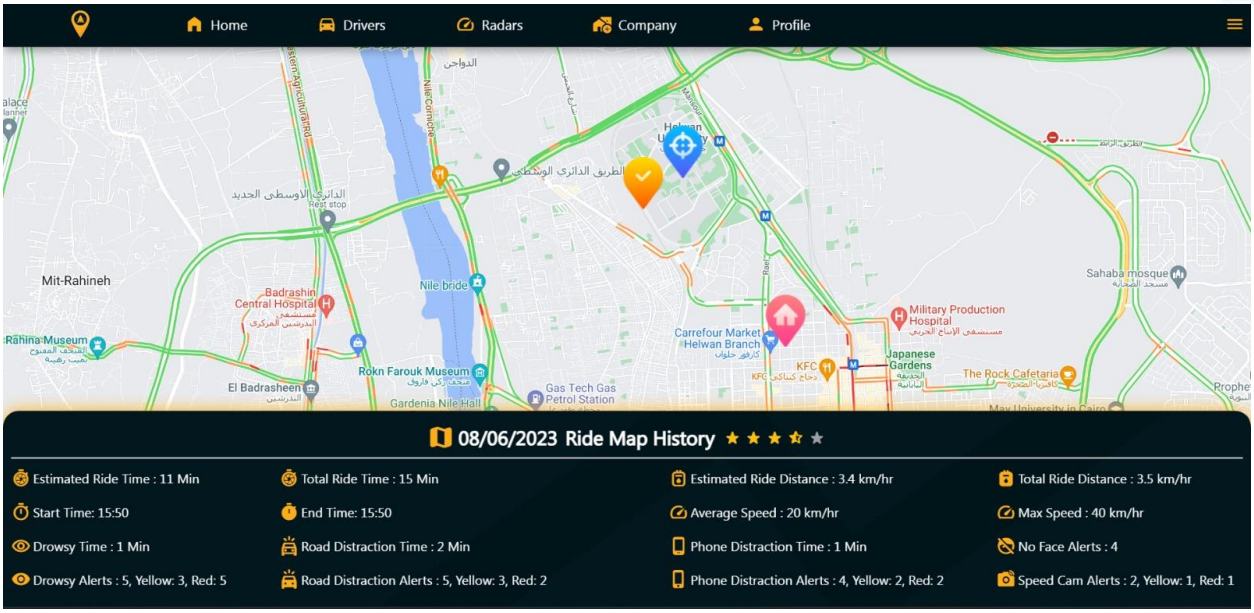
4.4.1.3.4) View Company Drivers' Past Rides

Showing the driver all rides history on the system.



4.4.1.3.5) View Company Drivers' Rides History on Map

Showing the driver history report of specific ride.

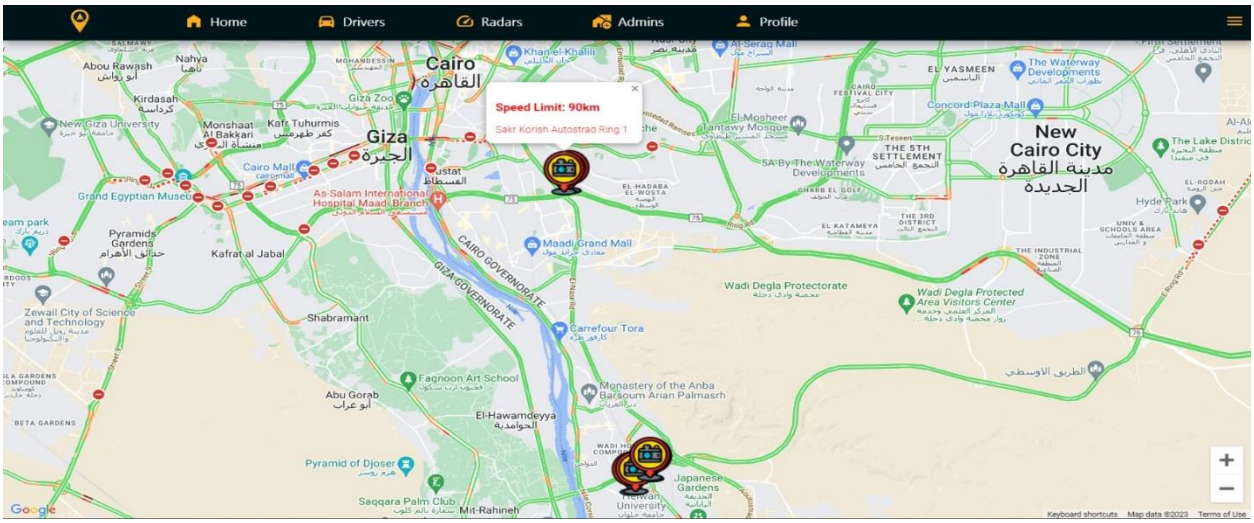


4.4.1.3.6) View Company Drivers' History Report

Showing the driver history report since he has created the account.

4.4.1.4 Saved Radars Page

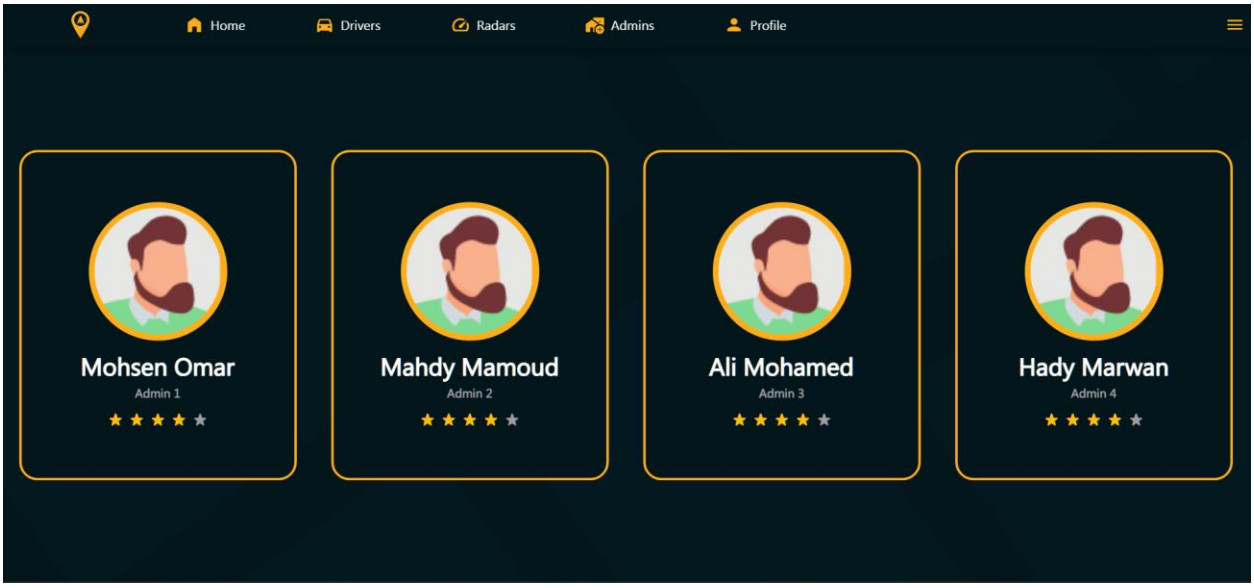
Showing all existing radars on the system on the map.



4.4.1.5 Company Admins Page

4.4.1.5.1 Company Admins Page

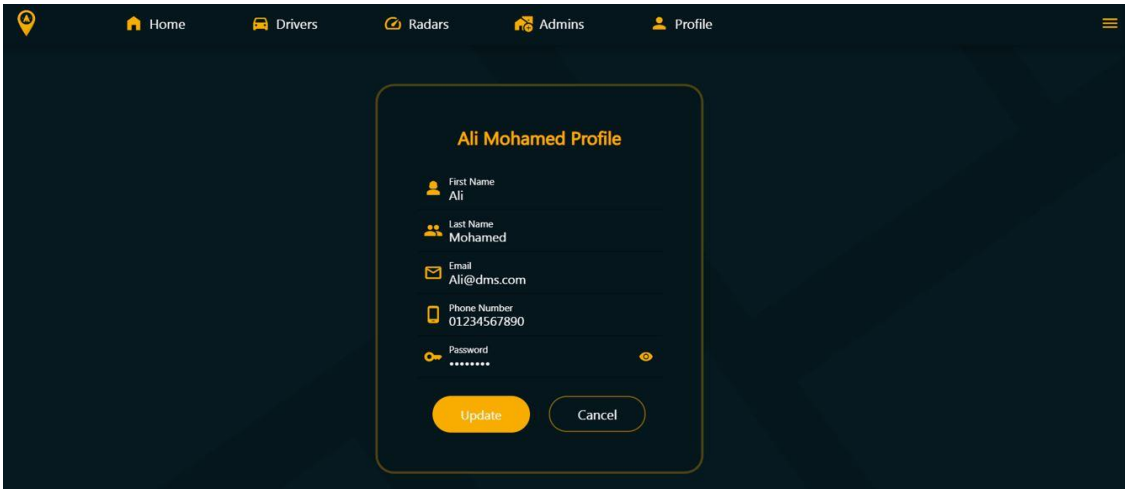
Showing list of all admins in the system.



4.4.1.5.2 Admin Profile Page

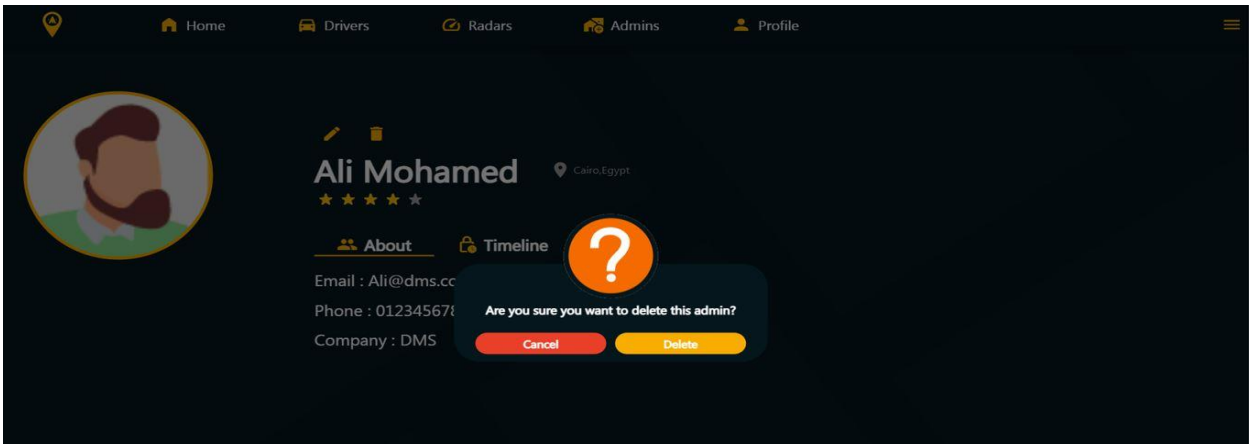
4.4.1.5.2.1 Edit Company Admins

Editing admin profile with the new information then validate this information from some safety functions if something doesn't satisfy the application rules the application will ask for correct information, else these information saved in firebase firestore.



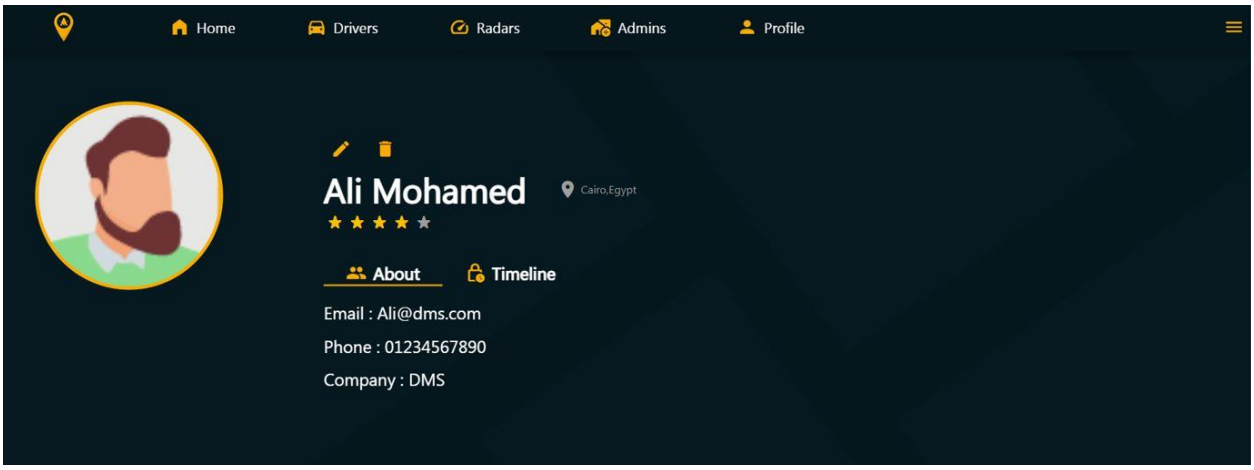
4.4.1.5.2.2) Delete Company Admins

Deleting admin profile information from the Firebase Firestore and image from Firebase Fire storage .



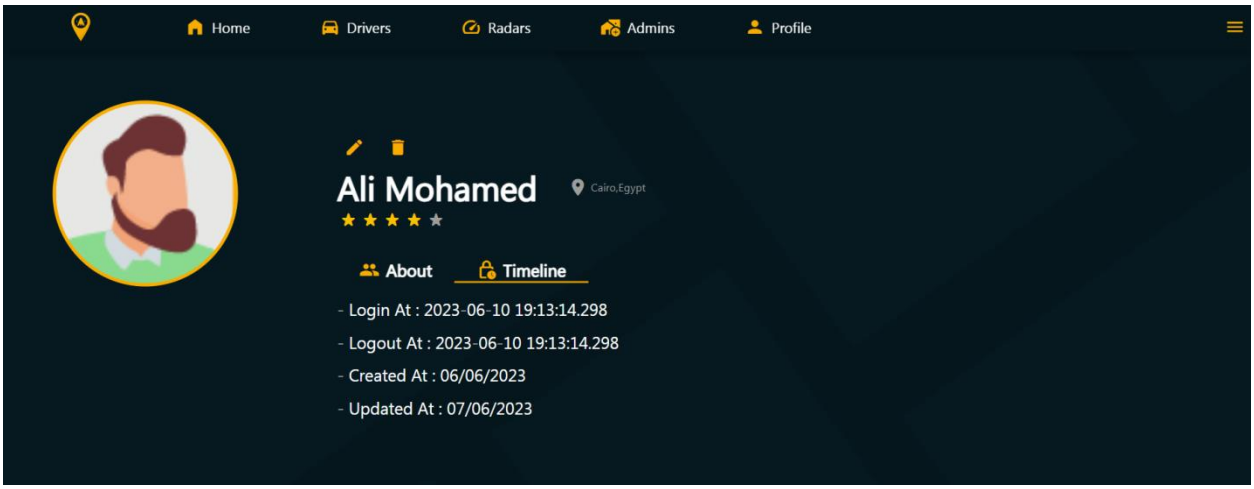
4.4.1.5.2.3) Admin Profile About

Showing admin profile with his information.



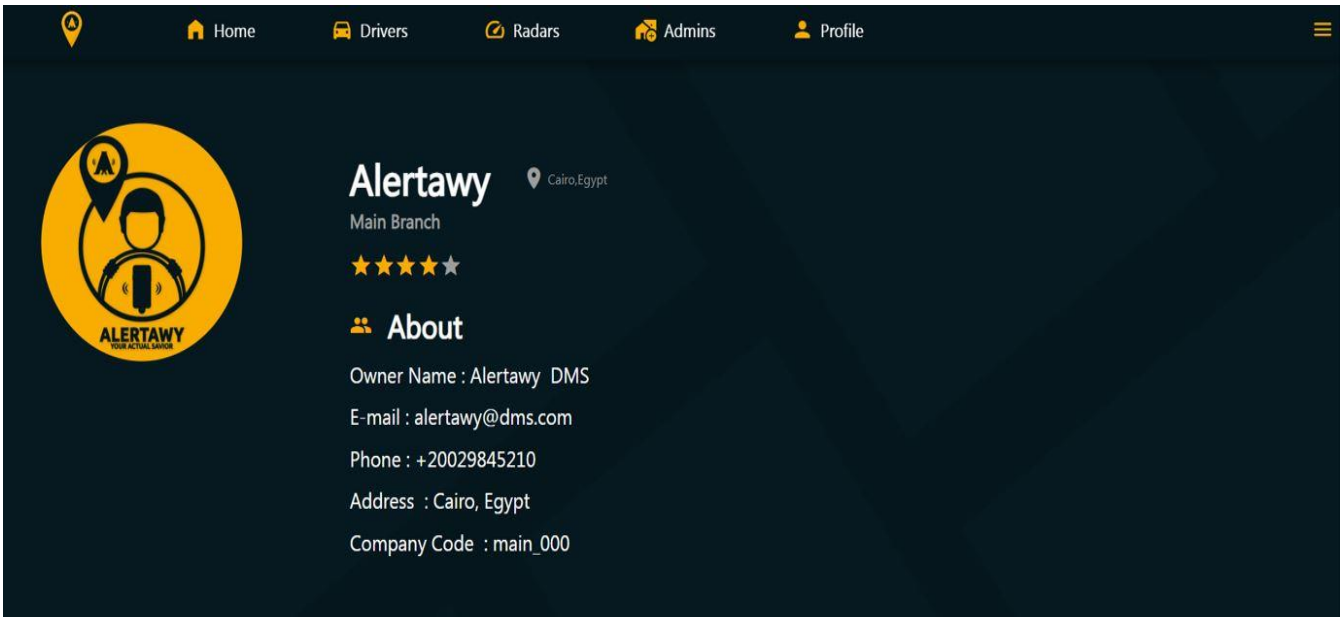
4.4.1.5.2.4) Admin Timeline

Showing the admin timeline since his account has been created on the system.

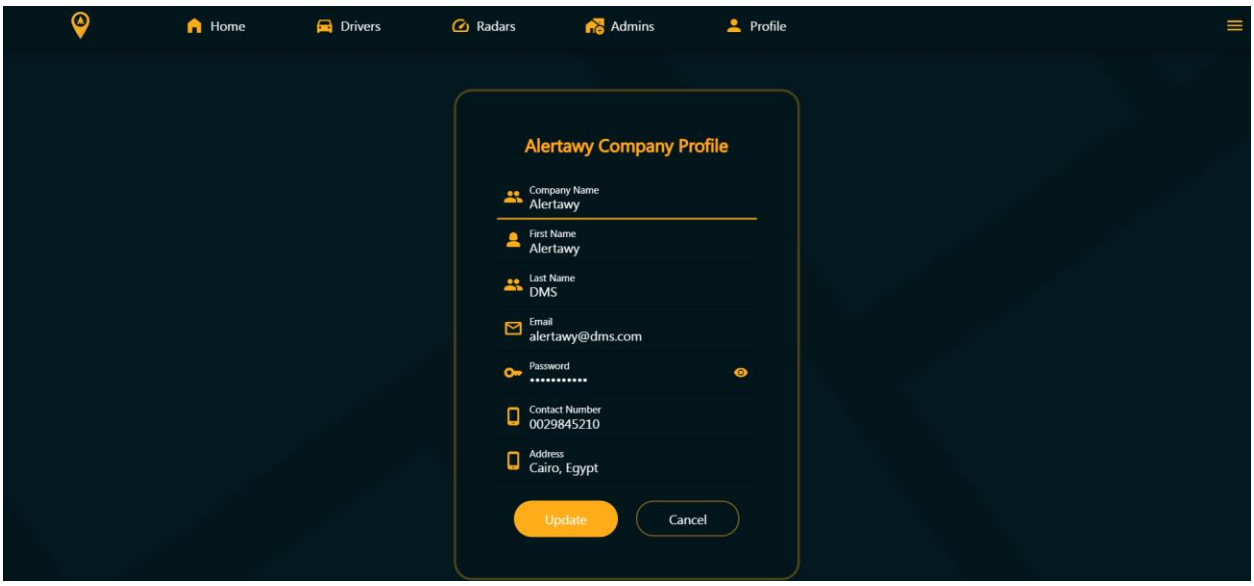


4.4.1.6 Company Profile Page

Showing company information on the system.



4.4.1.6.1 Edit Company Profile Page

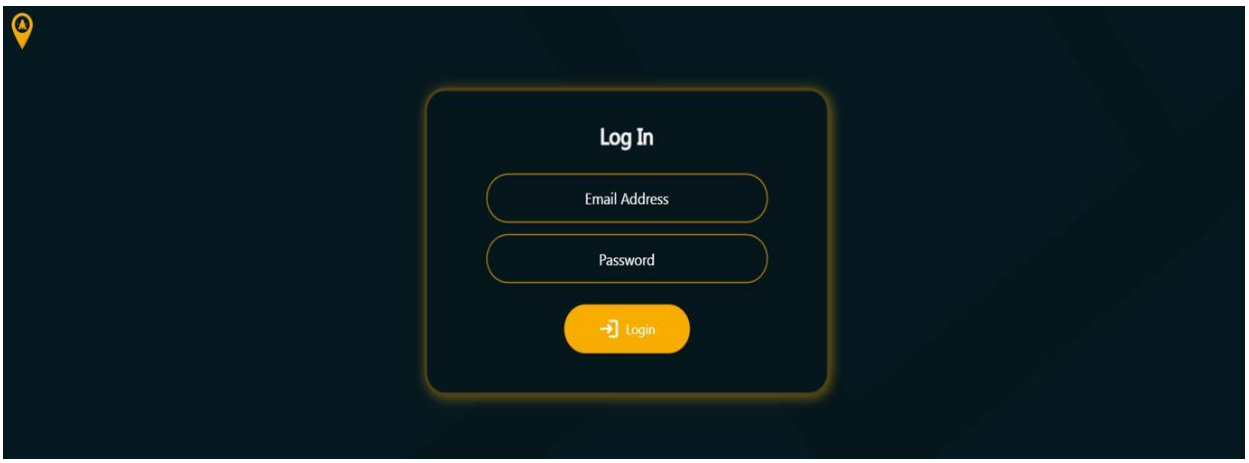


4.4.2 Administrators Role

4.4.2.1 Start Page.

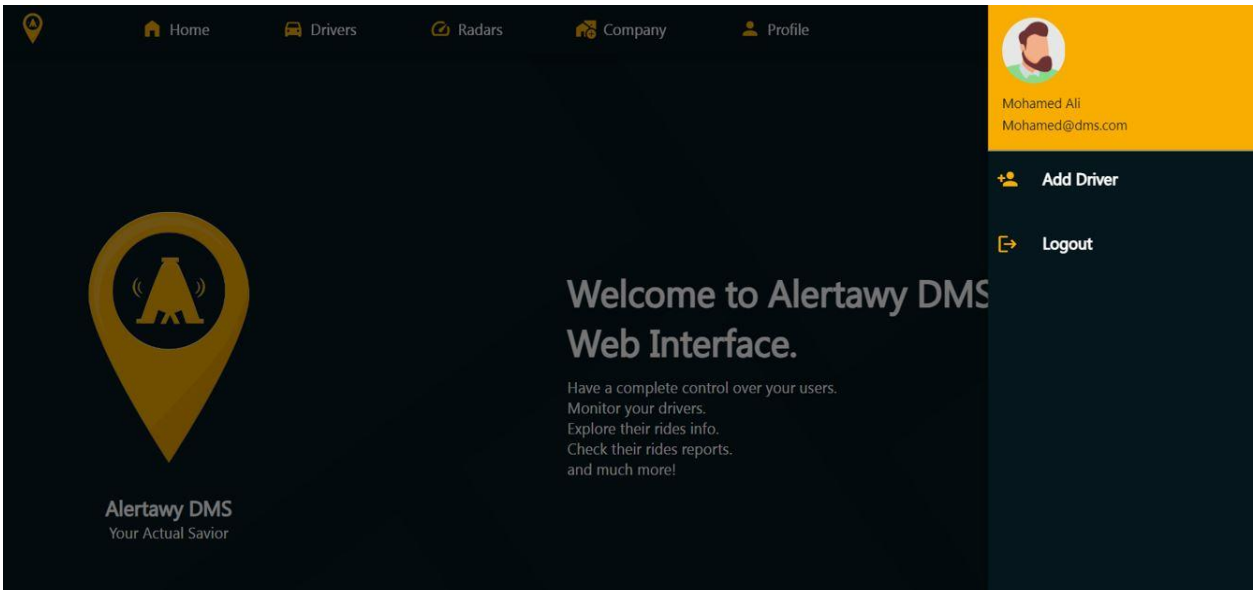
4.4.2.1.1 Login

The admin sign in with Admin email & password and verifies them through Firebase Firestore.



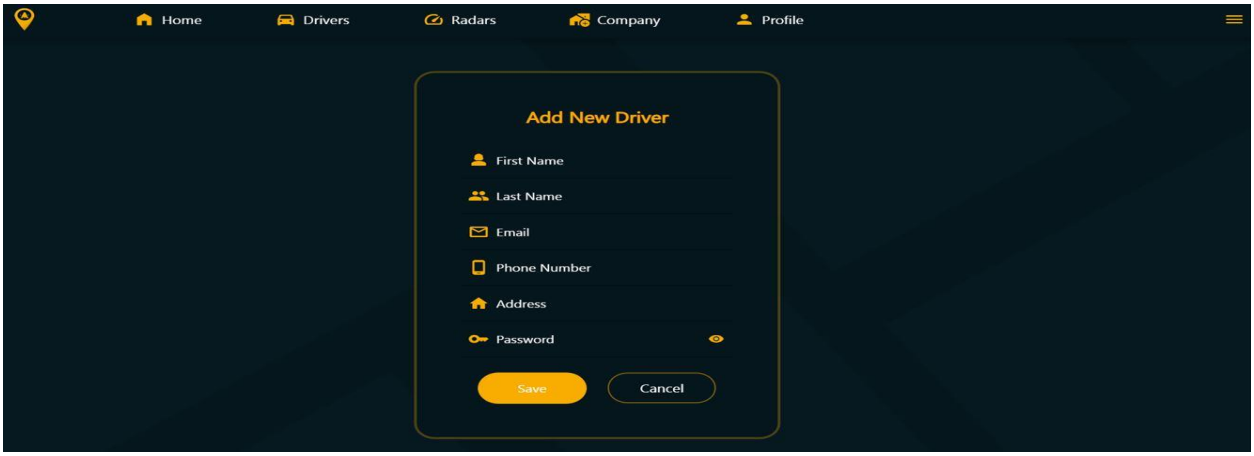
4.4.2.2 Home page

Admin can add driver or logout from home page.



4.4.2.2.1 Add Company Drivers

Add Driver information like First name, Last name, Email, EGY phone number, address & password, then validate these information from some safety functions if something doesn't satisfy the application rules the application will ask for correct information, else these information saved in firebase firestore and email and password stored in Firebase authentication.



4.4.2.2.2 Logout

Same as company look at section 4.5.1.2.2.

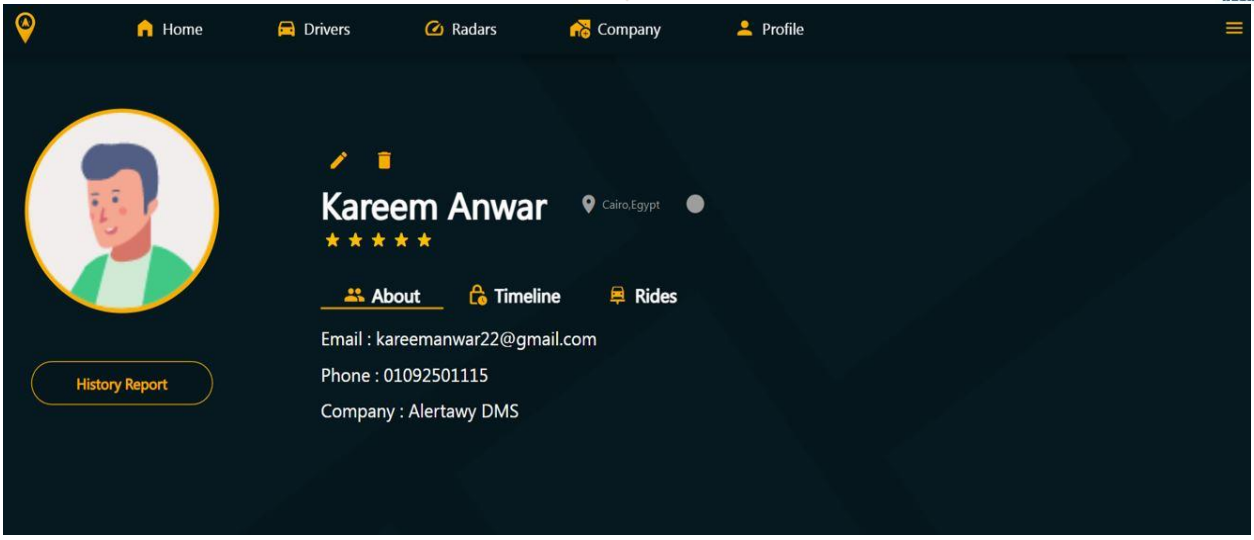
4.4.2.3 Driver's page

4.4.2.3.1 Company Drivers Page

Same as company look at section 4.5.1.3.1

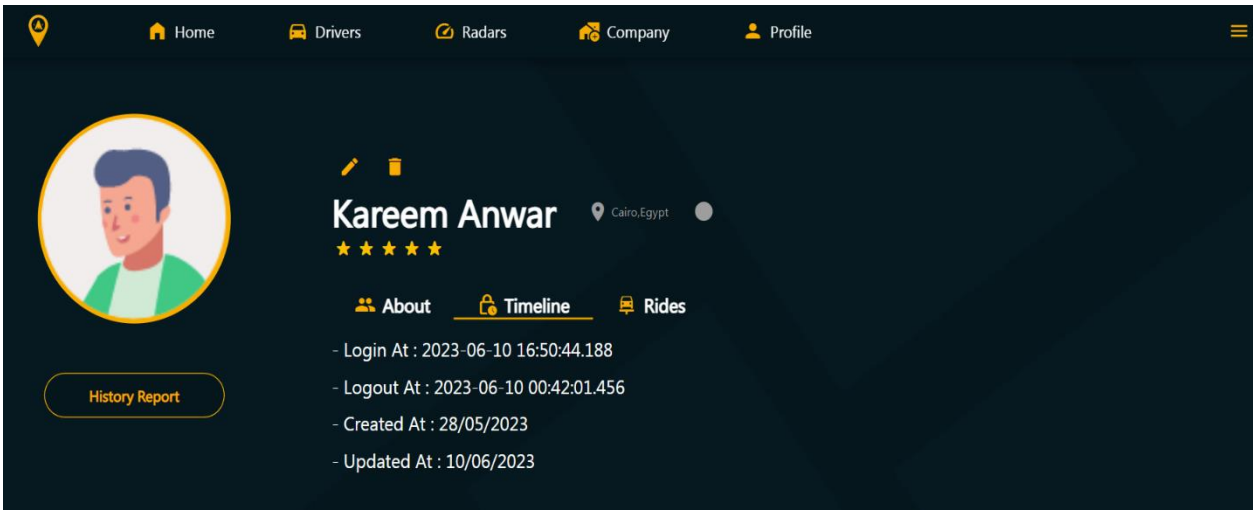
4.4.2.3.2 Driver Profile Page

Showing specific driver information since his account created on the mobile application also showing the driver in ride or not.



4.4.2.3.3 Driver Timeline

Showing the driver timeline since his account has been created on the system.



4.4.2.3.4 View Company Drivers' Past Rides

Same as company look at section 4.5.1.3.4

4.4.2.3.5 View Company Drivers' Rides History on Map

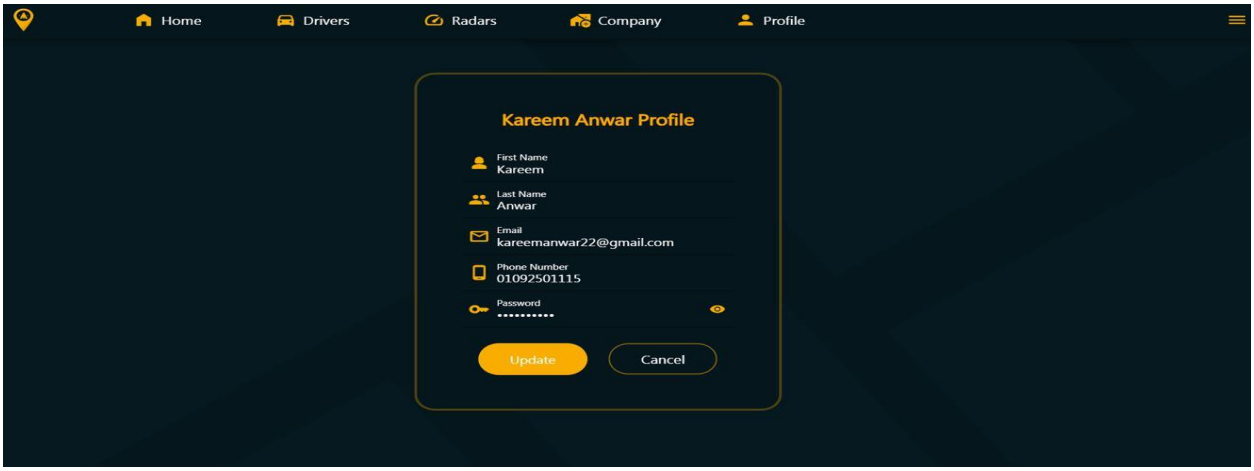
Same as company look at section 4.5.1.3.5

4.4.2.3.6 View Company Drivers' History Report

Same as company look at section 4.5.1.3.6

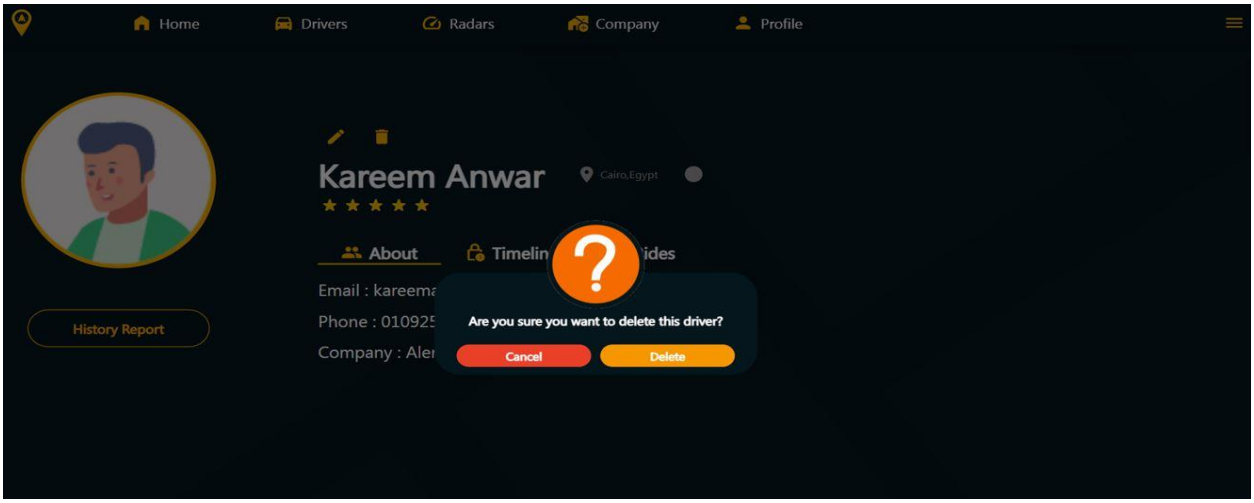
4.4.2.3.7 Edit Company Drivers Info

Editing driver profile with the new information then validate this information from some safety functions if something doesn't satisfy the application rules the application will ask for correct information, else this information saved in firebase firestore and email and password stored in Firebase authentication.



4.4.2.3.8 Delete Company Drivers

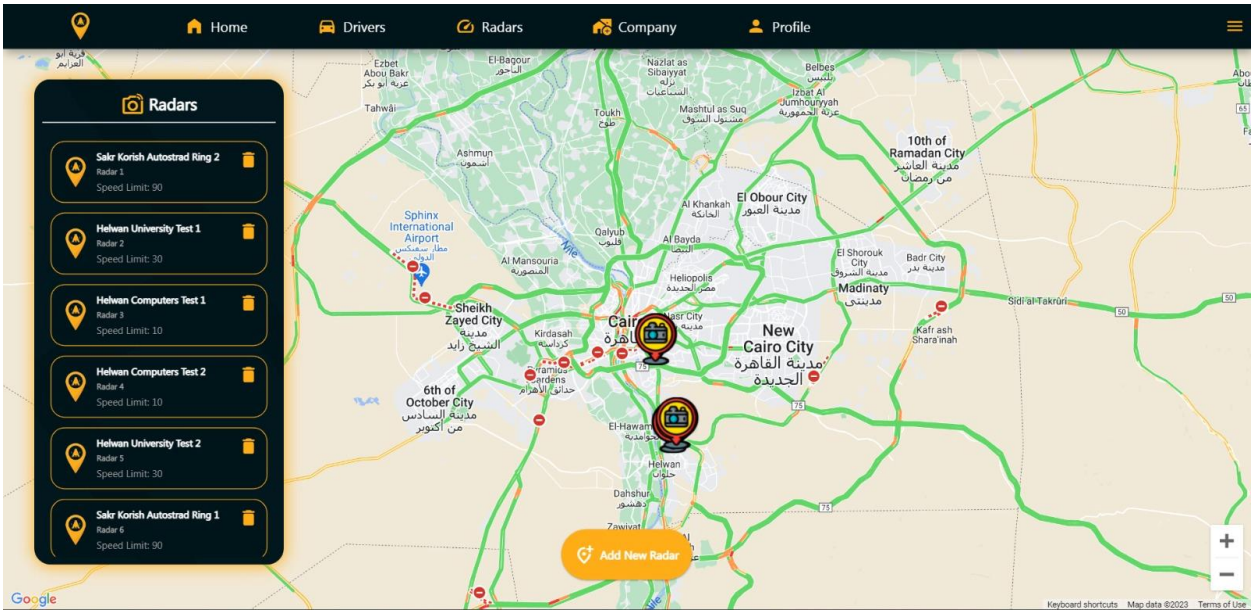
Deleting driver profile information from the Firebase Firestore and image from Firebase Fire storage, email and password from Firebase authentication.



4.4.2.4 Radars

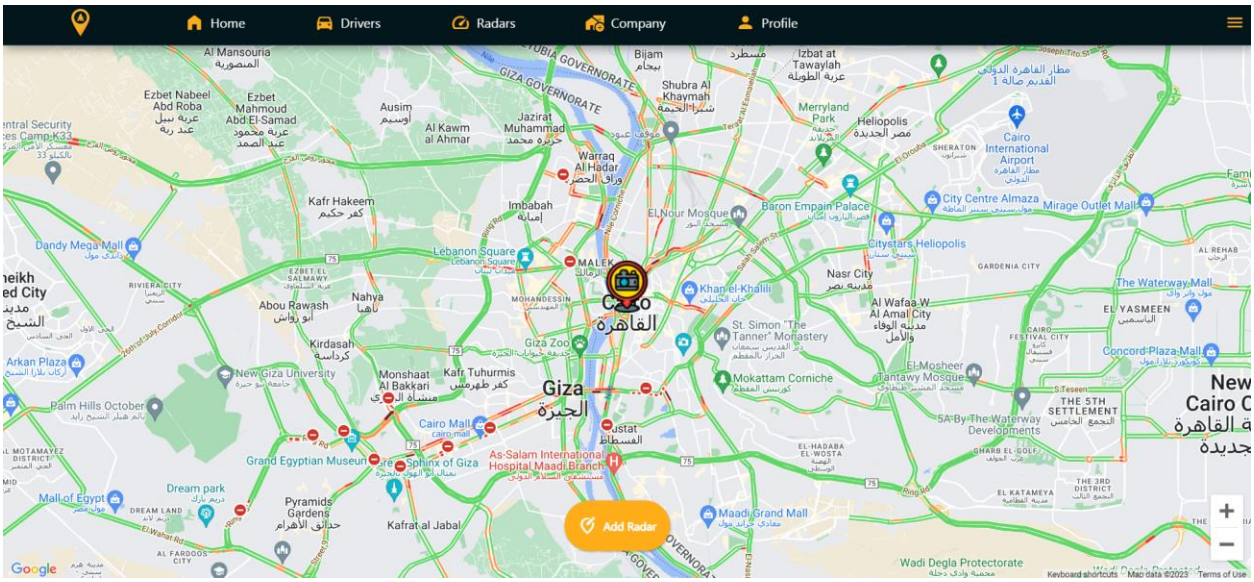
4.4.2.4.1 View Saved Radars on Maps

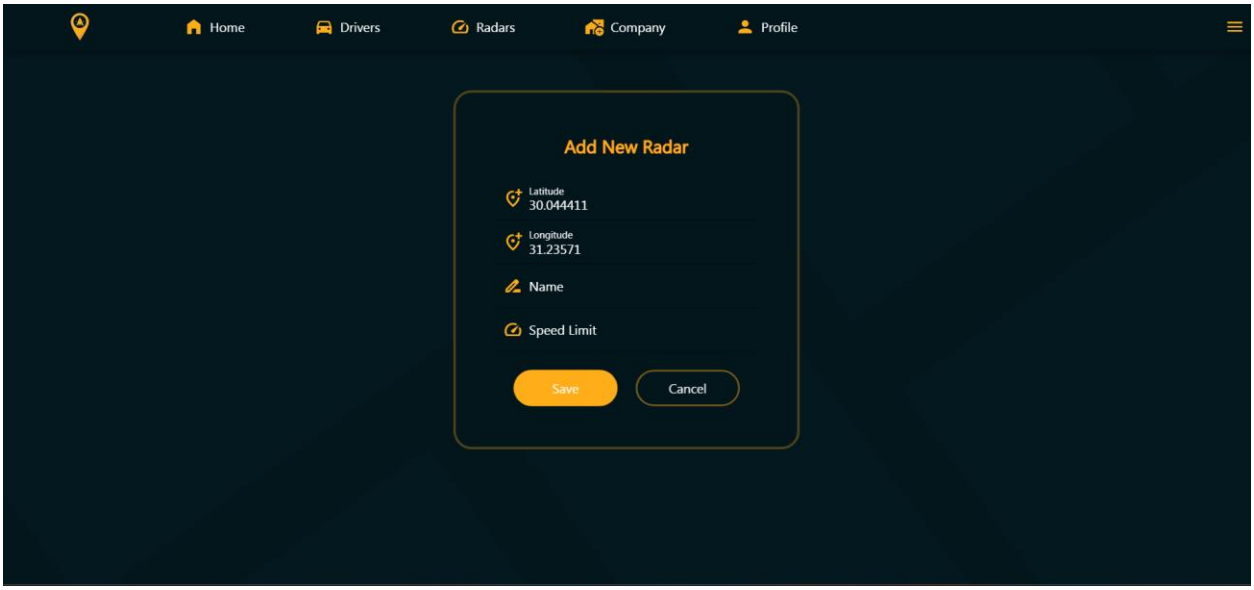
Admins have the ability to access a list of radars, displayed on an integrated map, which are stored in the system's database.



4.4.2.4.2 Add New Radars

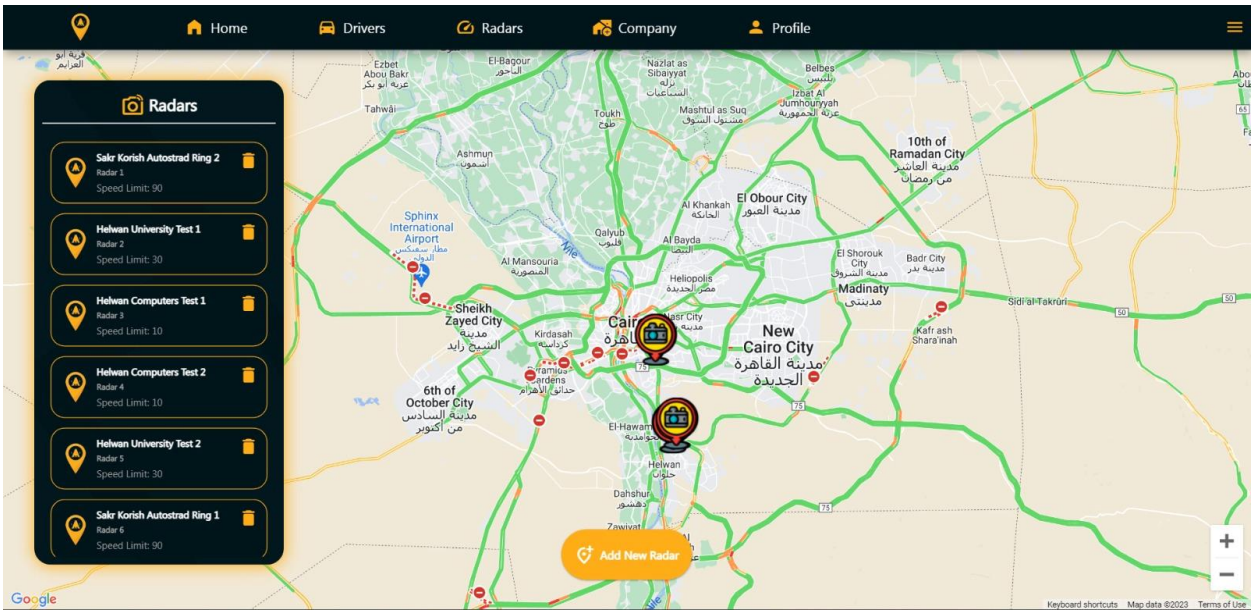
Admins have the ability to add new radars to the system's database, including radar coordinates and speed limit information.





4.4.2.4.3 Delete Saved Radars

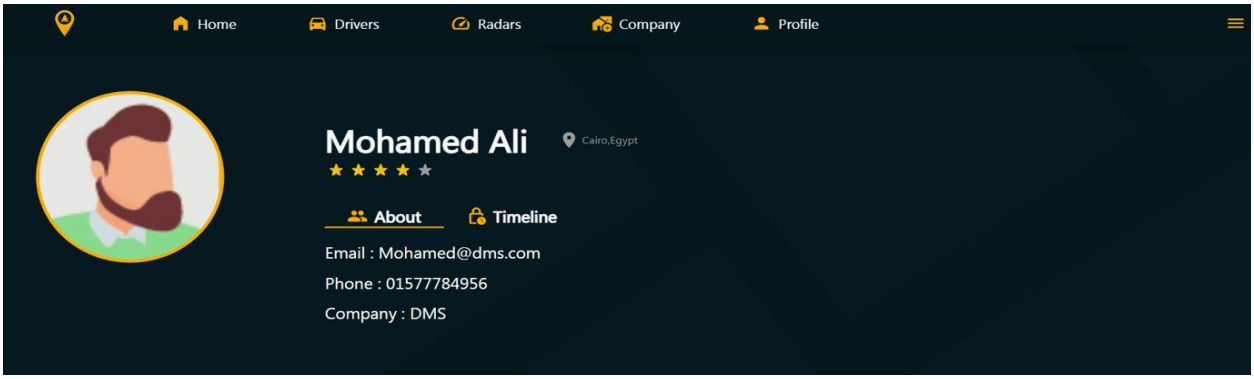
Admins can delete radars that are saved in the system's database.



4.4.2.5 Admin Profile page

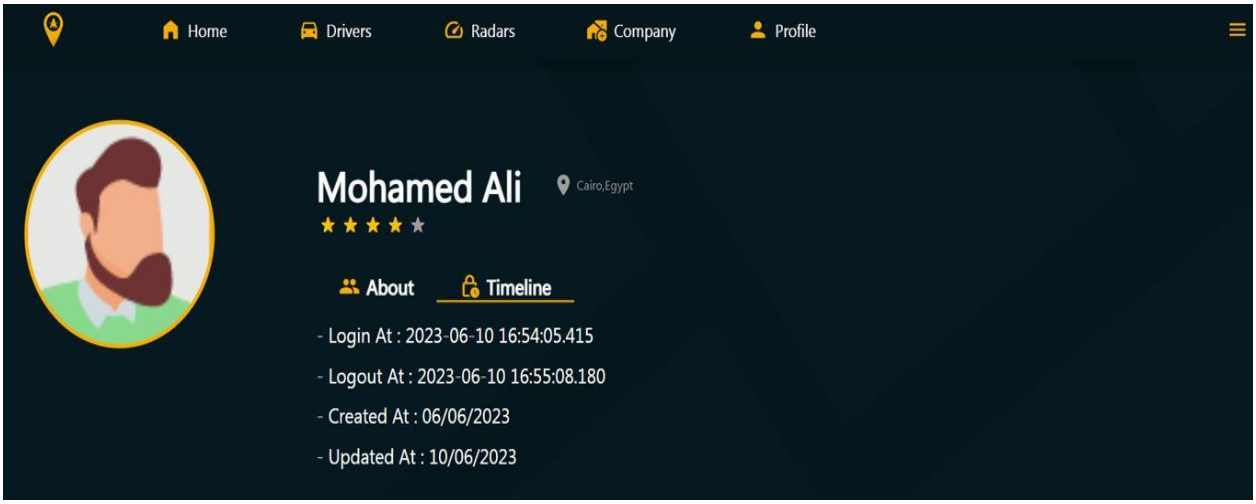
4.4.2.5.1 Admin Profile

Showing admin profile with his information.



4.4.2.5.2 Admin Timeline

Showing the admin timeline since his account was created on the system.



4.4.2.6 View Company Information

View Admin Company information, Same as company look at section 4.5.1.6

Chapter 5: Discussion & Future Work

5.1 Discussion

After an extensive journey, we have successfully developed the AI-Powered Driver Monitor Integrated System, encompassing the following components:

1. Mobile Application: Our mobile application, powered by AI and three deep learning models, plays a crucial role in enhancing driver safety. It detects and alerts drivers in real-time for drowsiness, distraction, and lack of focus on the road. Additionally, it provides features such as radar alerts, displaying speed limits, and the ability to use Google Maps while our application runs in the background.
2. Web Application: The web application serves as a monitoring tool for transportation companies like Uber and Careem. Through this platform, company administrators can access detailed reports generated by the mobile application after each trip, enabling effective driver management and evaluation.

By implementing this integrated system, we have made significant progress in our mission to safeguard drivers, protect lives, and reduce road accidents.

However, it is important to acknowledge the limitations we encountered during our research and development process. One such limitation was the occurrence of false positives in our phone usage detection model. This issue stemmed from resource constraints, particularly the lack of GPU availability for model training. Additionally, training the distraction model proved challenging due to the scarcity of datasets capturing the required shooting angles. Nonetheless, these limitations do not significantly impact the system's overall functionality, and we

remain committed to addressing and improving these areas in future endeavors.

5.2 Summary & Conclusion

At present, our driving monitor system, Alertawy, incorporates a wide range of features, including distraction detection, gaze estimation, drowsiness detection, and radar alerting. Through the development of our mobile application, we aim to protect drivers, preserve lives, and minimize road accidents. Moving forward, our primary focus will be on continuously refining these features to achieve greater accuracy and performance, while consistently updating the application to meet evolving user needs.

5.3 Future Work

1. Enhancing Deep Learning Models: We aim to enhance the performance of our deep learning models by retraining them with newly available datasets. Incorporating these datasets will help improve the accuracy and effectiveness of our models.
2. Distraction with Object Detection: Our future plan includes the development of an advanced object detection model that can accurately determine if a driver is distracted. This model will analyze the object the driver's hands are currently interacting with, enabling precise detection of distractions. To accomplish this, we will rely on the availability of a comprehensive and freely accessible database specifically designed for training the object detection model.
3. Fatigue Detection Model Enhancement: We recognize the importance of accurately detecting driver fatigue to ensure road safety. Therefore, we plan to further enhance our fatigue detection model. This involves refining the algorithm and training the model on additional diverse datasets to improve its accuracy and reliability. By incorporating advanced techniques such as eye

movement analysis, facial expression recognition, and physiological indicators, we aim to create a more robust fatigue detection system. This enhancement will contribute to better driver monitoring and proactive alerts, reducing the risk of accidents caused by drowsiness.

4. Seat Belt Detection: We plan to develop a new deep learning model capable of identifying whether a driver is wearing a seat belt or not. This endeavor will be dependent on the availability of a comprehensive and freely accessible database specifically designed for this task.
5. Radar Camera Database: Following the successful operation of our application, our goal is to integrate a comprehensive database of radar cameras. Regular updates to this database will ensure accurate and up-to-date information for drivers.
6. Web Application Server Hosting: We will strive to provide a dedicated server rather than the firebase hosting to host our Alertawy web application, ensuring seamless operations and effective user service.
7. Release Beta Version for Testing: We plan to release a beta version of our system to gather feedback and conduct extensive testing. This will allow us to gather valuable insights from users and identify any potential issues or areas for improvement before the official launch. By involving users in the testing phase, we can ensure a more refined and reliable system.
8. Cloud Services Constraints: In terms of Cloud Firestore, considering an average usage of 10 document reads, 5 document writes, and 5 document deletions per day per user, we can accommodate approximately 100 users within the free tier limits on a daily basis. As for the Google Maps API, assuming each driver

makes 10 requests to the API per ride, we can support up to 20,000 rides per month under the Maps API free tier.

9. Upgrade Firebase Firestore Tier: As our user base expands, we recognize the need to scale our infrastructure. To accommodate a larger number of users and ensure optimal performance, we will upgrade the current tier of Firebase Firestore, our database management system. This upgrade will provide us with increased storage capacity, higher data transfer limits, and improved scalability, enabling seamless user management and efficient handling of data.
10. Update Google Maps API Plan: With the growing number of users relying on our system for interacting with maps, it is crucial to ensure uninterrupted access to accurate and up-to-date maps. To meet this demand, we will update our current plan of the Google Maps API to a higher tier. This upgrade will enable us to handle a larger volume of user requests, maintain real-time map updates, and provide a smooth navigation experience to our users.
11. Multilingual Support: To cater to the diverse linguistic needs of our user base, we aim to enhance the application's versatility by making it capable of supporting multiple languages.

By pursuing these avenues of future work, we are committed to advancing the capabilities and effectiveness of our AI-Powered Driver Monitor Integrated System, with the ultimate goal of creating a safer and more secure driving experience for all.