FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Real-Time Detection of Driver Fatigue Using Mobile Device Sensors and Artificial Intelligence On-Device

Gonçalo Fernandes Diogo de Almeida

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. António Miguel Pimenta Monteiro

Supervisor at Pink Room: Eng. Mário Gago

July 10, 2025

Real-Time Detection of Driver Fatigue Using Mobile Device Sensors and Artificial Intelligence On-Device

Gonçalo Fernandes Diogo de Almeida

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. José Manuel De Magalhães Cruz
External Examiner: Prof. Paulo Baltarejo Sousa
Supervisor: Prof. António Monteiro

July 10, 2025

Resumo

A fadiga em condutores é uma das principais causas de acidentes rodoviários a nível mundial, comprometendo significativamente o desempenho cognitivo e o tempo de reação. Embora veículos comerciais de gama alta integrem frequentemente Sistemas Avançados de Assistência ao Condutor (ADAS), a sua acessibilidade limitada em contextos de baixos e médios recursos económicos evidencia a necessidade de alternativas viáveis e acessíveis. Esta dissertação apresenta um sistema de deteção de fadiga em tempo-real para smartphones, recorrendo exclusivamente a sensores do próprio dispositivo e a Inteligência Artificial (AI). Ao contrário de abordagens que dependem de sensores fisiológicos ou de hardware integrado no veículo, o sistema implementa métodos baseados em imagem, concebidos especificamente para funcionar eficazmente em dispositivos móveis. Através da integração de modelos de AI de baixa complexidade computacional, como o MediaPipe e o ML Kit, o sistema proposto extrai coordenadas de pontos faciais (*landmarks*) e/ou probabilidades associadas a indicadores de fadiga. Com base nos dados extraídos, são então calculadas métricas como a Percentagem de Fecho das Pálpebras (PERCLOS), o índice de abertura dos olhos (EAR), o índice de abertura da boca (MAR) e a postura da cabeça, todas amplamente reconhecidas na literatura como associados à fadiga. De seguida, os valores calculados são processados por uma camada de decisão baseada em regras personalizadas, de modo a garantir a execução em tempo-real. O sistema foi validado ao utilizar três datasets públicos para investigação (DMD, YawDD e NTHU-DDD) e testado posteriormente em cenários reais com recurso a uma aplicação Android desenvolvida pelo autor. As sessões experimentais envolveram dois participantes distintos, tendo cada um sido monitorizado com dois dispositivos móveis diferentes, e foram validadas com base num vídeo independente de referência, assegurando uma avaliação rigorosa do desempenho em diferentes dispositivos e condições de fadiga. Os resultados confirmaram que o sistema é capaz de detetar fadiga em tempo-real e com elevada precisão, demonstrando assim a viabilidade da sua implementação, exclusivamente em smartphones, de soluções de monitorização baseadas em AI. Este trabalho contribui para a área da segurança rodoviária ao propor um sistema de deteção de fadiga de condutores precoce, acessível e não intrusivo, alinhado com os objetivos da mobilidade inteligente.

Abstract

Driver fatigue is a leading contributor to road accidents worldwide, significantly impairing cognitive performance and reaction time. While high-end commercial vehicles often incorporate Advanced Driver Assistance Systems (ADAS), their limited accessibility in low and middle-income contexts highlights the need for cost-effective alternatives. This dissertation presents a smartphone-based system for real-time fatigue detection, relying solely on on-device sensors and Artificial Intelligence (AI). Unlike approaches that depend on physiological sensors or vehicle-integrated hardware, the system implements image-based methods specifically designed for mobile platforms. By integrating lightweight AI models, such as MediaPipe and ML Kit, the proposed system retrieves facial landmarks and/or fatigue-related probability scores, from which it calculates metrics such as Percentage of Eyelid Closure (PERCLOS), Eye Aspect Ratio (EAR), Mouth Aspect Ratio (MAR), and head pose, which are widely recognized in the literature as indicators of fatigue-related behavior. These features are then processed through a custom rule-based decision layer for real-time operation. The system was validated using the three public research datasets (DMD, YawDD, and NTHU-DDD) and further tested in real-world scenarios through a custom Android application developed by the author. Experimental sessions were performed with two different participants, each assessed individually while simultaneously monitored with two different mobile devices, and validated against independent video ground truth, to ensure a rigorous performance evaluation across different devices and fatigue conditions. The results confirm that the system achieves accurate real-time fatigue detection, demonstrating the feasibility of deploying AI-based monitoring solutions entirely on smartphones. This work contributes to the road safety field of research by proposing an early, accessible, and non-intrusive driver fatigue detection system, aligning with the objectives of intelligent transport safety.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. It includes 17 goals to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

The specific Sustainable Development Goals addressed in this work include:

SDG 3 Ensure healthy lives and promote well-being for all at all ages;

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation;

SDG 11 Make cities and human settlements inclusive, safe, resilient and sustainable.

SDG	Target	Contribution	Performance Indicators and Metrics
3	3.6	Contributing to the reduction of road traffic injuries and fatalities by detecting driver fatigue in real-time.	Decrease in fatigue-related accident rates; Number of usages of fatigue detection mobile applications; Accuracy of fatigue detection models.
	3.d	Contributing to early warning and risk reduction in public health by enabling real-time detection of driver fatigue.	Number of research prototypes; Validation studies completed.
9	9.5	Advancing research and innovation in AI-based safety systems through real-world experimentation with mobile devices.	Publication of research outcomes; Development of mobile AI inference frameworks; Performance benchmarks under resource-constrained environments.
11	11.2	Improving road safety as part of sustainable transport systems by integrating real-time fatigue detection capabilities into vehicular applications, particularly benefiting vulnerable road users such as older drivers or individuals with impaired attention.	Potential integration with Intelligent Transport Systems; Reduction in accident risk under urban and highway scenarios.

Acknowledgements

I would like to thank Professor António Pimenta Monteiro for the support and guidance provided during the development of this dissertation, which coincided with his final year of academic teaching.

To Mário Gago and Bruno Correia, for believing in me and supporting my growth from day one.

To Rafael, a friend before a colleague, whose weekly support and calm reassurance helped me push through the most stressful moments, and, just as he said, I made it to the end.

To André, João, Lucas, and Rodrigo, for welcoming me into Pink Room and patiently putting up with all my questions, a natural consequence of my curiosity about everything.

To my family, my mother, for all the phone calls we had while I was in Porto, always knowing exactly what to say, or when to just listen. To my father, for the quiet support and strength behind every decision I made. To my sister, who is no longer the "little one", for constantly annoying me just enough to remind me what really matters. And thank you all for the countless times you heard: “I can’t, I have a meeting.”

To my university friends from Coimbra, where I did my bachelor’s, especially Bernardo, Faria, Filipe, Miguel, Nuno, and Santana, for all the laughs, support, and memorable times that kept me going even from afar.

To my university friends from Porto, most of them from JuniFEUP, a Junior Enterprise that became much more than a company, I’m grateful for all the friendships, lessons, and moments we shared.

To António, Guilherme, Mendes, and Nuno, for these seventeen years together and for many more to come.

To Mafalda and all my friends from Porto, for a friendship that started in France and grew into something truly special, filled with unforgettable memories.

To all who marked my journey positively, I truly appreciate your presence. These past three plus two years have been amazing!

Gonçalo

"You only have to do a very few things right in your life
so that you don't do too many things wrong."

Warren Buffett

Contents

List of Figures

List of Tables

List of Abbreviations

ADAS	Advanced Driver Assistance Systems
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSV	Comma-separated values
DMD	Driver Monitoring Dataset
EAR	Eye Aspect Ratio
ECG	Electrocardiogram
EEG	Electroencephalography
FN	False Negative
FP	False Positive
FR	Functional Requirement
GB	Gigabyte
GPU	Graphics Processing Unit
HRV	Heart Rate Variability
iOS	iPhone Operating System
IoT	Internet of Things
KNN	K-Nearest Neighbors
KSS	Karolinska Sleepiness Scale
MAR	Mouth Aspect Ratio
MLOps	Machine Learning Operations
MVI	Model-View-Intent
MVVM	Model-View-ViewModel
NFR	Non-Functional Requirement
NLR	Nose Length Ratio
NTHU-DDD	National Tsing-Hua University Drowsy Driving Dataset
PERCLOS	Percentage of Eye Closure
PoC	Proof of Concept
PPG	Photoplethysmography
RAM	Random access memory
SDGs	Sustainable Development Goals
SDK	Software Development Kit
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
UDF	Unidirectional Data Flow
UI	User Interface
YawDD	Yawning Detection Dataset

Chapter 1

Introduction

This dissertation aims to explore and develop innovative solutions by leveraging artificial intelligence models and mobile device sensors. The study is conducted in collaboration with Pink Room[1], a digital product studio specializing in developing Android and iPhone Operating System (iOS) mobile applications, guiding projects from the initial idea to the final product.

1.1 Context

The issue of driver fatigue has emerged as a significant global concern in road safety, with fatigue-related incidents being identified as a primary cause of severe accidents [1]. According to the British and American collision statistics, in countries with high and middle incomes, approximately 93% of all road collisions are attributed to tiredness, driver error, and other human variables [2]. Among the leading contributors to these errors, often collectively referred to as the "Big Three", are alcohol, drowsiness, and inattention [3]. Fatigue, as a key component of drowsiness, severely impairs drivers' cognitive functions, reaction times, and decision-making capabilities. This impairment consequently elevates the risk of road accidents, notably contributing to 36% of all fatal ones [4].

The modern driving environment, characterized by extended periods of monotonous driving on highways and an increased prevalence of vehicle automation, heightens the risk of driver disengagement [5]. Because of that, companies such as Tesla[2] and Volvo[3] have developed Advanced Driver Assistance Systems (ADAS) to address these issues. However, their widespread adoption is constrained by cost and technological complexity, making them inaccessible to the majority of drivers, particularly in low and middle-income regions [6]. A more accessible alternative lies in mobile devices. Although they lack continuous vehicular data integration, which may affect robustness in certain scenarios, they are widely available and equipped with sensors such as accelerometers, gyroscopes, and cameras. When combined with artificial intelligence (AI) algorithms, these sensors offer significant potential to facilitate real-time cost-effective monitoring of

[1] https://pinkroom.dev/
[2] https://www.tesla.com/
[3] https://www.volvocars.com/

driver fatigue, playing a critical role in preventing accidents on a broader scale by enabling detection of these factors earlier [7].

1.2 Motivation

The primary motivation for this research originates from the need to explore and democratize access to practical and accessible solutions for detecting driver fatigue using devices already integrated into everyday use: smartphones [8]. Leveraging its sensors and computing capabilities embedded in these devices, AI, particularly in mobile-compatible forms, has shown increasing promise in applications for understanding and interpreting human expressions and cognitive states. However, the integration of AI for fatigue detection on mobile platforms introduces unique challenges: limited processing power, energy constraints, and the variability of sensor data. This research seeks to examine these limitations directly by developing and testing a fully on-device fatigue detection system.

While the potential of mobile AI for fatigue detection is significant, the current landscape of smartphone-based solutions remains fragmented. Several existing approaches depend on cloud infrastructure [9], additional hardware [7], or sensor data that is unsuitable for deployment in typical driving environments [2]. In contrast, this research aims to explore the viability of capturing and processing head movements and facial features, particularly eye and mouth behaviors, without reliance on external computation or infrastructure. This proof of concept is intended to assess the feasibility and performance trade-offs of on-device driver fatigue detection, rather than to produce a commercial-ready product.

This research implements AI tools suitable for on-device deployment, such as ML Kit and MediaPipe, and will include an evaluation and analysis of driver fatigue under real-world, driving-like conditions, along with a critical assessment of the limitations and strengths of the proposed approaches. In doing so, it contributes to the broader field of driver monitoring by providing insights into how far mobile-only solutions can go in addressing a well-established safety challenge, and where future work may be required to scale or refine such approaches.

1.3 Research Questions

As established in the previous sections, this research aims to develop a comprehensive driver fatigue detection system leveraging mobile devices and AI techniques. The research questions guiding this dissertation are as follows:

1. How can mobile device sensors and lightweight AI algorithms be effectively integrated to detect driver fatigue in real-time on resource-constrained mobile platforms? [2, 10, 11]

2. What trade-offs exist between latency, energy consumption, and inference accuracy in smartphone-based driver fatigue detection using real-time AI processing? [12, 13]

These research questions are designed to address the gaps in existing solutions for driver fatigue detection, particularly focusing on real-time monitoring on mobile platforms. By answering these questions, this dissertation aims to explore a solution that overcomes the current limitations and provides accurate results in a resource-constrained platform.

1.4 Document Structure

This document is structured into seven chapters, each addressing a fundamental stage of the research and development process. Chapter 2 provides a review of the literature related to driver fatigue detection, focusing on existing methods, AI-based approaches, and current technological challenges. Chapter 3 outlines the requirements that shaped the proposed proof of concept solution. Chapter 4 details the system architecture and development, while Chapter 5 describes the experimental methodology, including data-driven pre- and post-processing, as well as on-device validation through both dataset-based and real-world testing. Chapter 6 discusses the results and analysis of the evaluation of the system. Lastly, Chapter 7 concludes the work with the key findings, implications, and potential directions for future research.

Chapter 2

State of the Art

Driver fatigue detection has emerged as a multidisciplinary research field intersecting transportation safety, computer vision, and mobile computing. Given the various methods and technologies proposed over the years, this chapter adopts a narrative approach to review the literature. Rather than presenting an article-by-article analysis, the selected works are grouped into conceptual categories that reflect the methods for detecting driver fatigue using subjective, behavioral, physiological, and visual cues; the application of artificial intelligence (AI) for fatigue detection on mobile and embedded platforms; and the main challenges and limitations that affect current approaches, including environmental variability and real-time performance constraints.

2.1 Overview of Driver Fatigue Detection in Road Safety

Driver fatigue and drowsiness are critical factors contributing to traffic accidents [14]; with fatigue referring to a state of reduced alertness and performance resulting from prolonged mental or physical activity, leading to slower reaction times and increased risk of accidents [1]; and drowsiness represents an unstable, rapidly fluctuating state of reduced alertness along the sleep-wake continuum, which can result in large variability in performance impairment over short periods of time [15]. The two conditions manifest similarly with observable indicators such as eye closure, head tilts, and yawning [16]. Given their overlapping symptoms and the fact that they are commonly detected using the same techniques, this dissertation treats fatigue and drowsiness under a unified framework, as both contribute comparably to decreased driver alertness.

2.2 Methods for Detecting Driver Fatigue

Identifying driver fatigue represents a pivotal objective in pursuing enhanced road safety, given the considerable influence that these factors exert on the occurrence of traffic accidents. A variety of approaches have been developed for the monitoring and assessment of a driver's state, utilizing a diverse range of data sources and technologies.

Figure 2.1 provides an overview of the categories of detection methods used for this purpose that will be detailed in the subsequent sections. In them, it will be described each of these methods, discussed their principles, and the tools utilized, with references to prior research that has applied these techniques in the context of driver fatigue detection.

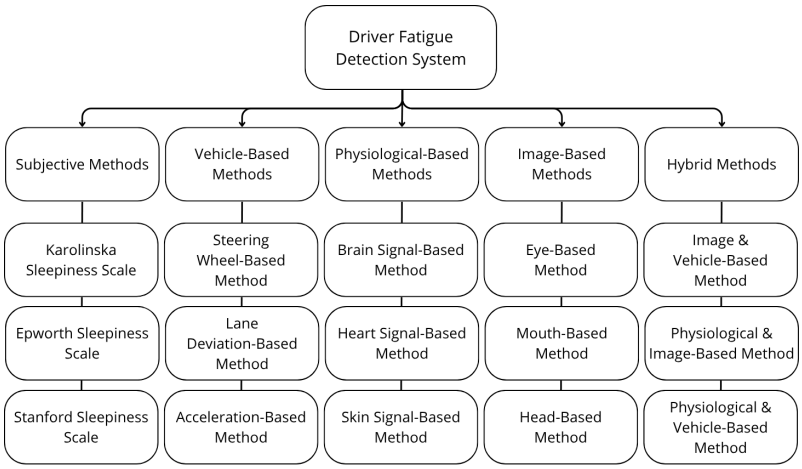


Figure 2.1: Different types of detection methods.

2.2.1 Subjective Methods

This type of methods rely on self-reported assessments or the driver's perception of their state to evaluate the presence and extent of fatigue. These methods are frequently used due to their simplicity and ease of implementation, as they typically involve questionnaires or scales that require minimal equipment. However, their reliability can be compromised by subjective bias and inconsistencies in individual responses, which restrict their use as standalone methods in critical safety applications.

One widely used instrument is the Karolinska Sleepiness Scale (KSS), which offers a transient self-evaluation of sleepiness on a 9-point scale ranging from "Extremely alert" to "Very sleepy". The scale has been validated in real-world driving studies and has served as a benchmark for developing more objective fatigue detection systems [17].

Another frequently utilized subjective tool is the Epworth Sleepiness Scale, which measures a driver's general level of daytime sleepiness through a series of questions assessing their likelihood of dozing off in different scenarios [18]. This tool has been extensively applied in studies examining the correlation between excessive sleepiness and impaired driving performance, thereby establishing it as a valuable resource for detecting long-term fatigue. However, its efficacy in real-time fatigue detection is constrained by its reflection of long-term patterns rather than momentary states.

Complementing these tools, the Stanford Sleepiness Scale provides an additional subjective instrument that, like the KSS, presented in Table 2.1, is designed to assess momentary sleepiness. The scale uses a seven-point classification system, ranging from "feeling active and vital" to "almost in a dream-like state" [19].

Table 2.1: Karolinska Sleepiness Scale Table (Adapted from [20]).

Level	**Sleepiness Description**
1	Extremely alert
2	Very alert
3	Alert
4	Rather alert
5	Neither alert nor sleepy
6	Some signs of sleepiness
7	Sleepy, but no effort to keep awake
8	Sleepy, some effort to keep awake
9	Very sleepy, great effort to keep awake, fighting sleep

In conclusion, subjective methods are valuable for their simplicity and low-cost implementation, rendering them suitable for preliminary assessments [1] or as part of comprehensive detection frameworks. However, their dependence on self-reported data introduces intrinsic limitations, such as subjective bias, underreporting of fatigue, and variability in individual responses. To overcome these shortcomings, subjective methods are most effective when integrated with objective approaches, enhancing their reliability and robustness in real-world applications.

2.2.2 Vehicle-Based Methods

In vehicle-based methods, data generated by the vehicle's systems is leveraged to identify indications of fatigue. These approaches are non-intrusive, as they rely on external cues rather than the driver's physiological or visual behavior, but they are normally only found in modern vehicles.

The analysis of steering angle data offers invaluable insights into driver behavior, particularly in the context of fatigue. For example, Li et al. in [21] propose a fuzzy recurrent neural network model for the analysis of time-series data from steering wheel sensors. The model exhibited robust fatigue detection capabilities with an accuracy of 87.3% under real-world conditions, identifying patterns such as reduced steering precision and delayed corrective actions.

Another method is the monitoring of lane deviation by using sensors and cameras to track the position of the vehicle in relation to lane markings, thereby enabling the detection of unintended lane departures and the enhancement of driver safety. More sophisticated implementations, such as Volkswagen's Lane Assist, utilize cameras to monitor road markings and are capable of gently counter-steering the vehicle to maintain in its lane [22]. These systems are particularly effective in reducing accidents caused by lapses in driver attention.

Abrupt changes in acceleration or deceleration can also be indicative of driver fatigue or aggressive driving. Rishu Chhabra et al. in [23] mention the usage of smartphone sensors to analyze acceleration patterns, demonstrating the feasibility of using such data for detecting anomalous

driving events. However, the primary objective of these methods is not merely to assess driving behavior, but rather to identify signs of fatigue that may underlie such deviations. This shift in focus underscores the potential for these systems to enhance safety by addressing the root causes of risky driving behavior.

Modern Advanced Driver Assistance Systems (ADAS) integrate a variety of vehicle-based monitoring techniques, including lane-keeping assistance, collision avoidance systems, and other automation functions, to enhance safety [24]. These systems not only support driving tasks but also compensate for human errors caused by fatigue, improving real-time safety interventions [25].

2.2.3 Physiological-Based Methods

Physiological-based methods rely on the measurement of the driver's internal biological signals to assess states of fatigue or drowsiness. These methods provide accurate and robust insights into the driver's state by analyzing physiological parameters such as brain activity, heart rate variability, skin conductance, and blood volume changes.

Electroencephalogram (EEG) is considered the gold standard for monitoring brain activity and detecting fatigue, as it directly measures neural activity. In the study by Wang et al. [26], a real-time system was developed using dry EEG sensors to detect driver fatigue. The approach combined power spectrum density analysis and entropy metrics to identify fatigue-related brainwave patterns, such as theta and alpha bands, and achieved effective fatigue prediction in simulated driving scenarios.

Electrocardiogram (ECG) signals are widely used to monitor heart rate variability (HRV), a sensitive marker of fatigue and stress. Suganiya Murugan et al. [27] demonstrate a system that extracted features such as Root Mean Square of the Successive Differences and standard deviation of normal RR-interval from ECG signals to classify different driver states, including drowsiness and cognitive inattention. The system used machine learning classifiers such as Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) and achieved accuracies of up to 96.6% for binary classification.

On a similar note, Galvanic Skin Response measures the electrical conductance of the skin, which can be indicative of stress or fatigue. Munir et al. [28] proposed a wearable, an Internet of Things (Iot) based system that combined this measures and HRV data to predict driver fatigue. This system utilized real-time monitoring and alerting mechanisms, demonstrating its potential as a low-cost and wearable solution.

Photoplethysmography (PPG) is a non-invasive technique that detects changes in blood volume in the microvascular bed of tissue. Based on the findings of Andrea Amidei et al. [29], a PPG-based prototype for driver drowsiness detection has the ability to analyze cardiac activity and waveform parameters. Although the method demonstrated favorable outcomes, the researchers highlighted that motion artifacts and variable lighting conditions could potentially restrict its efficacy in authentic settings.

In summary, physiological-based methods provide an accurate and reliable approach for detecting fatigue and drowsiness by utilizing internal biological signals. Despite their potential,

challenges such as motion artifacts, environmental interference, and the necessity for contact sensors must be addressed to guarantee their efficacy in real-world applications.

2.2.4 Image-Based Methods

Image-based methods, also known as visual behavior-based methods, leverage the analysis of facial features, eye movements, and head posture to detect indications of fatigue in drivers. By utilizing computer vision and machine or deep learning, these methods provide non-intrusive and effective solutions for real-time monitoring. Incorporating these technologies into Driver Monitoring Systems has become increasingly prevalent in response to the rising demand for road safety technologies.

One of the most widely used metrics in the field of visual fatigue detection is the Percentage of Eye Closure (PERCLOS). The metric in question measures the proportion of time a driver's eyes remain closed within a specified interval, which may range from thirty seconds to one minute, and it is calculated by assessing varying states of eye openness, as exhibited in Figure 2.2. This metric is particularly effective because prolonged eye closures are strongly correlated with drowsiness, often preceding significant impairments in reaction time and cognitive performance [30].

Figure 2.2: Illustration of varying degrees of eye openness (Adapted from [31]).

In practice, the PERCLOS is calculated using real-time video streams from cameras that track the driver's eyelids. The algorithms identify key points around the eyes in order to compute the Eye Aspect Ratio (EAR), a value that changes as the eyes close and open. A sustained low EAR over several frames indicates partial or full eye closure. Eddie E. Galarza et al. in [30] demonstrate a PERCLOS-based system implemented on smartphones, which operates effectively under natural lighting conditions. The system incorporates blink frequency and eye closure duration to further enhance the robustness of fatigue detection, achieving high detection accuracy during testing. Furthermore, PERCLOS has been demonstrated to be a reliable indicator of fatigue in a range of studies, thereby establishing it as a fundamental component in the monitoring of commercial drivers.

Facial expressions offer additional indications of fatigue, with yawning frequency being one of the major indicators. Yawning is frequently indicative of decreased alertness and diminished cognitive function, which makes it a pivotal metric for fatigue detection systems. Dong and Lin [7] developed a method for detecting yawning by calculating the Mouth Aspect Ratio (MAR), which is obtained from the distance between specific facial landmarks around the mouth. The importance of yawning frequency as a metric is further underscored in Figure 2.3, which illustrates its substantial contribution to fatigue detection in comparison to other visual indicators, such as blink and head nodding/tilting frequency.

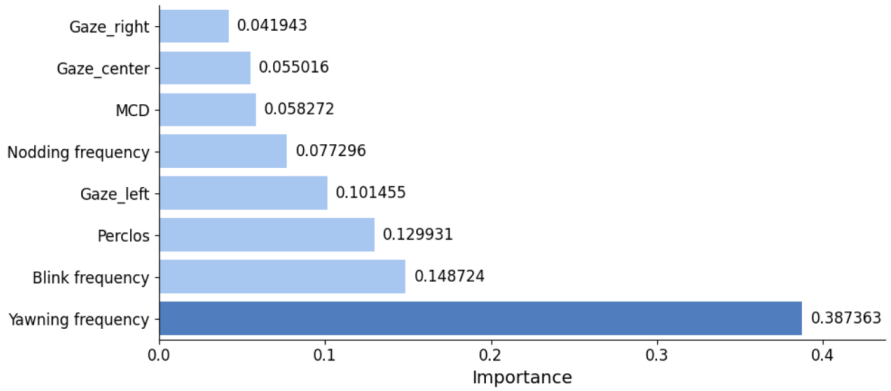


Figure 2.3: Proportional contribution of yawning relative to other facial metrics for fatigue detection. Yawning frequency is the most influential factor, accounting for over one-third of the total importance [7].

Huang and Lin [32] proposed a system capable of distinguishing yawning from regular speech by analyzing the duration and frequency of mouth openings, as shown in Figure 2.4. Yawning events were identified when the MAR remained above a pre-established threshold for an extended period, typically longer than regular speaking intervals. The integration of yawning detection with PERCLOS and blink metrics resulted in a notable enhancement in system accuracy, offering a more comprehensive assessment of driver fatigue.

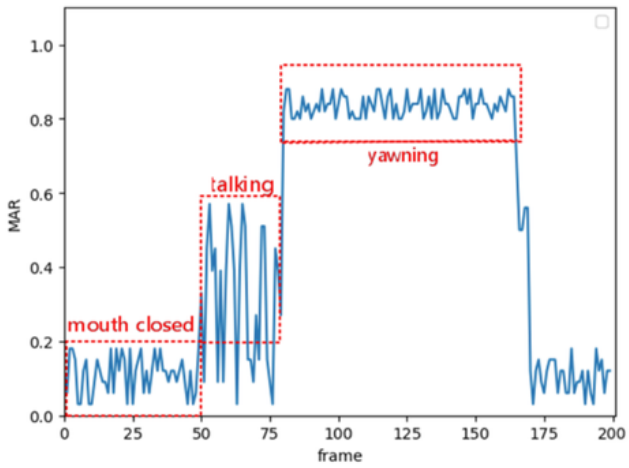


Figure 2.4: MAR value of 200 frames distinguishing yawning [32].

Head position and movement serve as valuable indicators for detecting driver fatigue, as drowsy drivers frequently exhibit forward head tilts and head nodding, reflecting declining postural control as fatigue progresses. Kim et al. [33] introduced an infrared-based head-tracking system capable of estimating horizontal and vertical head angles under varying lighting conditions, enabling the detection of inattentive behaviors such as gaze deviations and sustained head tilts. In parallel, Kumar et al. [34] analyzed facial landmark-based geometric to compute the Nose Length Ratio (NLR), where variations in the apparent nose length correlate with forward head bending.

These head-based indicators provide complementary information to facial expression metrics and remain particularly useful in scenarios where other visual cues may be partially obscured or less reliable.

Image-based methods offer a powerful and non-intrusive means of detecting driver fatigue. Metrics such as PERCLOS, yawning frequency, and head position tracking provide detailed insights into the driver's state. However, real-world challenges, including variable lighting, occlusions, and computational demands, highlight the need for continued advancements. By integrating multiple metrics and leveraging deep learning, these systems are becoming increasingly reliable and practical, contributing to safer roads and reduced accidents. Finally, this type of method will form the foundation of the approach utilized in this dissertation to detect driver fatigue effectively.

2.2.5 Hybrid Methods

Hybrid methods for detecting driver fatigue integrate features from diverse methods, including image-based, physiological-based, and vehicle-based metrics. This integration allows the usage of the respective strengths of each method, addressing the inherent limitations of individual approaches. These systems typically enhance accuracy and robustness, particularly in real-world scenarios where single-method approaches may prove inadequate.

One approach integrates image and vehicle-based methods, combining eye-tracking metrics with vehicle parameters such as lane deviations and steering patterns. For example, Baccour et al. [17] demonstrated how a SVM classifier could combine such metrics to achieve reliable fatigue detection, highlighting the effectiveness of multi-modal sensor fusion for driver state monitoring.

Another noteworthy hybrid system employs a combination of physiological and image-based methods. Karuppusamy et al. [35] presented a multimodal system incorporating EEG signals for monitoring brain activity, gyroscope data for head motion, and visual data for facial expression analysis. This hybrid approach demonstrated a detection accuracy of 93.91%, emphasizing its potential for robust fatigue detection in a variety of scenarios.

Furthermore, there is a growing trend of combining physiological and vehicle-based methods, and in some cases, also image-based data, for multi-modal fatigue monitoring. In the paper by Wang et al. [36], a fusion model that integrates physiological data from smartwatches and vehicle dynamics data from smartphones is proposed. The approach utilizes an Attention Long Short-Term Memory neural network to process and combine these data streams, achieving an F1-score of 80.31% for fatigue detection under real-world driving conditions, a result that reflects a strong balance between correctly identifying fatigue and avoiding false detections.

Hybrid systems are not of particular relevance to this dissertation, as its focus is on the non-intrusive and lower-cost way to detect fatigue, especially through the use of a smartphone. These devices are already widely used by drivers, which makes them an ideal platform for developing affordable fatigue detection systems that do not require the use of specialized equipment or complex integrations. By focusing on smartphone-based solutions, this dissertation aims to explore a scalable and user-friendly approach that can be easily deployed in everyday driving environments.

2.3 AI for Fatigue Detection on Mobile and Embedded Platforms

The application of AI in driver fatigue detection has seen significant advancements in recent years, particularly as mobile and embedded platforms become increasingly capable of executing complex models directly on-device [37]. This approach not only offers real-time detection capabilities but also addresses significant concerns related to data privacy, connectivity dependency, and system latency.

In contrast to conventional server-based solutions, on-device AI processes data locally, thereby eliminating the need to continuously transmit sensitive video streams or physiological data to external servers. This localized processing significantly enhances privacy and data security, a particularly important consideration when handling personal data such as facial expressions or behavioral cues during driving [38, 39].

For smartphone-based fatigue detection, several studies have demonstrated the feasibility of running lightweight AI models directly on-device, leveraging their computational resources without external processing. Fatigue detection systems often rely on rule-based algorithms, where predefined thresholds for eye closure duration, head nodding frequency, or yawning frequency are applied after AI models extract relevant facial features. Qiao et al. [40] developed a system that analyzes real-time facial features such as eye closure and head pose using smartphone cameras, achieving high detection accuracy (90%) on-device. Extending these approaches, Galarza et al. [30] proposed a smartphone-based system utilizing convolutional neural networks (CNN) for PERCLOS estimation under varying lighting conditions, demonstrating robust performance through adaptive thresholding and infrared enhancement. More recently, Jose et al. [41] introduced SleepyWheels, a lightweight model incorporating EfficientNetV2 and facial landmark detection to identify driver fatigue, demonstrating robustness to facial obstructions and feasibility for on-device execution on smartphones.

Beyond smartphones, embedded platforms such as Raspberry Pi and NVIDIA Jetson Nano have been increasingly used to deploy on-device fatigue detection due to their improved processing capabilities and modularity. For instance, Khunpisuth et al. [42] developed a drowsiness detection system using a Raspberry Pi 3 and Raspberry Pi Camera, combining face detection with Haar Cascade classifiers and template matching to accurately analyze eye blinking frequency and head tilting angles in real time. Similarly, Rahman et al. [2] leveraged the GPU-accelerated hardware of the NVIDIA Jetson Nano, system shown in Figure 2.5, to deploy facial fatigue detection models capable of efficiently processing high-resolution video streams, achieving both high inference speed and detection reliability.

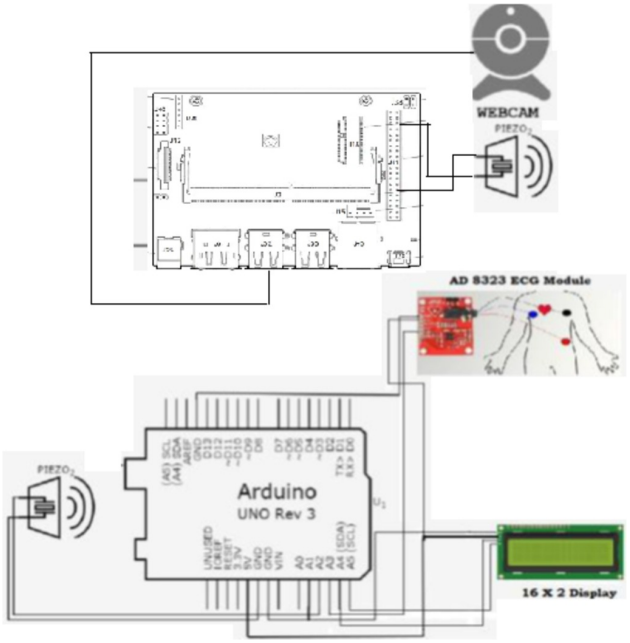


Figure 2.5: Schematic diagram of the complete embedded system [2].

To address the computational limitations of mobile and embedded platforms, some on-device implementations adopt combination approaches, where AI models are employed for feature extraction, such as facial landmark detection or eye closure percentage estimation, and rule-based thresholds are subsequently applied to infer fatigue states. This division allows for reduced model complexity while still achieving practical accuracy under resource constraints [40]. In addition, model compression techniques, such as pruning, quantization, and knowledge distillation, have been widely explored to further reduce the computational footprint while maintaining acceptable accuracy [43]. Moreover, frameworks like TensorFlow Lite and NVIDIA TensorRT enable highly optimized inference by taking advantage of hardware acceleration available on modern mobile and embedded devices.

Another significant advantage of on-device processing is the reduction of latency, which is critical for safety-critical applications like driver monitoring. By processing sensor data locally, these systems can provide immediate fatigue alerts without dependency on network connectivity or cloud server availability. Furthermore, by processing data locally and avoiding continuous data transmission, on-device systems inherently reduce communication overhead, contributing indirectly to energy efficiency, which is crucial for battery-operated mobile deployments [44].

In this dissertation, these principles are extended through the design and evaluation of a real-time fatigue detection system that runs entirely on a smartphone.

2.4 Challenges and Limitations of Current Approaches

Despite the notable advancements in driver fatigue detection systems, numerous challenges and limitations persist, particularly in real-world deployments. Environmental variability, such as

changes in lighting, weather, and road conditions, can impair the accuracy of monitoring systems, while computational demands and real-time performance requirements place additional strain on hardware capabilities and energy resources. Furthermore, issues such as sensor noise, network latency in cloud-based systems, and the integration of multimodal data contribute to the complexity of system reliability and scalability. This section addresses these challenges, emphasizing the influence of environmental factors, computational constraints, and real-time processing demands on these systems.

2.4.1 Environmental Variability

A significant challenge in the detection of driver fatigue is the impact of environmental variability. Factors such as lighting conditions, weather changes, and road surface irregularities often degrade the performance of monitoring systems, which affects their reliability and robustness in real-world settings.

Lighting conditions are a primary concern, especially in the context of image/vision-based methods. Variations in ambient light, such as glare, shadows, or low-light conditions, may obscure facial features and diminish the accuracy of visual detection systems. For instance, Abbas et al. in [45] highlight how lighting changes due to weather conditions, like rain or snow, can further complicate the recognition of subtle facial cues associated with fatigue. Such challenges may require the use of infrared imaging, which can increase the overall cost and complexity of these systems.

Another environmental factor is the road surface irregularities, such as potholes or uneven pavement, which introduce noise into the sensor data, particularly in systems relying on vehicle-based measures, such as steering wheel movements. For example, Li et al. in [21] demonstrate how these irregularities affect steering behavior, making it harder to distinguish between patterns caused by fatigue and those caused by rough terrain. This highlights the need for integrating multiple data sources to improve robustness in diverse driving environments.

In the context of physiological sensor data, the presence of background noise represents a significant challenge. In particular, ECG readings may be corrupted by vibrations or motion artifacts caused by the vehicle's movement. The authors of the study [27] addressed this issue by proposing noise-filtering algorithms for the data collected, which are capable of improving the clarity of ECG signals, as illustrated in Figure 2.6. Similarly, Munir et al. [28] emphasized the importance of combining HRV features with signal processing techniques, such as Fast Fourier Transform, to enhance detection reliability even under environmental interference.

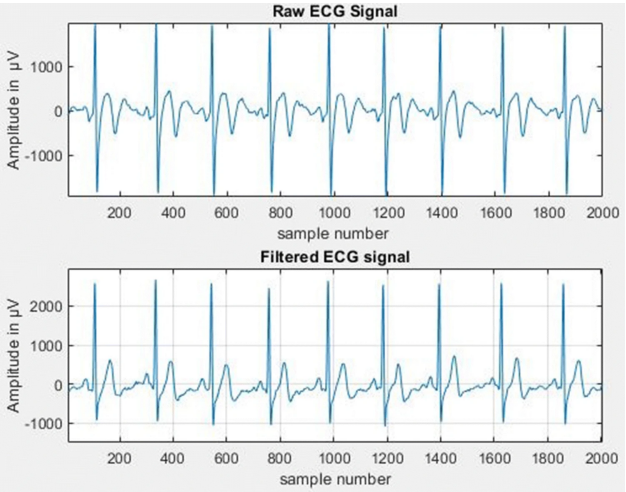


Figure 2.6: Raw and filtered signal of an ECG [27].

Another issue is the interference caused by driver accessories such as glasses or face masks, which can obstruct visual detection algorithms. For instance, Rahman et al. in [2] explored computer vision techniques that account for these obstructions, such as using EAR and mouth openness metrics, to ensure robust performance under such conditions. However, ensuring consistent performance across a diverse user base with varying accessory use remains an open challenge.

In summary, environmental variability imposes considerable constraints on the deployment of real-time driver monitoring systems. To overcome these challenges, it is required a combination of advanced data processing techniques, robust sensor designs, and adaptive algorithms is required to ensure consistent performance under varying conditions.

2.4.2 Real-Time Performance and Computational Load

The challenges of real-time performance and computational load in driver fatigue detection systems arise from the combination of hardware limitations, system latency, and algorithmic complexity. The reliability of data transmission between the vehicle and cloud servers is of critical importance in cloud-based detection systems. The high volume of video and sensor data requires significant bandwidth, which may be limited by factors such as network congestion, data packet losses, and transmission delays, especially in areas with poor network coverage. These limitations often render cloud-based solutions impractical for real-time applications, underscoring the importance of bandwidth-efficient transmission protocols and the adoption of edge computing paradigms [46].

Embedded systems, including the Raspberry Pi and Arduino, are commonly used in cost-sensitive applications. However, Jiang et al. [47] demonstrated that these devices are unable to execute computationally intensive deep learning models due to the limitations of their processing power and memory. Techniques such as model quantization and optimization can mitigate these

challenges, as exemplified by the Tensor Virtual Machine pipeline for efficient inference on devices like Raspberry Pi. Nonetheless, these optimizations may lead to some compromise in model accuracy.

Latency represents a significant obstacle to real-time monitoring, particularly in cloud-centric architectures. This challenge is highlighted in a comparative study from 2021 [45], which demonstrates that many existing hypovigilance detection systems suffer from poor response times, thereby reducing the effectiveness of the system in scenarios that require immediate intervention. Mobile Edge Computing represents a promising solution, as it brings computation closer to the data source and significantly reduces latency. However, the adoption remains in its early stages and may entail supplementary deployment costs and infrastructure updates [48].

Another remaining significant concern is energy efficiency, particularly for wearable and mobile platforms. The continuous operation of sensors, cameras, and processing units rapidly depletes battery life. While specific studies addressing energy efficiency in driver fatigue detection remain limited, general research on IoT edge computing highlights similar challenges. For instance, Chen et al [49] observed that although advances in energy-efficient components have been made, the integration of high-performance algorithms further aggravates this issue in resource-constrained devices. In order to improve energy efficiency, it is necessary to optimize software algorithms, leverage hardware acceleration, and/or apply energy-aware scheduling methods.

Finally, the integration of data from various sensors, such as video, physiological and behavioral inputs, improves the precision of detection systems; however, it collectively elevates the computational intricacy. As outlined by Du et al. [50], the integration of data from various sources presents several challenges, including the synchronization of heterogeneous data streams and the increased processing demands that this entails. To address these complexities, advanced algorithms such as multimodal approaches and adaptive execution strategies have been explored. Xu et al. [51] benchmarked end-to-end multimodal deep neural networks for hardware-software integration in autonomous systems. However, these algorithms require sophisticated hardware and substantial software development efforts [52]. This growing complexity further underscores the technical challenges involved in deploying real-time fatigue detection systems on embedded platforms.

2.5 Summary

This chapter provided a comprehensive review of the current state of the art in driver fatigue detection, focusing on the methodologies and technologies applied to monitor and assess driver alertness. Several approaches have been proposed over the years, encompassing subjective methods such as self-report scales, vehicle-based methods utilizing steering and lane deviation metrics, and physiological-based methods leveraging biomarkers such as EEG and ECG. Image-based methods, which analyze facial features such as eye closure, yawning, and head movements, have also become prominent due to their non-intrusive nature and the advancements in computer vision

and deep learning. Furthermore, hybrid methods integrating multiple data sources have been explored to enhance detection accuracy, particularly in real-world scenarios where single-method approaches may fall short.

The chapter also discussed the application of AI in fatigue detection, especially the shift toward on-device AI processing on mobile and embedded platforms such as smartphones and NVIDIA Jetson Nano. These platforms facilitate real-time detection while addressing concerns such as data privacy, latency, and system complexity. Despite the advancements, several challenges persists, including environmental variability, computational load, and energy efficiency remain critical barriers.

Given these considerations, the Chapter 3 defines the system requirements for the proof of concept application developed in this dissertation. It translates the insights gained from the literature into a concrete and practical specification tailored to a mobile-based driver fatigue detection system.

Chapter 3

System Analysis and Requirements

This chapter defines the system requirements for the driver fatigue detection proof of concept (PoC) application developed in this dissertation. The specification includes functional and non-functional requirements, as well as scope boundaries, assumptions, and constraints that shaped the design and implementation. Since this research follows a PoC approach, the goal was to establish a minimal, yet complete, feature set that enables the system's feasibility validation while limiting unnecessary complexity.

3.1 Scope Definition

To establish a clear and manageable foundation for the development, this section defines the scope boundaries of the proposed solution. It outlines what falls outside the system's objectives, the key assumptions adopted, and the technical constraints considered during design and implementation.

3.1.1 Out of Scope

In order to establish a well-defined and manageable scope, several functionalities were explicitly excluded from this dissertation. The system was developed exclusively for the Android platform, with no support for iOS. This decision was driven by the technical guidelines provided by Pink Room, the mentoring company, which recommended a native Android development approach to ensure optimal performance, maintainability, and integration with device-specific resources. Additionally, the system does not incorporate personalized driving recommendations, such as suggesting rest breaks or advising on corrective actions. Instead, the core objective is limited to detecting driver fatigue in real time and issuing corresponding alerts without prescribing behavioral interventions.

Furthermore, advanced customization options were deliberately excluded to minimize system complexity and potential driver distraction. The system does not allow users to modify notification types, personalize alert sounds, or configure alternative notification behaviors. Instead, a standardized alert mechanism is implemented to ensure consistency, ease of use, and immediate recognition during testing sessions. These scope exclusions contribute to maintaining a focused

development effort and support the timely delivery of a functional PoC prototype suitable for feasibility validation.

3.1.2 Assumptions

The successful implementation of the system relies on several foundational assumptions that define this dissertation's boundaries. The system operates on Android mobile devices equipped with a front-facing camera, gyroscope, and accelerometer, as these components are required to perform facial feature extraction and monitor device movement. It is assumed that the smartphone is mounted on a fixed holder inside the vehicle, ensuring stable alignment with the driver's face.

Furthermore, the system assumes unobstructed visibility of the driver's face during operation. Users are expected not to wear items that cover critical facial landmarks, such as opaque sunglasses or face masks, to ensure maximum fatigue detection accuracy. Sufficient ambient lighting is also assumed, ensuring that facial features are clearly captured by the camera. Extreme lighting conditions, such as excessive glare or insufficient cabin illumination, are not considered within the scope of this system. These assumptions help establish this dissertation's transparent and manageable scope and requirements, focusing on developing and testing within well-defined operational parameters.

3.1.3 Constraints

The system is developed under a set of technical and operational constraints that define the boundaries for its design and evaluation. The minimum Android platform version is set to Android 14.0. Although application store deployment is not a goal of this dissertation, selecting this version ensures alignment with current platform guidelines and reflects best development practices. Additionally, the application is implemented exclusively in Kotlin, following recommendations from Pink Room and Google's official recommendation for Android development, as Kotlin offers advantages in terms of code maintainability, readability, and integration with the Android ecosystem.

Furthermore, the system relies exclusively on built-in sensors available on standard mobile devices, including the front-facing camera, gyroscope, and accelerometer. No external or specialized hardware is used, as the focus is strictly on evaluating the feasibility of fatigue detection using widely accessible smartphone components. This constraint allows the system to be fully developed and tested within a controlled and reproducible environment, minimizing dependencies on proprietary or external devices.

3.2 Non-Functional Requirements

The non-functional requirements (NFR) define the essential quality attributes that characterize the system's performance and operational behavior under realistic conditions. These attributes were

selected to ensure that the system remains usable during operation, delivers timely fatigue detection alerts, supports model maintainability, and achieves sufficient detection accuracy. Each attribute is defined with measurable criteria to support objective evaluation throughout development and testing, and is described in detail in the following sections.

3.2.1 Usability

Usability ensures that the system can operate with minimal driver interaction, thereby reducing cognitive load and preventing additional distractions. Interactions related to initiating, pausing, or stopping monitoring must be straightforward, requiring minimal user effort. The system interface is designed to provide immediate feedback on any action performed, ensuring that the system state is always clearly communicated to the user. These usability principles are formalized in Table 3.1.

Table 3.1: Usability Specific Quality Attribute Scenario

ID	NFR1
Actor	User
Trigger	User wants to start, stop or restart monitoring
Environment	Normal runtime
Artifact	Application User Interface
Response	The system ensures that monitoring-related interactions are minimal and intuitive, reducing user effort and cognitive load during operation.
Response Measure	Users must be able to start, stop, or restart monitoring with a single action from key screens (Home, Monitoring, or Trip Summary). The interface should provide immediate visual feedback to confirm the state change, ensuring clarity and reducing interaction time.

3.2.2 Performance

Performance addresses the system's ability to detect and respond to fatigue events in real-time. As fatigue detection is a safety-critical and continuous process, the system must ensure minimal delay between face presence detection and alert notification, while also avoiding false positives and negatives. The detection pipeline is designed to process events and issue alerts within one second after a critical fatigue event occurs. Table 3.2 presents this quality attribute.

Table 3.2: Performance Quality Attribute Scenario

ID	NFR2
Actor	User
Trigger	A critical fatigue event is detected, such as prolonged eye closure
Environment	Normal operation
Artifact	Application
Response	The system triggers an alert within a maximum of 1 second from the moment the user exhibits a detectable fatigue indicator.
Response Measure	The alert must be displayed within 1 second after the fatigue event occurrence ($T_{alert} - T_{event} \leq 1s$), verified through timestamp logging and latency analysis.

3.2.3 Maintainability

Maintainability ensures that the AI model can be updated, replaced, or improved without requiring significant modifications to the core system architecture. The system follows a modular architecture where AI models follow a standardized input-output format, allowing seamless integration across different models. This approach follows AI engineering and Machine Learning Operations (MLOps) best practices, minimizing technical debt and ensuring framework scalability. Not only that, but by decoupling the AI model from business logic, the system supports plug-and-play AI updates, enabling continuous improvements without significant architectural modifications, as defined in Table 3.3.

Table 3.3: Maintainability Quality Attribute Scenario

ID	NFR3
Actor	Developer
Trigger	Replace or update the AI models
Environment	During development
Artifact	AI Processing Module
Response	The system allows seamless integration, replacement, or updates of the AI model without requiring extensive modifications to the application's core logic or architecture.
Response Measure	AI model updates require $\leq 20\%$ changes to non-model-specific code, preserving the functionality of other system components. Models must adhere to a standardized input-output format, and integration must be handled via configuration files rather than hardcoded logic.

3.2.4 Efficacy

Efficacy ensures that the AI model achieves sufficient detection accuracy under controlled test conditions. The system is evaluated using on-device AI running on mobile hardware, both on

labeled datasets and real-world tests, to verify its capability to detect fatigue states. By prioritizing efficacy, the system enhances safety through consistent and accurate fatigue detection, as defined in Table 3.4.

Table 3.4: Efficacy Quality Attribute Scenario

ID	NFR4
Actor	Application
Trigger	Process video feed and classify driver as "Fatigued" or "Awake"
Environment	Normal operation
Artifact	AI Processing Module
Response	The system classifies driver fatigue state accurately.
Response Measure	The model must achieve a mean F1-score of $\geq 85\%$, evaluated on labeled test datasets and real-time mobile testing, using precision and recall as evaluation metrics.

3.3 Functional Requirements

A total of seventeen functional requirements (FR) were defined to guide system development and ensure a structured and traceable implementation process. Each requirement follows a standardized set of specification fields, as summarized in Table 3.5. The complete detailed list of functional requirements is provided in Appendix A.

Table 3.6 describes all the functional requirements, and the detailed specification of each requirement is described in Appendix A.

Table 3.5: Requirements Specification Fields

Field	**Description**
Unique ID	A unique and easily readable identifier that allows precise reference to the requirement within the system specification.
Title	A formal and concise title describing the requirement.
Description	An optional field providing additional clarification in cases where the requirement may involve complex behavior or potential ambiguity.
Priority	The requirement's implementation priority, classified using the MoSCoW method: Must Have, Should Have, Could Have, Won't Have (where the "Won't Have" category is acknowledged but not included in the implemented system scope).
Estimation	An estimation of development effort based on a Fibonacci-like numerical sequence, used to support planning and workload distribution. Estimates above 13 were decomposed into smaller subtasks in order to improve manageability.
Dependency	A list of any prerequisite requirements that must be completed before implementing the current requirement.
Mockup Reference	A cross-reference to the associated application screen, supporting system traceability; the full set of mockups is provided in Appendix B.

Table 3.6: Functional Requirements

ID	**Title**	**Priority**	**Estimation**
FR1	Splash Screen	Must Have	2
FR2	Home Screen	Must Have	3
FR3	Display Live Front Camera Feed	Must Have	3
FR4	Display Facial Landmarks Overlay on Live Camera Feed	Should Have	3
FR5	Detect and Analyze Eye State	Must Have	8
FR6	Detect and Analyze Mouth Movements for Yawning	Must Have	8
FR7	Detect and Analyze Head Movements for Tilts	Must Have	13
FR8	Collect Battery, CPU, GPU, and RAM Usage Data Every Minute	Should Have	8
FR9	Allow User to Stop Monitoring	Must Have	2
FR10	Provide an Alert When Critical Fatigue Signs Are Detected	Must Have	8
FR11	Show a Summary of Detected Fatigue Events	Should Have	5
FR12	Record Video During Monitoring Session	Could Have	5
FR13	Export Monitoring Session Data	Should Have	5
FR14	About Screen	Should Have	3
FR15	Display Privacy Policy on the About Screen	Should Have	3
FR16	Display Terms & Conditions on the About Screen	Should Have	3
FR17	Enable Hands-Free Mode Using Voice Commands	Could Have	5

3.4 Summary

The present chapter defined the scope, assumptions, constraints, and system requirements that guided the development of the driver fatigue detection PoC application. In order to ensure a focused, measurable, and technically feasible implementation, both functional and non-functional requirements were specified. The subsequent chapter, Chapter 4, presents the system architecture and describes the design decisions that enable real-time fatigue detection using on-device sensors and AI.

Chapter 4

System Architecture and Development

4.1 Architectural Design

The design and implementation of the proof of concept (PoC) application developed within this work is grounded on an architecture that prioritizes modularity, scalability, maintainability, and real-time performance in the detection of driver fatigue. This architectural solution was derived directly from the functional and non-functional requirements previously defined in Chapter 3, which established strict guidelines on system responsiveness, privacy preservation, and deployment feasibility on mobile platforms.

4.1.1 Architectural Foundation

The application follows a layered architecture primarily based on the Model-View-ViewModel (MVVM) architectural pattern, extended with principles inspired by the Model-View-Intent (MVI) paradigm. This hybrid approach ensures a clear separation of responsibilities across components, promotes unidirectional data flow (UDF), and simplifies state management by enforcing centralized control over state transitions.

Within this architecture, the User Interface (UI) layer passively observes changes in the application state, which is exposed by the ViewModel component as a single immutable object. User actions, as well as system events, are translated into discrete events that are dispatched through a single event-handling function, responsible for updating the state in response to each interaction. This unidirectional communication flow, displayed in Figure 4.1, ensures deterministic behavior, simplifies debugging, and facilitates future extensibility.

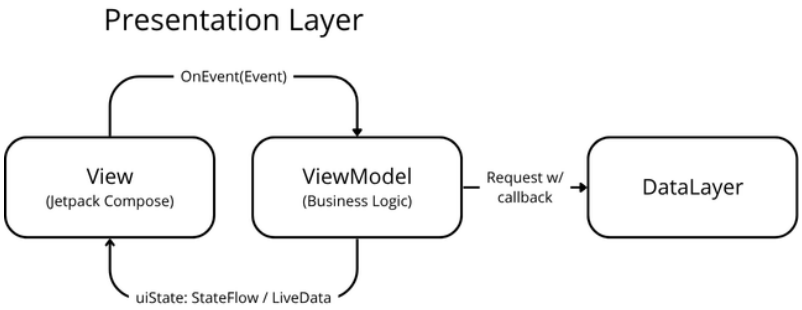


Figure 4.1: Unidirectional Communication Architecture

4.1.2 Layered Architecture

The application architecture is structured following the guidelines of the Modern Android Application Architecture, Google's recommended architecture, which emphasizes strict separation between the presentation and data layers [53]. Additionally, a shared package is used to centralize utility functions, constants, and common components that are used across layers, avoiding redundant code and promoting clarity in codebase organization. As the business logic implemented did not introduce enough complexity to warrant additional abstraction, no explicit domain layer was required, deviating from the traditional Clean Architecture approach, as proposed by Martin [54].

4.1.2.1 Presentation Layer

The presentation layer is responsible for rendering the user interface and orchestrating all interactions with the end-user. The structure of this layer, implemented using Jetpack Compose, is organized into distinct packages, where each screen corresponds to a functional module containing: (i) the composable screen function that defines the UI components, (ii) the ViewModel that maintains state and processes events, and (iii) strongly-typed definitions of all possible UI states and user events. Additional feature packages, such as about, home, and session_summary, follow similar structural conventions. The overall folder structure of the presentation layer is illustrated in Figure 4.2.

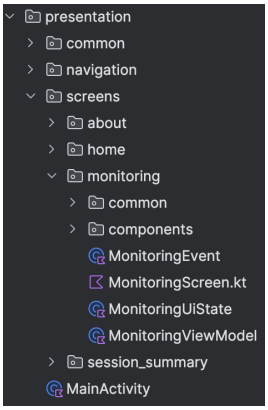


Figure 4.2: Presentation Layer Package Structure

4.1.2.2 Data Layer

The data layer is responsible for acquiring sensor data, processing it, invoking the AI models, and logging fatigue-related events. Its internal structure is modularized into multiple repositories, each dedicated to the processing of specific behavioral indicators such as blinking, yawning, and head tilt. A central repository coordinates these subcomponents, aggregating the results and managing fatigue event logging, as illustrated in Figure 4.3.

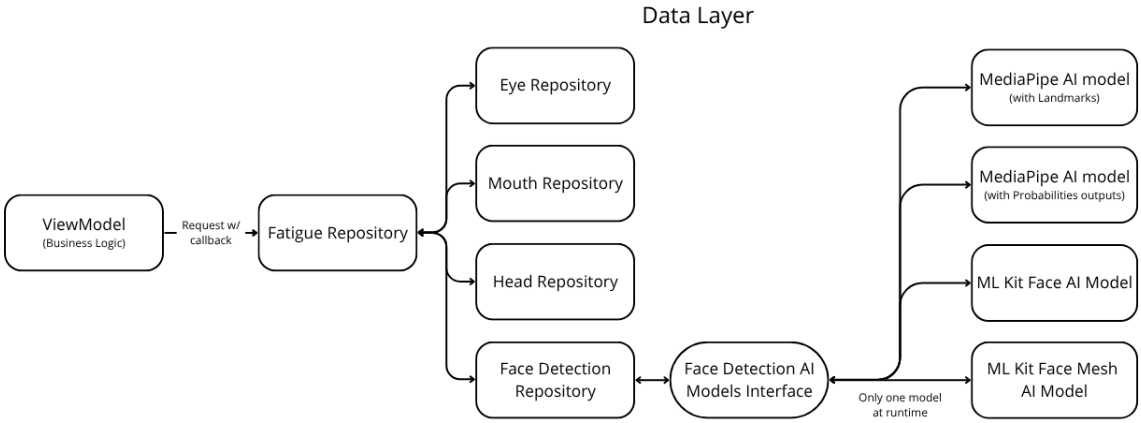


Figure 4.3: Repository-Service Dependency Diagram

The facial detection subsystem is encapsulated within a dedicated repository that exposes a unified interface to the different facial landmark detection services. This architectural abstraction enables different AI models to be integrated or replaced without affecting higher-level application logic or user interface components.

4.1.3 Shared Components and Dependency Management

Although the architecture does not define a formal shared layer, several utility components are designed for cross-layer reuse. These include mathematical functions for geometric computations, image processing utilities for facial landmark transformations, and constant definitions of configuration parameters and threshold values used throughout the detection pipeline. Additionally, the application employs dependency injection using the Hilt framework to manage object lifecycles and promote loose coupling between components. This approach supports compliance with the Dependency Inversion Principle, simplifying unit testing and future system modifications by abstracting service instantiation and dependency resolution.

4.1.4 Development Environment

The PoC application was developed as a native Android application using Android Studio as the integrated development environment and Kotlin as the primary programming language. The user interface was implemented fully in Jetpack Compose, allowing declarative UI development that integrates directly with the system's state management architecture. All AI inference is performed

entirely on-device, with no reliance on external servers or cloud processing, ensuring privacy preservation, minimizing latency, and maintaining operational stability even under unstable network conditions. The system integrates three different AI models: MediaPipe Face Landmark Detection model[55] and ML Kit Face and Face Mesh Detection models[56, 57]. These models are optimized for mobile hardware, enabling real-time facial landmark extraction and fatigue detection while preserving responsiveness and controlling resource utilization.

4.2 Core Implementation Components

The functional core of the system integrates multi-sensor acquisition modules with real-time AI inference pipelines to support driver fatigue detection. This section details the principal data flows, processing stages, and decision logic that collectively support the system's ability to extract, process, and evaluate behavioral fatigue indicators in real-time.

4.2.1 Sensor Integration and On-Device Inference

To capture the required facial features for fatigue detection, the system primarily relies on video-based facial landmark extraction obtained from the mobile device's front-facing camera. Image frames are continuously acquired using the Android CameraX Application Programming Interface (API), which streams video data through a callback interface optimized for real-time processing. Frames are sampled at 24 frames per second, following both industry standards for cinematic video and previous literature [58, 59], providing a sufficient temporal resolution for fatigue detection while balancing computational load and thermal constraints on mobile devices.

Each acquired frame is subsequently processed by the facial landmark detection module, which allows for dynamic backend selection between MediaPipe and ML Kit, depending on runtime configuration. These AI models provide either sets of facial keypoint coordinates or probabilistic outputs related to fatigue indicators, including eye closure and mouth opening likelihoods.

In parallel to visual data acquisition, the system continuously collects inertial measurements from the accelerometer and gyroscope using the Android Sensor API. These motion signals are pre-processed in the data layer to estimate the real-time orientation of the mobile device relative to gravity. Such normalization compensates for mounting variability and ensures that head posture estimations reflect genuine driver behavior rather than artifacts caused by device positioning inconsistencies. The full sensor and inference pipeline is schematically represented in Figure 4.4.

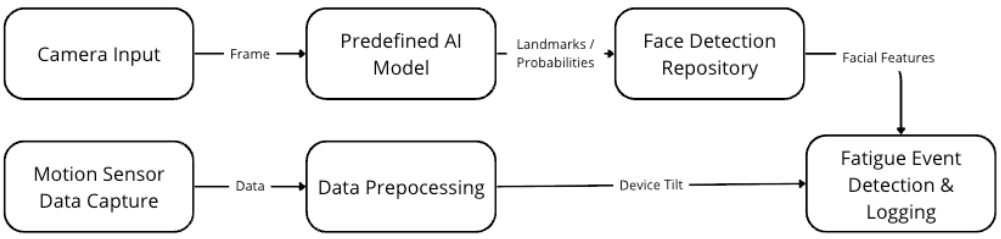


Figure 4.4: Data Flow Diagram for Sensor and Inference Pipeline

4.2.2 Local Persistence and Data Export

All fatigue events detected during a monitoring session are persistently logged locally on the device to support subsequent performance evaluation and system validation. Each logged entry includes the fatigue event category, such as blink, yawn, head tilt, and its associated timestamp, enabling chronological reconstruction of detected behavior.

In addition to behavioral event logging, the system continuously records resource utilization metrics related to system performance, such as Central Processing Unit (CPU) usage, memory allocation, Graphics Processing Unit (GPU) load, and battery consumption. These system-level metrics are sampled at fixed three-second intervals and are stored alongside the behavioral event logs to facilitate resource profiling, efficiency analysis, and computational feasibility assessments during experimental evaluations.

Upon session completion, users are offered the option to export session logs for offline post-processing. Exported files adopt a structured format, exclusively containing non-identifiable metadata such as event timestamps, event types, and system resource metrics. No raw biometric images or personally sensitive data are stored or transmitted, thereby ensuring full compliance with privacy and data protection principles applicable to real-world deployments.

4.2.3 Rule-Based Logic

The Fatigue Repository integrates multiple facial analysis metrics, from the coordinates and probabilities extracted in real time by the AI models, and organizes them into a unified processing pipeline that enables fatigue assessment through a rule-based approach. This framework allows the system to operate based on transparent and explainable criteria, ensuring traceability and reproducibility of the detection outcomes.

The MediaPipe Face Landmarker and ML Kit Face Mesh models provide a comprehensive facial mesh with 468 landmarks, illustrated in Figure 4.5, which offers a dense and detailed representation of the face.

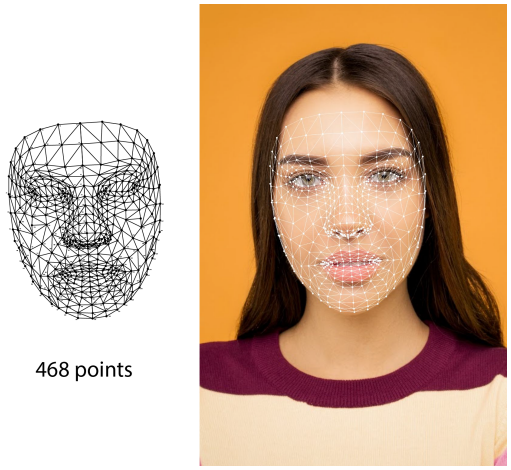
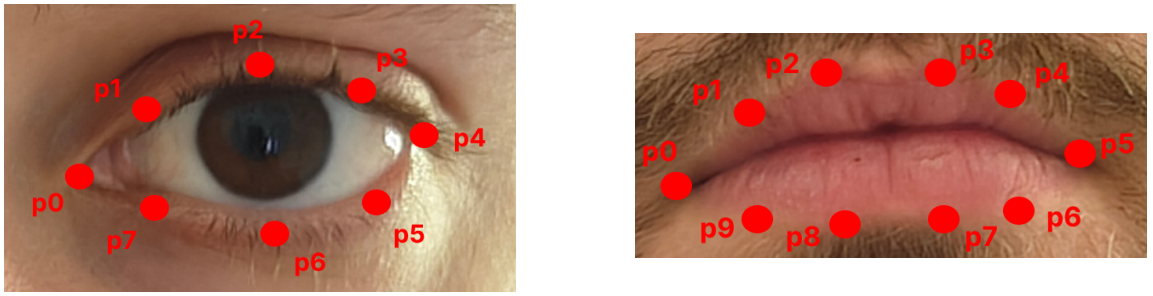


Figure 4.5: Example of facial landmark mesh (468 points) used for feature extraction [57]

In contrast, the ML Kit Face model returns a reduced set of 133 facial landmarks. However, it includes the specific landmark indices required to compute the EAR and MAR metrics, making it fully compatible with the feature extraction formulas.

The extracted landmark coordinates serve as the basis for deriving multiple quantitative fatigue indicators, described below and illustrated in Figure 4.6.



(a) Landmarks used for EAR calculation

(b) Landmarks used for MAR calculation

Figure 4.6: Facial landmarks used to extract EAR and MAR.

Eye Aspect Ratio (EAR)

The EAR is used to detect eye closure by measuring the vertical distance between the upper and lower eyelids relative to the horizontal distance of the eye, as shown in Equation 4.1. A blink is detected when the EAR drops below the predefined threshold (around 0.25) and remains below this value for a short duration.

$$EAR = \frac{\|p_1 - p_7\| + \|p_2 - p_6\| + \|p_3 - p_5\|}{3 \cdot \|p_0 - p_4\|} \tag{4.1}$$

Mouth Aspect Ratio (MAR)

The MAR metric evaluates mouth openness, which is critical for yawning detection, and is calculated using the formula defined in Equation 4.2. A yawning event is identified when the MAR exceeds a predefined threshold and remains high for more than 1.5 seconds.

$$MAR = \frac{\|p_1 - p_9\| + \|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{4 \cdot \|p_0 - p_5\|} \tag{4.2}$$

Percentage of Eye Closure (PERCLOS)

The PERCLOS metric quantifies the proportion of time the eyes are closed within a specified window, as described in Equation 4.3:

$$PERCLOS = \frac{\text{Closed Eye Frames}}{\text{Total Frames}} \tag{4.3}$$

Pitch (Head Tilt Analysis)

The head tilt is estimated by calculating the pitch angle between the nose tip and chin facial landmarks extracted from the facial mesh. Let (x_{nose}, y_{nose}) and (x_{chin}, y_{chin}) denote the pixel coordinates of the nose tip and chin, respectively, after scaling to match the resolution of the video frame. The raw pitch angle is computed as shown in Equation 4.4:

$$\theta_{pitch} = \arctan 2(y_{nose} - y_{chin}, x_{nose} - x_{chin}) \tag{4.4}$$

This value is converted to degrees and corrected for smartphone orientation by subtracting the device tilt angle (θ_{device_tilt}), resulting in the compensated pitch angle presented in Equation 4.5:

$$\theta_{pitch_corrected} = \theta_{pitch} - \theta_{device\_tilt} \tag{4.5}$$

A head tilt event is detected when $\theta_{pitch_corrected}$ exceeds 15 degrees for at least 2 seconds.

Probability-Based Facial Features

In addition to geometric landmark analysis, the system can leverage probabilistic outputs provided by certain face detection models to enhance robustness in challenging conditions, such as partial occlusions or extreme head poses.

- **MediaPipe Face Landmarker** provides both facial landmarks and, optionally, eye closure and mouth openness probabilities (ranging from 0.0 to 1.0). When enabled, these probabilities can directly replace computed EAR and MAR values for blink and yawn detection.
- **ML Kit Face Detection** provides probabilistic estimates for eye openness and the head pitch angle. However, as it does not include mouth-related probability outputs, the model's reduced set of facial landmarks (133 facial keypoints) is leveraged to compute the MAR, thereby enabling yawn detection through geometric analysis.

These models enable the system to switch between geometry- and probability-based indicators, allowing comparative evaluation of performance and resource efficiency across models and devices.

4.2.4 Rule-Based System

The rule-based system integrates the extracted facial metrics with threshold values defined in prior literature and was subsequently validated in this study using testing subsets of the selected datasets, explored in Chapter 5. These thresholds, summarized in Table 4.1, remained constant throughout the evaluation to ensure consistent behavior across different subject profiles.

Table 4.1: Fatigue Detection Thresholds from Literature

Metric	Threshold / Condition	Reference
Eye closure	< 0.25 for ≥ 10 consecutive frames (Frames per second = 24; $24 \cdot 0.4 = 10$)	Prasitsupparote et al. 2023 [58]
PERCLOS	$> 70\%$ over 1 minute	Arefnezhad et al. 2022 [60]
Blink Rate	< 8 or > 20 per minute	Karar et al. 2023 [61]
Yawns	≥ 2 in 10 sec or ≥ 4 in 1 min	Kashevnik et al. 2019 [38], Walizad et al. 2022 [62]
Head Nods	≥ 4 (>15°) in 2 minutes	Kashevnik et al. 2019 [38]

4.3 Design Principles and Modularity

While the previous sections have detailed the architectural structure and functional implementation of the system, this section discusses the underlying design principles that guided its development.

4.3.1 Modularity in AI Model Integration

A central architectural decision involved abstracting AI model integrations behind unified service interfaces to decouple model-specific logic from higher-level application components. In particular, the face detection repository exposes a common interface that encapsulates all interactions with facial landmark extraction backends, including both MediaPipe and ML Kit implementations. This abstraction promotes a clean separation of concerns, allowing new AI models to be integrated, substituted, or updated with minimal disruption to the surrounding system architecture.

This architecture highlights the decoupling of AI model-specific implementations from the system's presentation layer. This modular design is particularly advantageous in research contexts, where alternative AI models can be benchmarked with minimal integration effort. In production environments, it enables controlled model updates without introducing unintended side effects in unrelated system components.

4.3.2 Modularity in Feature Extraction Pipelines

The modular design philosophy extends beyond AI model integration to the system's internal feature extraction pipelines. Within the data layer, individual repositories are responsible for processing each behavioral fatigue indicator, such as eye closure, yawning, or head posture estimation. This separation allows new behavioral indicators to be incorporated with minimal architectural disruption.

Adding new features requires the development of specialized repositories that encapsulate the extraction logic and associated decision functions. These repositories are subsequently integrated into the central fatigue repository, which aggregates outputs from all available indicators. Minimal

updates to ViewModels, state definitions, or user interface components may be necessary when new indicators are intended for in-screen monitoring.

4.4 Summary

This chapter presented the system architecture and implementation of the developed proof of concept mobile application for real-time driver fatigue detection. The solution adopts a hybrid MVVM-MVI architectural pattern that ensures clear separation of concerns, centralized state management, and maintainability. Sensor integration combines video-based facial landmark extraction with inertial data normalization, while all AI inference is performed entirely on-device to guarantee privacy and operational autonomy. Fatigue detection is driven by a rule-based logic combining geometric and probabilistic facial metrics, calibrated through literature-based thresholds and experimentally validated. The modular architecture further supports the seamless integration of alternative AI models and facilitates the scalable extension of fatigue detection features, establishing a flexible foundation for the experimental evaluations described in the following chapters.

The evaluation of this architecture and detection pipeline is addressed in Chapter 5, where the datasets, testing procedures, and performance metrics are presented in detail.

Chapter 5

Experimental Methodology and System Validation

This chapter describes the methodology adopted to validate the proposed fatigue detection system. It outlines the data-driven design process and details the experimental procedures used to evaluate the system both in dataset-based and real-world scenarios.

5.1 Data-Driven Design and Pre- and Post-processing

The system's evaluation process required a structured pipeline for handling data before and after inference. This section presents the datasets used, along with the data exploration, cleaning, and annotation steps necessary to align them with the system's fatigue detection logic.

5.1.1 Selected Datasets

Three publicly available research datasets were used to validate the fatigue detection system, each contributing specific fatigue-related indicators.

Driver Monitoring Dataset (DMD)

The DMD [63] contains extensive data on driver fatigue-related activities captured through optical (color), infrared, and depth videos. The dataset is organized into six recording sessions (named from "S1" to "S6"), each focusing on specific behaviors and scenarios. However, only S5 includes relevant data for fatigue detection, featuring activities such as sleepy driving, yawning (with and without hand), and microsleep. The other sessions primarily address distraction, including texting and talking on the phone, making them unsuitable for the dissertation' objectives.

Yawning Detection Dataset (YawDD)

Although the YawDD [64] is not specifically a video dataset for drowsiness detection, it played a crucial role in yawning detection. The dataset consists of 322 videos captured using a camera

positioned under the front mirror, plus 29 dashboard-mounted videos that were excluded due to their lack of annotation. The dataset documents drivers in three main scenarios: everyday driving (normal), talking and yawning, and yawning while driving under realistic and varying lighting conditions.

National Tsing-Hua University Drowsy Driving Dataset (NTHU-DDD)

The NTHU-DDD [65] includes approximately 9.5 hours of video footage featuring 36 participants of various ethnicities, recorded under a range of simulated driving scenarios. These scenarios incorporate drowsy and non-drowsy behaviors, including yawning, slow blinking, nodding off, laughing, and everyday driving, captured under day and night lighting conditions. The dataset offers diverse conditions with participants wearing glasses, sunglasses, or no eyewear and involves five scenarios: *BareFace*, *Glasses*, *Night_BareFace*, *Night_Glasses*, and *Sunglasses*.

The selected datasets provide diverse video samples capturing different fatigue-related behaviors under various conditions. However, due to differences in recording settings, participant demographics, and annotation methodologies, a data cleaning and preprocessing step was necessary to ensure consistency and usability for the fatigue detection system.

5.1.2 Data Exploration and Cleaning

This process was essential in transforming raw datasets into usable samples. Given the datasets' heterogeneity, each required tailored cleaning strategies to ensure data quality, relevance, and consistency.

From the DMD, not all of the available samples were selected. For each of the 37 participants, only one of the recording sessions (S5) contained fatigue-related behaviors. The selection process was guided by file naming conventions (S1, S2, S3, S4, S5, S6), which facilitated the extraction of relevant samples while excluding sessions focused on distraction behaviors.

Although YawDD was not explicitly designed for fatigue detection, it was valuable for yawning recognition evaluation. The dataset included videos recorded with a camera under the front mirror and dashboard-mounted videos, but these were excluded due to the lack of annotation.

From the NTHU-DDD dataset were selected the videos from the dataset's testing subset. As infrared recordings were outside the scope of this dissertation, only the visible-spectrum videos were included and considered for analysis. After this filtering step, a total of 42 videos were selected, each with an average duration of approximately six minutes.

Following the data cleaning process, ensuring that the selected samples were accurately labeled to support reliable fatigue detection was essential. Each dataset used distinct annotation methodologies, requiring alignment and verification to maintain consistency across different sources.

5.1.3 Data Annotation

Following the cleaning and selection process, the remaining samples were reviewed to ensure that their annotations were consistent with the fatigue indicators under analysis. As each dataset had distinct annotation schemes, further verification was performed prior to quantitative evaluation. For instance, the DMD dataset used frame-by-frame annotation, capturing detailed facial actions related to eye state and yawning behaviors. The annotations distinguished between open and closed eyes and transitional states such as opening and closing. They also identified blinking events and yawning with and without hand coverage, providing a valuable level of detail for detecting signs of driver fatigue.

Similarly, the NTHU-DDD dataset included frame-level annotations indicating whether the subject appeared fatigued or alert. These labels were provided as plain text files, with each character representing a single video frame: ’1’ denoting a fatigued state and ’0’ indicating alertness. This annotation format facilitated direct alignment between the extracted image frames and their corresponding labels during evaluation.

In contrast, the YawDD dataset used video-level annotation, where each video was assigned a single label describing the driver’s overall state throughout the recording. Each video was labeled one of the following class labels: "yawning", "normal", "talking", or "talking & yawning". To ensure consistency, all "talking & yawning" instances were reclassified as "yawning" in the preprocessing step, maintaining uniformity in fatigue detection. Additionally, since the goal was to assess yawning-specific behaviors, videos labeled solely as ’talking’ were excluded due to their non-fatigue-related nature. To better understand the labeling, Figure 5.1 shows the internal folder structure of the dataset.

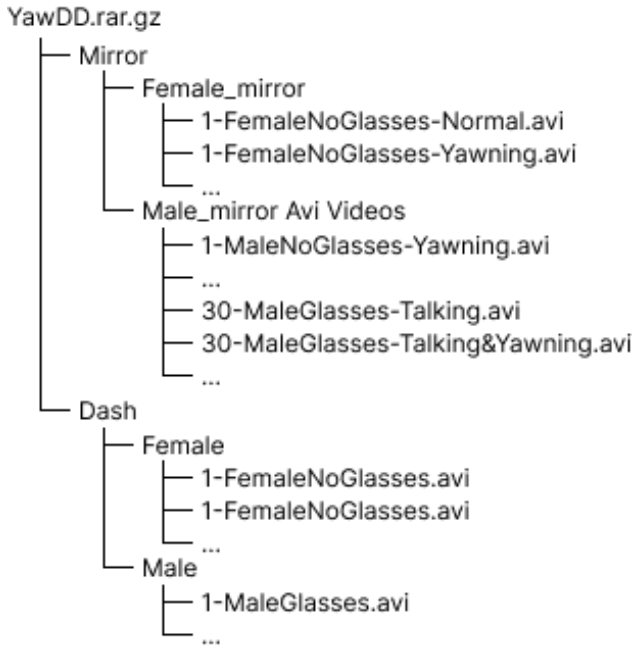


Figure 5.1: Folder structure of the YawDD dataset.

5.2 Experimental Validation (On-device)

To assess the practical performance and operational reliability of the developed fatigue detection system, a two-stage experimental validation procedure was conducted. First, controlled dataset-based testing was performed by replaying annotated video sequences through the full inference pipeline, allowing for quantitative evaluation against ground truth labels. Subsequently, real-world validation was performed by executing the proof of concept (PoC) mobile application inside a stationary vehicle while simulating realistic driving fatigue scenarios, allowing system behavior to be assessed under semi-realistic operational conditions, including variable lighting and driver positioning. Both testing formats were then analysed using the following evaluation metrics.

Confusion Matrix

The confusion matrix is a widely used diagnostic tool that offers a structured overview of a system's classification performance. Rather than reducing performance to a single metric, it organizes predictions into four categories: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). In the specific context of this dissertation, a TP corresponds to a correctly identified fatigue event, while a FP represents an alert state incorrectly flagged as fatigue. On the other hand, a FN reflects a missed fatigue event that went undetected by the system, and a TN denotes a correct classification of an alert (non-fatigued) state.

This matrix structure facilitates the identification of error patterns and detection inconsistencies, such as the misclassification of frequent blinking events as prolonged eye closures. The structure of the confusion matrix is shown in Equation 5.1.

$$\text{Confusion Matrix} = \begin{bmatrix} \text{TP} & \text{FN} \\ \text{FP} & \text{TN} \end{bmatrix} \tag{5.1}$$

Accuracy Score

The accuracy score measures the overall correctness of the system by evaluating the ratio of correctly predicted instances to the total number of instances. It provided a general performance overview, particularly in balanced datasets, and it is defined in Equation 5.2.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{5.2}$$

Precision and Recall

Precision and recall offered a more nuanced view of the system's detection capabilities:

- **Precision** (Positive Predictive Value) evaluates the proportion of true positive predictions out of all positive predictions, focusing on minimizing false positives, as defined in Equation 5.3.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{5.3}$$

- **Recall** (Sensitivity or True Positive Rate) assesses the system's ability to correctly identify fatigued states, emphasizing the reduction of false negatives, and it is given by Equation 5.4.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.4}$$

F1 Score

The F1 Score combines precision and recall into a single metric, offering a balanced measure of the system's accuracy, particularly in cases of class imbalance. It is defined as the harmonic mean of precision and recall, as presented in Equation 5.5.

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.5}$$

The performance metrics described above demonstrate the rule-based system's ability to accurately detect driver fatigue across the conducted experiments, which the corresponding validation procedures are detailed in the following subsections.

5.2.1 Dataset-Based Testing

Following the data exploration, cleaning, and annotation steps described in the previous section, the dataset-based testing phase was performed to systematically evaluate the fatigue detection pipeline under the annotated datasets. All evaluations were conducted fully on-device using a Realme GT 6 smartphone, equipped with a Qualcomm Snapdragon 8s Gen3 processor and 16 gigabytes (GB) of random access memory (RAM). The testing procedure involved three consecutive stages: data transfer and application configuration, on-device inference execution, and post-processing of the obtained outputs.

In the first stage, the pre-processed datasets were converted into sequences of individual image frames, which were transferred to the mobile device for evaluation. The PoC mobile application was temporarily adapted to operate without the use of the camera, disabling the standard CameraX-based video acquisition pipeline. Instead of capturing frames from the front-facing camera, the application was configured to sequentially ingest the dataset frames as pre-loaded image files, simulating the frame acquisition loop.

In the second stage, the full fatigue detection inference pipeline was executed on-device for each dataset sample. This pipeline included facial landmark extraction, feature computation, rule-based fatigue indicator evaluation, and event logging, identical to the processing applied during real-world monitoring sessions. The entire inference workflow was executed entirely on-device, without any external computation or server-side processing.

In the final stage, upon completion of the dataset inference runs, the application exported the fatigue event logs and performance metrics in a comma-separated values (CSV) file. These files contained per-performance-metric and per-fatigue-event data, including rule-based fatigue detections, feature values (EAR, MAR, and head pitch), and system resource utilization metrics.

The CSV logs file was subsequently transferred back to a workstation for post-processing and quantitative analysis, as described in Chapter 6.

5.2.2 Real-World Testing

To complement the dataset-based evaluation, a real-world testing procedure was conducted to assess the behavior of the PoC mobile application under semi-controlled operational conditions. The main objective of this validation step was to examine the system's ability to detect fatigue-related behaviors from real-time camera input, while confirming the feasibility and responsiveness of on-device inference during uninterrupted execution. Testing was carried out in a parked vehicle, positioned in an outdoor environment to replicate natural lighting variations and minimize the already existing environmental control. Although the car remained stationary throughout the tests, this setup allowed for realistic variations in ambient light and driver posture without introducing additional safety concerns.

The experimental configuration consisted of two Android smartphones mounted on the vehicle's dashboard using air vent phone holders positioned on the left and right sides of the steering wheel, respectively labeled as **B** and **A** in Figure 5.2. The left holder **(B)** supported a Google Pixel XL (model year 2016), equipped with a Snapdragon 821 processor and 4 GB of RAM. The right holder **(A)** supported a Realme GT 6 (model year 2024), the same device used during the dataset-based testing phase, featuring a Qualcomm Snapdragon 8s Gen3 processor and 16 GB of RAM. The Pixel XL was intentionally selected to represent a legacy, resource-constrained mobile platform, while also providing access to embedded motion sensors such as a gyroscope and accelerometer. In contrast, the Realme GT 6 illustrated the capabilities of modern hardware. This heterogeneous device setup allowed for direct comparison of performance across devices with distinct computational capabilities.

Figure 5.2: In-vehicle experimental setup used for real-world testing

In addition to these two evaluation devices, a third smartphone **(C)**, an iPhone 14 Pro, was mounted in the central console area and used exclusively to record high-definition footage of the participant's face during testing. This video recording was used as the ground truth for annotation and subsequent alignment of detected fatigue events.

The tests were conducted with two participants, one without glasses and another wearing transparent eyeglasses, to assess how facial occlusions and light reflections might affect detection accuracy. Before each session, the observer (the author) read a standardized introduction explaining the test's purpose, the roles of the three mobile devices, and the actions participants would be asked to perform. The test then proceeded according to a predefined action script, also read aloud by the observer, containing timed instructions for simulating fatigue-related behaviors such as eye closure, blinking, yawning, and head tilts. Both the introductory explanation and the full action script are present in Appendix C.

To accommodate real-world variability, in cases where the system did not immediately trigger a fatigue alert, the participant was instructed to repeat the expected behaviour until the condition was fulfilled. As a result, alerts generated within the same interaction block were counted as true positives, reflecting realistic user behavior in mobile contexts.

During testing, each Android device independently executed the fatigue detection system using the same models and similar software environment previously used in the dataset-based evaluation. For both participants, two test sessions were conducted per AI model, totaling eight sessions per device. The tested models included the MediaPipe Face Landmarker, evaluated both with and without the probability outputs enabled, as well as the ML Kit Face Mesh and ML Kit Face Detection models. Before conducting the full evaluation, preliminary differences in inference speed were noted: the ML Kit models demonstrated to be slighly faster processing than the MediaPipe variants, and the Realme GT 6 achieved end-to-end inference times around 50ms faster than the Google Pixel XL.

To ensure that the real-world testing sessions accurately reflected the expected performance of a production-ready application, the mobile application was built using a release configuration. This included enabling several optimization techniques commonly used in mobile deployment: resource minification, which removes unused UI layouts and assets to reduce application size; compressed assets, where media and static resources are optimized to lower loading times and memory usage; and code shrinking, using tools to eliminate unused classes and methods while obfuscating the final bytecode. These measures ensured that the application was evaluated under realistic execution conditions, minimizing overhead introduced by debugging tools or unnecessary code paths that are typically present in development builds.

During each session, the mobile application recorded system performance metrics, including CPU load, GPU usage, RAM consumption, and battery level. While device temperature was not quantitatively logged, it was qualitatively monitored throughout the evaluation to assess thermal behavior. At the end of each session, the application exported timestamped fatigue events and performance logs, following the same procedure established during dataset-based testing.

To assess the accuracy of the detected events, initial efforts involved the use of formal behavioral annotation software, specifically Behavioral Observation Research Interactive Software [66]. However, the complex nature of this tool rendered it impractical for the scenario at hand. Manual cross-referencing with the recorded video footage was therefore adopted to validate the occurrence and timing of the fatigue indicators and events detected, or not detected, by the system. Overall, this real-world testing phase enabled both the verification of detection reliability and the assessment of feasibility for mobile deployment across heterogeneous hardware configurations.

5.3 Summary

This chapter detailed the experimental methodology adopted to evaluate the developed fatigue detection system. Publicly available research datasets were selected and pre-processed to ensure consistency with the system's rule-based fatigue indicators, supported by additional data cleaning and annotation. Both dataset-based testing and real-world validation were performed using the PoC mobile application, enabling a controlled evaluation of the system's functionality and preparation for quantitative performance assessment, which results and discussions are presented in Chapter 6.

Chapter 6

Results and Discussion

6.1 Dataset-Based Evaluation Results

This section presents the results and discussions of the evaluation of the fatigue detection system developed, using three publicly available research datasets: DMD, YawDD, and NTHU-DDD. Each dataset was processed using four face detection models: the MediaPipe Face Landmarker (evaluated both with and without probability outputs), the ML Kit Face Mesh, and the ML Kit Face Detection model. For each dataset, the fatigue detection pipeline was executed on-device, and the outputs were compared against the provided ground truth annotations to assess the system's performance.

6.1.1 Results on DMD Dataset

The DMD dataset provides frame-level annotations for both blinking and yawning, allowing for independent evaluation of these two fatigue-related indicators. For each frame, relevant facial features were extracted, either as geometric ratios (EAR and MAR) or as model-generated probabilities. These features were then processed by a rule-based system to detect the presence of fatigue indicators, which were subsequently aligned with the corresponding ground truth annotations. The detection performance for each model and indicator is summarized in Table 6.1.

Table 6.1: Performance Metrics Summary in DMD

Tool	Approach	Indicator	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
MediaPipe Face Landmarker	Landmarks	Blink	95	95	85	90
		Yawn	87	87	71	78
MediaPipe Face Landmarker	Probabilities	Blink	98	98	89	93
		Yawn	91	91	82	86
ML Kit Face Mesh	Landmarks	Blink	92	92	88	90
		Yawn	93	93	71	81
ML Kit Face Detection	Probabilities	Blink	86	86	89	88
	Landmarks (*)	Yawn	87	87	65	74

(*) Yawn detection in the ML Kit Face Detection model was based on MAR computed from landmarks, as the model does not provide mouth openness probabilities.

Discussion

The MediaPipe Face Landmarker with probability outputs demonstrated strong reliability in identifying blink events, reducing incorrect detections in eye-open frames. Despite this, sometimes it missed subtler blinks that the landmark-only version was able to capture more consistently. This suggests that while probabilistic outputs help refine detection, they may filter out borderline cases. The ML Kit Face Mesh performed robustly overall, especially in blink detection, but its ability to detect yawns was less consistent, with several short-duration yawns not being flagged. As for the ML Kit Face Detection model, blink detection was based on the eye state probabilities provided by the model. While generally effective, it was less reliable for subtle eyelid movements compared to the MediaPipe model. Since no mouth openness probabilities are available, yawn detection relied on MAR computed from landmarks. Despite the limited landmark set, this approach worked reasonably well and achieved results comparable to those of the MediaPipe model (with landmarks).

6.1.2 Results on YawDD Dataset

The YawDD dataset focuses on yawning behavior and provides annotations at the video level, if there are yawns or not. Therefore, the system was evaluated solely for its ability to detect the fatigue indicator: Yawn. Each video was segmented into frames and processed, and the results of the evaluation are presented in Table 6.2.

Table 6.2: Performance Metrics Summary in YawDD

Tool	**Approach**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
MediaPipe Face Landmarker	Landmarks	92	99	85	92
MediaPipe Face Landmarker	Probabilities	95	97	93	95
ML Kit Face Mesh	Landmarks	93	99	87	93
ML Kit Face Detection (*)	Landmarks	94	99	90	94

(*) As before, yawn detection relied on MAR since no mouth openness probabilities were available.

Discussion

All models demonstrated strong performance on the YawDD dataset. The MediaPipe model with probabilities outputs once again provided the most robust results. However, even the ML Kit Face Detection model, despite being limited to MAR-based inference, achieved competitive performance, suggesting that mouth geometry alone is a reliable fatigue indicator under the controlled yawning conditions present in this dataset.

6.1.3 Results on NTHU-DDD Dataset

The NTHU-DDD dataset differs fundamentally from the others used in this research, as it provides binary fatigue labels for each frame. Rather than identifying individual fatigue indicators such

as eye blinks, yawns, or head tilts, it requires the system to interpret visual cues holistically and classify the overall state of the driver as either "fatigued" or "alert". This introduces a more realistic and challenging evaluation context, aligning more closely with real-world monitoring scenarios.

The evaluation was conducted using the system with the same four face detection model configurations previously mentioned: the MediaPipe Face Landmarker, tested both with and without probability outputs, the ML Kit Face Mesh, and the ML Kit Face Detection model. For each model, the full fatigue detection pipeline was executed frame by frame, and the predicted fatigue labels were compared directly against the ground truth annotations provided in the dataset.

In order to assess the system's performance under diverse visual conditions, the dataset was, as said in Chapter 5, evaluated only on specific subsets, based on the participants' eyewear: no glasses, transparent glasses, and sunglasses. Additionally, due to the nature of the dataset, five of the forty-two analyzed videos did not contain any instances of driver fatigue, resulting in no true positives (TP = 0). As a result, precision, recall, and F1-score were either zero or undefined for those cases; even so, accuracy remained relatively high, with an average of 96%, across all models. So, to avoid skewing the results, overall performance metrics were computed only on the other thirty-seven videos where fatigue events were present (TP > 0). Therefore, Table 6.3 presents the performance metrics for each model when evaluated across the selected dataset subsets, without separation by eyewear category.

Table 6.3: Performance Metrics on NTHU-DDD (All Categories Combined)

Tool	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
MediaPipe Face Landmarker (landmarks only)	73	84	62	71
MediaPipe Face Landmarker (with probabilities)	72	91	54	68
ML Kit Face Mesh	64	74	72	73
ML Kit Face Detection	63	62	83	71

In addition to the global performance overview, further analysis was performed by categorizing the data according to the presence and type of eyewear. In the first subset, drivers are not wearing any glasses, and facial features are fully visible under natural lighting. The second subset includes drivers wearing transparent glasses, which introduce minor eye obstructions and some reflection artefacts while still leaving the eye region mostly visible. In the third subset, drivers wear sunglasses, which completely obscure the eyes and significantly limit the visibility of one of the most important regions for visual fatigue estimation. Tables 6.4, 6.5, and 6.6 present the performance metrics, calculated independently for each category.

Table 6.4: Performance Metrics on NTHU-DDD: No Glasses

Tool	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
MediaPipe Face Landmarker (landmarks only)	76	89	61	72
MediaPipe Face Landmarker (with probabilities)	73	88	51	65
ML Kit Face Mesh	81	86	80	83
ML Kit Face Detection	51	53	94	68

Table 6.5: Performance Metrics on NTHU-DDD: Glasses

Tool	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
MediaPipe Face Landmarker (landmarks only)	70	82	55	66
MediaPipe Face Landmarker (with probabilities)	72	95	50	66
ML Kit Face Mesh	54	54	94	69
ML Kit Face Detection	75	71	79	75

Table 6.6: Performance Metrics on NTHU-DDD: Sunglasses

Tool	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
MediaPipe Face Landmarker (landmarks only)	74	81	67	73
MediaPipe Face Landmarker (with probabilities)	71	88	61	62
ML Kit Face Mesh	57	83	41	55
ML Kit Face Detection	60	61	78	69

Discussion

The results obtained from the NTHU-DDD dataset reveal clear trade-offs among the evaluated face detection models under varying eyewear categories. Although no single model outperformed across all metrics, there are patterns that help characterize the strengths and weaknesses of each configuration.

The MediaPipe Face Landmarker (landmarks only) configuration achieved a balanced performance. When configured with probability outputs, precision increased further, especially in scenarios involving eyewear accessories. However, this came at the cost of reduced recall, indicating a more conservative detection behavior that limits false positives but leads to missed detections in subtle or short-lived fatigue behaviors, such as a prolonged blink or brief yawn.

ML Kit Face Mesh achieved high recall in most categories, but its performance dropped significantly in the "Sunglasses" subset, where partial eye obstructions affected landmark reliability. In contrast, ML Kit Face Detection performed better in the "Sunglasses" category, likely due to its use of eye-state probabilities. However, its overall precision remained lower than the other models, indicating a tendency to over-detect and generate false positives.

An important factor influencing these results is the difference between the operational logic of the developed system and the dataset used for evaluation. In the developed application, once a fatigue alert is triggered and acknowledged by the user, the system assumes the driver is in the state "alert" again, preventing prolonged classification as fatigued. However, in the NTHU-DDD dataset, fatigue labels are applied continuously throughout prolonged behaviors, without any mechanism to reset the state. As a result, short detection gaps are penalized as false negatives, leading to lower recall scores. This mismatch also revealed unrealistic transitions within the videos of the dataset, where a single event, such as an isolated yawn, would immediately shift the label to fatigued, which does not align with a physiologically realistic fatigue progression.

Overall, these findings reinforce the benefit of hybrid models that integrate both geometric and probabilistic features, as they demonstrate greater resilience in ambiguous and visually challenging conditions.

6.2 Real-World Testing Results

This section presents the results from real-world testing of the mobile fatigue detection system implemented through an Android application. Unlike dataset-based evaluations that isolate detection performance, these sessions evaluated the full system pipeline, from live camera input to model inference and user interface updates, on physical devices and with real participants. The results help determine whether the system meets the functional and performance expectations defined earlier.

6.2.1 Detection Performance per Model and Participant

Detection performance was evaluated for each system configuration using a different face detection model across both test devices. P1 and P2 refer to Participant 1 and Participant 2, respectively. Each model-device-participant combination was tested twice in immediate succession using the predefined script present in Appendix C.2. As there was no negligible variation observed between repetitions, results are averaged across both trials.

The evaluation focused on fatigue alerts generated based on the six predefined behavioral rules, summarized in Table 4.2.4, present in Chapter 4. Although PERCLOS is included as one of these rules, it was not independently triggered during the tests. This occurred because the Eye Closure rule was always activated prior to the PERCLOS one. This functional redundancy was first identified during informal tests and later confirmed under the structured evaluation. As a result, while PERCLOS remains conceptually part of the rule-based logic, it had no operational impact on detection results in these real-world scenarios.

Now, regarding the performance metrics used for evaluation, accuracy was excluded as True Negatives could not be consistently defined within the scope of these targeted test scenarios. Precision, recall, and F1-score were selected as more representative indicators of detection reliability, with results presented in Tables 6.7 and 6.8.

Table 6.7: Detection Performance on Realme GT 6

Tool	Participant	Precision (%)	Recall (%)	F1-score (%)
MediaPipe Face Landmarker (landmarks only)	P1	85.7	85.7	85.7
	P2	85.7	75.0	80.0
MediaPipe Face Landmarker (with probabilities)	P1	100.0	100.0	100.0
	P2	100.0	85.7	92.3
ML Kit Face Mesh	P1	75.0	85.7	79.9
	P2	75.0	85.7	79.9
ML Kit Face Detection	P1	85.7	75.0	80.0
	P2	85.7	75.0	80.0

Table 6.8: Detection Performance on Google Pixel XL

Tool	**Participant**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
MediaPipe Face Landmarker (landmarks only)	P1	85.7	75.0	80.0
	P2	85.7	75.0	80.0
MediaPipe Face Landmarker (with probabilities)	P1	100.0	85.7	92.3
	P2	100.0	85.7	92.3
ML Kit Face Mesh	P1	66.7	85.7	75.0
	P2	75.0	85.7	79.9
ML Kit Face Detection	P1	75.0	85.7	79.9
	P2	85.7	75.0	80.0

Discussion

Overall, detection performance was generally consistent across participants, as expected given that the same script was followed, with only minor deviations observed. Participant 2 experienced occasional misclassifications due to strong sunlight interfering with eye detection. This resulted in some false positives, where the system mistakenly interpreted the participant's eyes as closed when they were, in fact, open. These situations were limited and clearly identifiable by reviewing the ground truth video. As these errors were isolated and the testing procedure ensured repeated attempts for each action, they did not result in any meaningful performance differences between participants.

Regarding the tested systems, the application configured with the MediaPipe Face Landmarker with probability outputs enabled showed the most reliable and stable performance. The availability of probabilities allowed for more robust thresholding in detecting blinks and yawns, maintaining both high precision and recall. When probability outputs were disabled, the system relied only on landmarks, which led to slightly lower consistency and occasional false positives, especially under varying lighting conditions. This occurred because the absence of probability scores made the system more susceptible to misinterpreting shadows or glare on the eyes as fatigue signs. While both configurations outperformed the others, the probabilistic version proved more effective in reducing detection noise and maintaining balanced results.

In contrast, the application configured with the ML Kit Face Mesh model displayed greater variability across sessions. While it often achieved high recall by detecting most of the expected fatigue behaviors, it did so at the cost of reduced precision. The lower precision was caused by instability in facial landmark detection and noise in the derived feature of head tilt angle, which led to frequent false positives. As a result, its F1-score suffered, making the model less suitable compared to other setups. Meanwhile, the application using the ML Kit Face Detection model produced balanced but moderate results. It performed reliably overall, although it did not reach the same level as either configuration of the MediaPipe model. Additionally, during testing on the Google Pixel XL, this model exhibited occasional false positives that were caused by drops in frame rate below 24 frames per second. These interruptions in frame continuity disrupted facial signal tracking, which in turn led to misclassification of behaviors such as prolonged eye closure.

In summary, the tested models revealed distinct trade-offs between sensitivity and precision. The ML Kit Face Mesh model triggered more detections due to unstable landmarks, not the threshold configuration, as it is the same for other models. In contrast, the MediaPipe setup with probability outputs delivered more accurate results, as probability scores helped suppress false activations in noisy conditions. Its consistent performance across sessions, even under varying lighting, makes it the most robust choice for driver safety applications, where avoiding false alerts is key to maintaining user trust and system reliability.

6.2.2 Model Inference Latency

The average inference latency for the system configured with each face detection model was measured on both test devices, based on timestamped log events. These values represent the mean time taken from image acquisition to fatigue detection output for a single frame. The results, in milliseconds, are summarized in Table 6.9.

Table 6.9: Average Inference Latency per Model and Device

Tool	**Realme GT 6 (ms)**	**Google Pixel XL (ms)**
MediaPipe Face Landmarker (landmarks only)	57	109
MediaPipe Face Landmarker (with probabilities)	97	143
ML Kit Face Mesh	43	71
ML Kit Face Detection	83	135

While all configurations maintained latency within acceptable limits for real-time processing, the Google Pixel XL consistently exhibited higher and more variable inference times compared to the Realme GT 6, highlighting the influence of hardware constraints. It is also important to note that running the application in release mode had a significant impact on performance: for example, on the Realme GT 6, the latency of the MediaPipe model dropped from over 400 ms in debug mode to under 100 ms in release mode. This underscores the importance of building a configuration when assessing system responsiveness.

6.3 Device Resource Usage Analysis

To complement the detection and latency metrics, system resource usage was monitored during the testing sessions. This includes CPU load, memory usage, and battery consumption, providing insight into the practical feasibility of deploying the application on mobile devices with varying capabilities.

6.3.1 CPU Usage

CPU consumption was tracked throughout each session to evaluate the computational impact of the different face detection models in the system. Figure 6.1 illustrates the CPU usage during a session using the MediaPipe (with probabilities) model on the Realme GT 6.

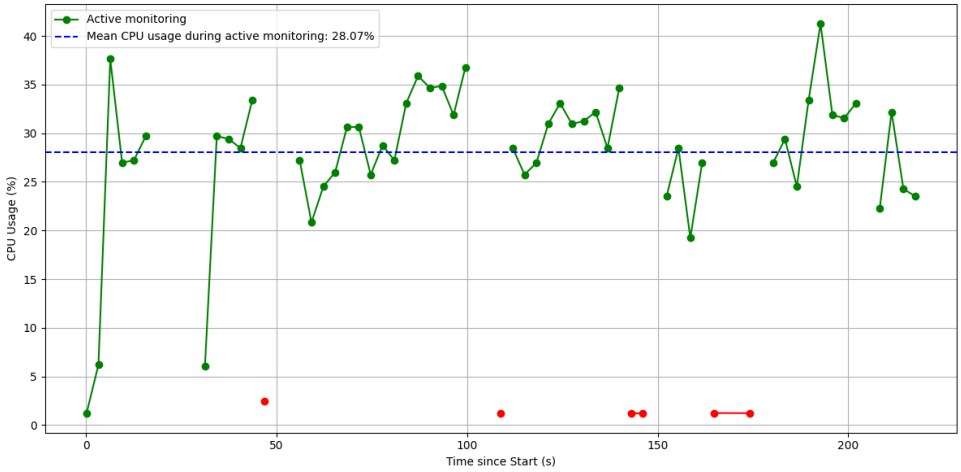


Figure 6.1: CPU usage during a testing session using the MediaPipe (with probabilities) model on Realme GT 6.

Each data point corresponds to a sample collected at regular intervals (5 seconds), with the green markers indicating active monitoring and red markers denoting temporary interruptions, when a fatigue alert is being shown. The dashed blue line represents the mean CPU usage during active monitoring, providing a reference for evaluating the system's computational demands under sustained operation.

To better understand the computational demands of each face detection model, average CPU usage was computed separately for active processing periods and idle periods, including when a fatigue alert was being shown or the session was paused, in both devices. Table 6.10 presents the results, highlighting clear differences in processing requirements depending on the model and hardware. Notably, the Realme GT 6 consistently required fewer CPU resources than the Google Pixel XL for all configurations, with active usage typically about half that observed on the older device.

Table 6.10: Average CPU Usage During Real-World Testing (Active vs. Inactive Periods)

Tool	**Device**	**Active CPU Usage (%)**	**Inactive CPU Usage (%)**
MediaPipe Face Landmarker (landmarks only)	Google Pixel XL	63.7	2.8
	Realme GT 6	31.4	0.9
MediaPipe Face Landmarker (with probabilities)	Google Pixel XL	64.0	2.4
	Realme GT 6	31.8	1.0
ML Kit Face Mesh	Google Pixel XL	30.6	2.2
	Realme GT 6	17.0	1.4
ML Kit Face Detection	Google Pixel XL	39.1	3.4
	Realme GT 6	18.7	0.8

When evaluating the computational efficiency of each setup, it is important to consider how the face detection AI models are integrated into the application. ML Kit is integrated as a native Android Software Development Kit (SDK), which makes it lightweight, straightforward to embed in mobile applications. MediaPipe, on the other hand, is a standalone library that, in this case,

manually loads and runs the model file. While this adds some processing overhead, it provides greater scalability as the same pipeline can be deployed across other platforms, including iOS and edge devices. This trade-off helps explain the observed CPU differences while highlighting the broader deployment potential of MediaPipe.

Consequently, despite using the same fatigue detection logic across all setups, ML Kit required about half the CPU resources compared to MediaPipe. This efficiency makes it more suitable for devices with limited processing power. Still, the system remained functional even on a 2016 smartphone, demonstrating that real-time fatigue detection is viable on older hardware, although with performance trade-offs.

6.3.2 RAM Usage & Battery Consumption

Across all models, RAM consumption remained stable throughout execution, with minor increases during initialization. On the Google Pixel XL, RAM usage reached approximately 240 MB, while on the Realme GT 6, it stayed around 100 MB. Overall usage remained low relative to total system capacity, with no significant variation between models, confirming that the application is viable even on older or memory-constrained devices. Battery consumption remained relatively low across all models, with only minor differences observed. Configurations using MediaPipe consistently consumed slightly more battery than those using ML Kit. On average, MediaPipe led to an additional 1% battery drain per session. This pattern was also confirmed during informal extended tests, where the battery dropped by roughly 1% every 10 minutes with ML Kit and around 2% with MediaPipe, reinforcing the impact of higher CPU usage in the latter configuration. Additionally, although not formally measured, a slight increase in device temperature was noticeable after approximately 30 minutes of continuous system use.

6.3.3 Summary

The system proved to be resource-efficient, with all models operating smoothly on both modern and older smartphones. While RAM usage remained consistently low across devices, differences emerged in CPU and battery performance. MediaPipe models consistently required more CPU resources and led to slightly higher battery consumption than ML Kit, despite relying on the same underlying fatigue detection logic. Because of that, a clear trade-off was observed between performance and integration approach: SDKs like ML Kit provide lightweight processing, making them ideal for maximizing battery life and minimizing CPU load, especially on low-end devices. In contrast, third-party libraries such as MediaPipe introduce higher computational overhead but offer greater flexibility and cross-platform deployment potential. Ultimately, the choice between SDK and library depends on the target use case and the range of supported hardware.

6.4 Functional Requirements and Quality Attributes Alignment

This section presents an overview of how the developed system aligns with the previously defined (Chapter 3) functional requirements (FRs) and non-functional requirements (NFRs), which served as guiding principles throughout the design and implementation phases. Tables 6.11 and 6.12 summarize the requirement coverage.

Table 6.11: Functional Requirements Compliance

ID	Title	Priority	Implemented
FR1	Splash screen	Must Have	✓
FR2	Home Screen	Must Have	✓
FR3	Display Live Front Camera Feed	Must Have	✓
FR4	Display Facial Landmarks Overlay on Live Camera Feed	Should Have	✓
FR5	Detect and Analyze Eye State	Must Have	✓
FR6	Detect and Analyze Mouth Movements for Yawning	Must Have	✓
FR7	Detect and Analyze Head Movements for Tilts	Must Have	✓
FR8	Collect Battery, CPU, GPU, and RAM Usage Data Every Minute	Should Have	✓
FR9	Allow User to Stop Monitoring	Must Have	✓
FR10	Provide an Alert When Critical Fatigue Signs Are Detected	Must Have	✓
FR11	Show a Summary of Detected Fatigue Events	Should Have	✓
FR12	Record Video During Monitoring Session	Could Have	✗
FR13	Export Monitoring Session Data	Should Have	✓
FR14	About Screen	Should Have	✓
FR15	Display Privacy Policy on the About Screen	Should Have	✓
FR16	Display Terms & Conditions on the About Screen	Should Have	✓
FR17	Enable Hands-Free Mode Using Voice Commands	Could Have	✗

Table 6.12: Non-Functional Requirements Fulfillment

ID	Title	Fulfilled
NFR1	Usability	✓
NFR2	Performance	✓
NFR3	Maintainability	✓
NFR4	Efficacy	Partially

All functional requirements categorized as "Must Have" and "Should Have" were successfully implemented, ensuring operational core features and a usable application. Two "Could Have" requirements were not implemented. FR12 (video recording) was excluded to avoid interfering with the acquisition of performance metrics, since recording while running the detection pipeline on the same device could distort the resource usage results. FR17 (voice interaction) was also left out to focus development on system stability and essential functionality. Consequently, these features can be included in future development with the other improvements mentioned in the "Limitations and Future Work" section of Chapter 7.

The non-functional requirements were also carefully considered. Usability (NFR1) was ensured through a streamlined and intuitive interface, which guides the user through the application with minimal effort, as seen in the screenshots present in Appendix B. Performance (NFR2) was successfully achieved, with latency remaining consistently below 150ms during real-time operation, thus complying with the predefined system requirement of responding to fatigue events within one second. Maintainability (NFR3) was achieved through the use of dependency injection (via Hilt), a layered architecture (with UI and data layers clearly separated), and a clean package structure. This design supports easier testing, refactoring, and long-term maintainability.

Efficacy (NFR4), related to the detection accuracy of fatigue states, was only partially fulfilled. While real-world tests and two out of three datasets showed strong performance, the system did not meet the expected 85% accuracy threshold on the NTHU-DDD dataset, where it achieved a mean of 65%. This result suggests that further model optimization and broader dataset validation are required to ensure consistently high detection reliability under diverse conditions.

In conclusion, the system fulfills all high-priority functional and non-functional requirements defined for this proof of concept. Its architecture and capabilities provide a solid foundation for ongoing development, testing, and eventual deployment through official mobile application stores.

6.5 Summary

This chapter presented an overview of the results of the evaluation of the developed fatigue detection system, combining both dataset-based and real-world testing. The analysis covered three public datasets, each contributing distinct fatigue indicators. Results showed that the MediaPipe model, with probability-based outputs, offered the most balanced performance, especially in blink and yawn detection, while also demonstrating robustness under varied visual conditions, including the presence of eyewear. In contrast, the ML Kit models, especially the Face Mesh, were more prone to false positives under less controlled scenarios, despite offering competitive detection coverage under ideal conditions. Real-world testing validated the system's ability to perform reliably in two different physical devices and participants, with consistent detection outputs and acceptable latency. In addition, the assessment of resource consumption measures validated the system's feasibility for mobile deployment, with all configurations maintaining low memory consumption and real-time responsiveness.

Overall, the findings support the use of hybrid geometric-probabilistic models for robust and efficient driver fatigue detection in realistic conditions. These insights lay the groundwork for the final reflections and future directions presented in Chapter 7.

Chapter 7

Conclusions

7.1 Key Findings

This dissertation set out to explore the feasibility of implementing a driver fatigue detection system directly on mobile devices using only built-in sensors and lightweight AI models. The explored solution combined on-device inference, facial landmark analysis, and rule-based logic to detect fatigue indicators in real-time, without relying on cloud infrastructure or external hardware.

A key outcome of this research was the development of a modular and efficient mobile application capable of detecting driver fatigue in real time. The system supports the integration and benchmarking of multiple AI frameworks, including MediaPipe and ML Kit, demonstrating that lightweight models can effectively extract facial features with sufficient speed and accuracy for real-time operation. Fatigue-related metrics such as PERCLOS, EAR, MAR, and head pose were computed directly on-device and were shown to correlate with fatigue levels, both in research datasets and real-world scenarios.

To evaluate the robustness and generalizability of the system, public research datasets such as DMD, YawDD, and NTHU-DDD were used. These tests confirmed that geometric indicators of fatigue can be reliably extracted under varying conditions, including different lighting environments, head poses, and driver accessories such as glasses. Subsequent real-world validation demonstrated consistent performance across different smartphones and participants. This reliable behavior, in diverse contexts, was made possible by a rule-based architecture, which integrated multiple visual cues, including geometric features and probabilistic outputs from MediaPipe and ML Kit models, into a coherent and interpretable pipeline. In addition, the system was capable of triggering alerts in less than one second of detecting critical events, fulfilling the responsiveness requirements for real-time feedback.

7.2 Practical and Scientific Implications

This work confirms that mobile devices can be effectively leveraged as standalone platforms for fatigue detection. To answer the first research question, this dissertation demonstrates that smart-

phone cameras, gyroscopes, and accelerometers, when combined with lightweight AI models, can form an effective and non-intrusive driver monitoring system. The modular architecture and on-device inference approach ensure privacy and reduce latency, contributing to safer and more accessible driver-assistance solutions, particularly in low- and middle-income settings.

Regarding the second research question, the research identified and measured key trade-offs between latency, energy consumption, and detection accuracy. While MediaPipe provided better accuracy due to its better probabilistic outputs, it incurred slightly higher CPU usage. In comparison, ML Kit variants were lighter on resources but less robust under challenging conditions, such as sunglasses and high-intensity lighting. Nonetheless, both frameworks operated within acceptable latency limits, confirming the feasibility of real-time inference under constrained conditions.

The outcomes of this dissertation have also led to peer-reviewed dissemination. A paper was accepted for publication in the proceedings of the 19th International Conference on Computer Graphics, Visualization, Computer Vision and Image Processing, where it will be presented between 23 and 25 July 2025. Furthermore, the findings of the research were also presented at the 6th Doctoral Congress in Engineering, held on 30 June and 1 July 2025.

7.3 Limitations and Future Work

Despite its promising results, the system presented in this dissertation has some limitations. The most relevant constraint was the lack of timely access to datasets annotated with fatigue states, which prevented the development of a custom AI model for fatigue classification. Although datasets were available to detect specific indicators such as eye closure or yawning, these were already covered by MediaPipe and ML Kit. As a result, the system used a rule-based decision layer that, while effective and explainable, limits adaptability and personalization.

Future work should focus on making use of the NTHU-DDD dataset to train dedicated deep learning models for fatigue classification. This would allow for comparison with the developed system and possibly improve its ability to generalize across different users and conditions by evolving it beyond rule-based logic.

It is also important to explore how motion data from the smartphone's inertial sensors can be integrated into the system. Previous research has shown that the accelerometer can detect sudden braking or loss of speed, both of which are potential signs of fatigue. This type of data was not explored in this dissertation, as the focus of the work remained on visual indicators, and no suitable datasets were timely found to support testing. However, integrating these behavioral signals with the visual metrics already implemented would be a valuable direction for future work, potentially leading to a more complete and reliable fatigue detection system.

These enhancements improve the system's ability to detect fatigue reliably, its flexibility, and real-world usability, reinforcing its promise as a scalable, on-device solution to mitigate one of the most critical and underestimated risks on today's roads.

References

[1] Gulbadan Sikander and Shahzad Anwar. “Driver fatigue detection systems: A review”. In: *IEEE Transactions on Intelligent Transportation Systems* 20.6 (2018), pp. 2339–2352.

[2] Ashiqur Rahman, Mamun Bin Harun Hriday, and Riasat Khan. “Computer vision-based approach to detect fatigue driving and face mask for edge computing device”. In: *Heliyon* 8.10 (2022).

[3] Jonathan J. Rolison. “Identifying the causes of road traffic collisions: Using police officers’ expertise to improve the reporting of contributory factors data”. In: *Accident Analysis & Prevention* 135 (2020), p. 105390.

[4] Shichen Fu et al. “Advancements in the intelligent detection of driver fatigue and distraction: a comprehensive review”. In: *Applied Sciences* 14.7 (2024), p. 3016.

[5] Pierre Thiffault and Jacques Bergeron. “Monotony of road environment and driver fatigue: a simulator study”. In: *Accident Analysis & Prevention* 35.3 (2003), pp. 381–391.

[6] Fangming Qu et al. “Comprehensive study of driver behavior monitoring systems using computer vision and machine learning techniques”. In: *Journal of Big Data* 11.1 (2024), p. 32.

[7] Bing-Ting Dong and Huei-Yung Lin. “An on-board monitoring system for driving fatigue and distraction detection”. In: *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*. Vol. 1. IEEE. 2021, pp. 850–855.

[8] Jacob Poushter, Sneha Gubbala, and Sarah Austin. *8 charts on technology use around the world*. Pew Research Center. 2024.

[9] Alexey Kashevnik et al. “Cloud-based driver monitoring system using a smartphone”. In: *IEEE Sensors Journal* 20.12 (2020), pp. 6701–6715.

[10] J. He et al. “Fatigue detection using smartphones”. In: *Journal of Ergonomics* 3.03 (2013), pp. 1–7.

[11] R. Manoharan and S. Chandrakala. “Android OpenCV based effective driver fatigue and distraction monitoring system”. In: *2015 International Conference on Computing and Communications Technologies (ICCCT)*. IEEE. 2015, pp. 262–266.

[12] Chaohui Yu et al. “Drowsydet: a mobile application for real-time driver drowsiness detection”. In: *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE. 2019, pp. 425–432.

[13] Iman Chatterjee and Sarbani Roy. “Smartphone-based drowsiness detection system for drivers in real-time”. In: *2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE. 2019, pp. 1–6.

[14] Jelica Davidović, Dalibor Pešić, and Boris Antić. "Professional drivers' fatigue as a problem of the modern era". In: *Transportation Research Part F: Traffic Psychology and Behaviour* 55 (2018), pp. 199–209.

[15] Clare Anderson et al. "Assessment of drowsiness based on ocular parameters detected by infrared reflectance oculography". In: *Journal of Clinical Sleep Medicine* 9.9 (2013), pp. 907–920.

[16] Bappaditya Mandal et al. "Towards detection of bus driver fatigue based on robust visual analysis of eye state". In: *IEEE Transactions on Intelligent Transportation Systems* 18.3 (2016), pp. 545–557.

[17] Mohamed Hedi Baccour et al. "Comparative analysis of vehicle-based and driver-based features for driver drowsiness monitoring by support vector machines". In: *IEEE Transactions on Intelligent Transportation Systems* 23.12 (2022), pp. 23164–23178.

[18] M. Ahmad Kamran, Malik M. Naeem Mannan, and Myung Yung Jeong. "Drowsiness, fatigue and poor sleep's causes and detection: a comprehensive study". In: *IEEE Access* 7 (2019), pp. 167172–167186.

[19] Azmeh Shahid et al. "Stanford Sleepiness Scale (SSS)". In: *STOP, THAT and One Hundred Other Sleep Scales*. Ed. by Azmeh Shahid et al. New York, NY: Springer New York, 2012, pp. 369–370. ISBN: 978-1-4419-9893-4.

[20] Arun Sahayadhas, Kenneth Sundaraj, and Murugappan Murugappan. "Drowsiness detection during different times of day using multiple features". In: *Australasian Physical & Engineering Sciences in Medicine* 36 (2013), pp. 243–250.

[21] Zuojin Li et al. "A fuzzy recurrent neural network for driver fatigue detection based on steering-wheel angle sensor data". In: *International Journal of Distributed Sensor Networks* 15.9 (2019), p. 1550147719872452.

[22] Volkswagen. *Driver Assistance Technologies*. Available: https://www.volkswagen.com.my/brand-experience/technology/driver-assistance.

[23] Rishu Chhabra, Seema Verma, and C. Rama Krishna. "Detecting aggressive driving behavior using mobile smartphone". In: *Proceedings of 2nd International Conference on Communication, Computing and Networking: ICCCN 2018, NITTTR Chandigarh, India*. Springer. 2019, pp. 513–521.

[24] Arief Koesdwiady et al. "Recent trends in driver safety monitoring systems: State of the art and challenges". In: *IEEE Transactions on Vehicular Technology* 66.6 (2016), pp. 4550–4563.

[25] Mark R. Rosekind et al. "Awake at the wheel: how auto technology innovations present ongoing sleep challenges and new safety opportunities". In: *Sleep* 47.2 (2024), zsad316.

[26] Hongtao Wang et al. "A novel real-time driving fatigue detection system based on wireless dry EEG". In: *Cognitive Neurodynamics* 12.4 (2018), pp. 365–376.

[27] Suganiya Murugan, Jerritta Selvaraj, and Arun Sahayadhas. "Detection and analysis: driver state with electrocardiogram (ECG)". In: *Physical and Engineering Sciences in Medicine* 43.2 (2020), pp. 525–537.

[28] Muhammad Atif Munir et al. "IoT based automotive driver drowsiness prediction system using heart rate variability and galvanic skin response". In: *2020 International Conference on Engineering and Emerging Technologies (ICEET)*. IEEE. 2020, pp. 1–6.

[29] Andrea Amidei et al. “Validating Photoplethysmography (PPG) data for driver drowsiness detection”. In: *2021 IEEE International Workshop on Metrology for Automotive (MetroAutomotive)*. IEEE. 2021, pp. 147–151.

[30] Eddie E. Galarza et al. “Real time driver drowsiness detection based on driver’s face image behavior using a system of human computer interaction implemented in a smartphone”. In: *Proceedings of the International Conference on Information Technology & Systems (ICITS 2018)*. Springer. 2018, pp. 563–572.

[31] Teik Jin Lim et al. “Eye fatigue algorithm for driver drowsiness detection system”. In: *Image Processing and Capsule Networks: ICIPCN 2020*. Springer. 2021, pp. 638–652.

[32] Juan Huang and Zihui Lin. “Multi-feature fatigue driving detection based on computer vision”. In: *Journal of Physics: Conference Series*. Vol. 1651. 1. IOP Publishing. 2020, p. 012188.

[33] Dohun Kim et al. “Real-time driver monitoring system with facial landmark-based eye closure detection and head pose recognition”. In: *Scientific Reports* 13.1 (2023), p. 18264.

[34] Ashish Kumar and Rusha Patra. “Driver drowsiness monitoring system using visual behaviour and machine learning”. In: *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*. IEEE. 2018, pp. 339–344.

[35] Naveen Senniappan Karuppusamy and Bo-Yeong Kang. “Multimodal system to detect driver fatigue using EEG, gyroscope, and image processing”. In: *IEEE Access* 8 (2020), pp. 129645–129667.

[36] Yiting Wang et al. “Intelligent Fatigue Driving Detection Method Based on Fusion of Smartphone and Smartwatch Data”. In: *China National Conference on Big Data and Social Computing*. Springer. 2024, pp. 186–198.

[37] Tahesin Samira Delwar et al. “AI-and Deep Learning-Powered Driver Drowsiness Detection Method Using Facial Analysis”. In: *Applied Sciences* 15.3 (2025), p. 1102.

[38] Alexey Kashevnik, Igor Lashkov, and Andrei Gurtov. “Methodology and mobile application for driver behavior analysis and accident prevention”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.6 (2019), pp. 2427–2436.

[39] B. Rajkumarsingh and D. Totah. “Drowsiness detection using android application and mobile vision face API”. In: *R&D Journal* 37 (2021), pp. 26–34.

[40] Yantao Qiao et al. “A smartphone-based driver fatigue detection using fusion of multiple real-time facial features”. In: *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2016, pp. 230–235.

[41] Jomin Jose, Kumudha Raimond, Shweta Vincent, et al. “SleepyWheels: An Ensemble Model for Drowsiness Detection leading to Accident Prevention”. In: *arXiv preprint arXiv:2211.00718* (2022).

[42] Oraan Khunpisuth et al. “Driver drowsiness detection using eye-closeness detection”. In: *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*. IEEE. 2016, pp. 661–668.

[43] Yu Cheng et al. “A survey of model compression and acceleration for deep neural networks”. In: *arXiv preprint arXiv:1710.09282* (2017).

[44] Chengyou Lin et al. “Early Driver Fatigue Detection System: A Cost-Effective and Wearable Approach Utilizing Embedded Machine Learning”. In: *Vehicles* 7.1 (2025), p. 3.

[45] Qaisar Abbas and Abdullah Alsheddy. “A methodological review on prediction of multi-stage hypovigilance detection systems using multimodal features”. In: *IEEE Access* 9 (2021), pp. 47530–47564.

[46] Weisong Shi et al. “Edge computing: Vision and challenges”. In: *IEEE Internet of Things Journal* 3.5 (2016), pp. 637–646.

[47] Ziheng Jiang, Tianqi Chen, and Mu Li. *Efficient deep learning inference on edge devices*. Available: https://www.amazon.science/publications/efficient-deep-learning-inference-on-edge-devices. 2018.

[48] Nasir Abbas et al. “Mobile edge computing: A survey”. In: *IEEE Internet of Things Journal* 5.1 (2017), pp. 450–465.

[49] Ying Chen et al. “Energy efficient dynamic offloading in mobile edge computing for internet of things”. In: *IEEE Transactions on Cloud Computing* 9.3 (2019), pp. 1050–1060.

[50] Yulun Du et al. “Multimodal polynomial fusion for detecting driver distraction”. In: *arXiv preprint arXiv:1810.10565* (2018).

[51] Cheng Xu et al. “Mmbench: Benchmarking end-to-end multi-modal DNNs and understanding their hardware-software implications”. In: *2023 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE. 2023, pp. 154–166.

[52] Cong Hao and Deming Chen. “Software/hardware co-design for multi-modal multi-task learning in autonomous systems”. In: *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE. 2021, pp. 1–5.

[53] Google. *App Architecture*. 2025. Available: https://developer.android.com/topic/architecture.

[54] Robert C. Martin. *Clean Architecture*. Prentice Hall. 2017.

[55] Camillo Lugaresi et al. “MediaPipe: A Framework for Perceiving and Processing Reality”. In: *Proceedings of the Third Workshop on Computer Vision for AR/VR at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2019.

[56] Google. *ML Kit Face Detection*. 2025. Available: https://developers.google.com/ml-kit/vision/face-detection.

[57] Google. *ML Kit Face Mesh Detection*. 2025. Available: https://developers.google.com/ml-kit/vision/face-mesh-detection.

[58] Amonrat Prasitsupparote, Pakorn Pasitsuparoad, and Sirathee Itsarapongpukdee. “Real-Time Driver Drowsiness Alert System for Product Distribution Businesses in Phuket Using Android Devices”. In: *2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE. 2023, pp. 351–355.

[59] Muhammad Faique Shakeel et al. “Detecting driver drowsiness in real time through deep learning based object detection”. In: *International Work-Conference on Artificial Neural Networks*. Springer. 2019, pp. 283–296.

[60] Sadegh Arefnezhad et al. “Driver drowsiness estimation using EEG signals with a dynamical encoder–decoder modeling framework”. In: *Scientific Reports* 12.1 (2022), p. 2650.

[61] Samarpit Karar and Tirupathiraju Kanumuri. “Assessment of Driver Fatigue and Drowsiness Based on Eye Blink Rate”. In: *International Conference on Data Analytics & Management*. Springer. 2023, pp. 311–324.

[62] Mohammad Elham Walizad, Mehreen Hurroo, and Divyashikha Sethia. “Driver drowsiness detection system using convolutional neural network”. In: *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE. 2022, pp. 1073–1080.

[63] Juan Diego Ortega et al. “DMD: A large-scale multi-modal driver monitoring dataset for attention and alertness analysis”. In: *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part IV*. Springer. 2020, pp. 387–405.

[64] Shabnam Abtahi et al. *YawDD: Yawning Detection Dataset*. 2020.

[65] Ching-Hua Weng, Ying-Hsiu Lai, and Shang-Hong Lai. “Driver drowsiness detection via a hierarchical temporal deep belief network”. In: *Computer Vision–ACCV 2016 Workshops, Taipei, Taiwan, November 20–24, 2016, Revised Selected Papers, Part III*. Springer. 2017, pp. 117–133.

[66] Olivier Friard and Marco Gamba. “BORIS: a free, versatile open-source event-logging software for video/audio coding and live observations”. In: *Methods in Ecology and Evolution* (2016), pp. 1324–1330. ISSN: 2041-210X.

All websites were consulted and available at the delivery date for this dissertation (30/06/2025)

Appendix A

Functional Requirements

F1 Splash Screen

Description: When the user opens the app, a splash screen with the logo is displayed until the app finishes loading.
Priority: Must Have
Estimation: 2
Dependency: Not Applicable
Wireframe Reference: Splash Screen (Figure B.1a)

F2 Home Screen

Description: The Home screen is shown after the app loads and must have two buttons:

- Start Monitoring, which redirects to the Monitoring Screen starting a session;
  - The session ID will be formatted as 'DeviceName_HH:MM:SS' and will be used to record session logs, which can be exported upon completion.
- About, which redirects to the About Screen.

Priority: Must Have
Estimation: 3
Dependency: Not Applicable
Wireframe Reference: Home Screen (Figure B.1b)

F3 Start Monitoring

Description: The system must provide a real-time video feed of the driver's face while monitoring facial features. The video feed should start after the user grants permission to access the camera upon opening the Monitoring Screen.

Failure Cases:

- If the user denies camera permission, the system must display a message informing that camera access is required for monitoring;
- If the camera fails to initialize, the system must display a message informing the user to check their device settings or restart the app;

Priority: Must Have
Estimation: 3
Dependency: Not Applicable
Wireframe Reference: Monitoring Screen (Figure B.1a)

F4 Display Facial Landmarks Overlay on Live Camera Feed

Description: The system should provide an option to display a mesh overlaid with real-time facial landmarks on the live camera feed while monitoring facial features. The overlay should be optionally activated/deactivated by the user through a button, present in the screen.
Priority: Should Have
Estimation: 3
Dependency: F3
Wireframe Reference: Monitoring Screen (Figure B.2c)

F5 Detect and Analyze Eye State

Description: The system must detect whether the driver's eyes are open or closed in real time and provide visual feedback by displaying the corresponding label:

- "Open", when eyes are detected as open;
- "Closed", when eyes are detected as closed;
- "Not detected", when eye(s) cannot be detected.

Notes:

- The system should log eye state changes with timestamps for performance analysis;
- The collected data should be stored temporarily and deleted if not exported on the Trip Summary Screen.

Priority: Must Have
Estimation: 8
Dependency: F3
Wireframe Reference: Monitoring Screen (Figure B.2c)

F6 Detect and Analyze Mouth Movements for Yawning

Description: The system must detect yawning in real time and provide visual feedback by displaying the corresponding label:

- "Yawn", when yawning is detected;

- "Not Yawn", when no yawning is detected;
- "Not detected", when mouth cannot be detected.

Notes:

- The system should log mouth state changes with timestamps for performance analysis;
- The collected data should be stored temporarily and deleted if not exported on the Trip Summary Screen.

Priority: Must Have
Estimation: 8
Dependency: F3
Wireframe Reference: Monitoring Screen (Figure B.2c)

F7 Detect and Analyze Head Movements for Tilts

Description: The system must detect head tilts in real time and provide visual feedback by displaying the corresponding label:

- "Tilted", when a significant head tilt is detected;
- "Normal", when the head is in a regular position;
- "Not detected", when the face cannot be detected.

Notes:

- The system should log head state changes with timestamps for performance analysis;
- The collected data should be stored temporarily and deleted if not exported on the Trip Summary Screen.

Priority: Must Have
Estimation: 8
Dependency: F3
Wireframe Reference: Monitoring Screen (Figure B.2c)

F8 Collect Battery, CPU, and RAM Usage Data Every Minute

Description: The system should collect and log hardware performance metrics every minute, including:

- Battery Status: Current battery percentage and charging state;
- CPU Usage: Percentage of CPU utilization, by the application;
- RAM Usage: Amount of RAM currently being used, by the application;
- GPU Usage: Percentage of GPU utilization, by the application, if available.

Notes:

- The collected data should be stored temporarily and optionally exported for performance analysis;
- The system should ensure minimal performance overhead while gathering these metrics;
- If data collection fails, the system should retry and log the failure;

- Data should be associated with a session (via ID) and deleted if not exported on the Trip Summary Screen.

Priority: Should Have
Estimation: 8
Dependency: F3
Wireframe Reference: Not Applicable

F9 Stop Monitoring

Description: The system must provide the user with an option to manually stop the monitoring session by clicking the "Stop Monitoring" button. Upon stopping, the user must be redirected to the "Trip Summary" screen.
Priority: Must Have
Estimation: 2
Dependency: Not Applicable
Wireframe Reference: Monitoring Screen (Figure B.2c)

F10 Provide an Alert When Critical Fatigue Signs Are Detected

Description: The system must generate an alert message in a modal window whenever the driver exhibits critical signs of fatigue.

Notes:

- The alert should be triggered when the AI model detects fatigue in the user;
- The modal should display a warning message;
- The alert should pause monitoring temporarily and require user acknowledgment before resuming, by pressing any place on the screen to resume the monitoring;
- The system should emit an alarm sound to ensure driver awareness;
- The system should log when the Alert UI change happens with timestamps for performance analysis;
- The collected data should be stored temporarily and deleted if not exported on the Trip Summary Screen.

Priority: Must Have
Estimation: 8
Dependency: F8
Wireframe Reference: Fatigue Detected Alert Screen (Figure B.3c)

F11 Show a Summary of Detected Fatigue Events

Description: After the user stops monitoring, the system should present a summary of the monitoring session, displaying:

- Total number of eye blinks;
- Total number of yawns;
- Total number of head tilts;
- Total number of fatigue alerts triggered;
- % of the time that the face wasn't detected;
- Total duration of the trip;
- A "Restart" button, to start a new monitoring session;
- A "Go to Home Screen" button, to return to the main menu.

Priority: Should Have
Estimation: 5
Dependency: F5, F6, F7, F8 & F11
Wireframe Reference: Session Summary Screen (Figure B.3d)

F12 Record Video During Monitoring Session

Description: The system could provide an option to record a video of the driver's phone front camera, the one used for monitoring, while monitoring is active.

The recorded video should:

- Start automatically when monitoring begins and stop when monitoring ends;
- Be linked to the session ID for easy reference;
- Be stored temporarily on the device;
- If not exported on the Trip Summary Screen, the recorded video should be deleted automatically to save storage space.

Notes:

- The feature should be optional and can be enabled/disabled in the Home Screen.

Priority: Could Have
Estimation: 5
Dependency: F3
Wireframe Reference: Not Applicable

F13 Export Monitoring Session Data

Description: The system must allow the user to export a detailed timeline report of the monitoring session, which includes event logs, device performance data and an optional recorded video of the session. The report should be available in CSV or JSON format, with the name the same as the session ID and with the following export options:

- If there is no connection to the internet: The file should be downloaded locally to the device;
- If there is a connection to the internet: The file should be uploaded to Pink Room's Google Drive for cloud storage.

The exported files must contain:

- Timeline of the Monitoring Session (with timestamps):
  - Eye Blinks - Each detected blink event with a timestamp;
  - Yawns - Each detected yawning event with a timestamp;
  - Head Tilts - Each detected head tilt event with a timestamp;
  - Fatigue Alerts - Each triggered fatigue alert with a timestamp.
- Device Performance Logs (collected throughout the session):
  - Battery Usage Timeline - Battery level percentage over time;
  - CPU Usage Timeline - CPU utilization during the session;
  - RAM Usage Timeline - Memory (RAM) consumption over time;
  - GPU Usage Timeline - GPU utilization during the session, if available.
- Recorded Video of the Session (Optional).

Priority: Should Have
Estimation: 5
Dependency: F5, F6, F7, F8, F11 & F12
Wireframe Reference: Session Summary Screen (Figure B.3d)

F14 About Screen

Description: The system should include an "About" screen containing:

- A brief description of the app and its purpose;
- Pink Room and Faculty of Engineering of the University of Porto logos, each hyperlinked to their respective websites, so when clicked, they redirect the user to the websites;
- A "Back" button to return to the Home Screen;
- Privacy Policy and Terms & Conditions sections (F13 and F14).

Priority: Should Have
Estimation: 3
Dependency: Not Applicable
Wireframe Reference: About Screen (Figure B.1c)

F15 Display Privacy Policy on the About Screen

Description: The system should include a Privacy Policy section within the About Screen, summarizing key privacy policies.

The Privacy Policy section should provide a short summary of:

- How user data is handled;
- Details about data collection, processing and user rights.

Priority: Should Have
Estimation: 3
Dependency: F15

Wireframe Reference: About Screen (Figure B.1c)

F16 Display Terms & Conditions on the About Screen

Description: The system should include a Terms & Conditions section within the About Screen, summarizing key terms of use.

The Terms & Conditions section should provide a short summary of:

- The limitations of liability in case of incorrect fatigue detection;
- The permitted and prohibited uses of the app.

Priority: Should Have
Estimation: 3
Dependency: F15
Wireframe Reference: About Screen (Figure B.1c)

F17 Enable Hands-Free Mode Using Voice Commands

Description: The system could allow users to interact with monitoring alerts hands-free using voice commands, ensuring a safer operation while driving. Users should be able to:

- Stop an alert using a voice command;
- Acknowledge and dismiss an alert using a voice command.

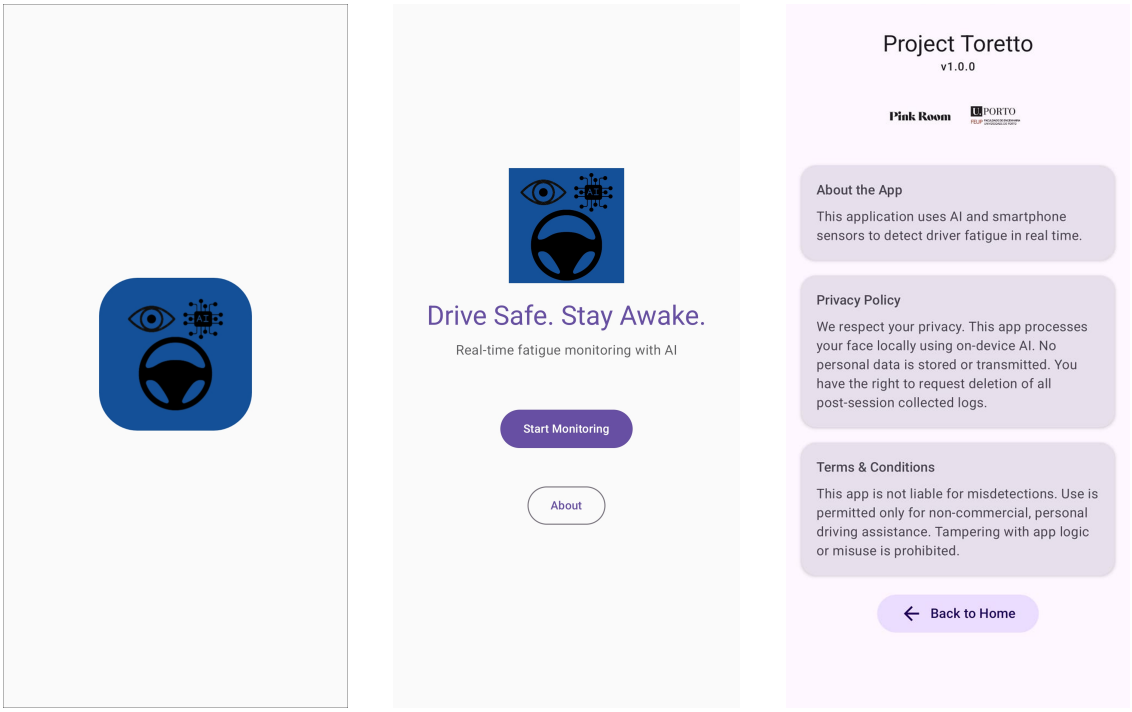
Notes:

- The feature should be optional and it's operational management should be in the Home Screen.

Priority: Could Have
Estimation: 5
Dependency: F3 & F11
Wireframe Reference: Not Applicable

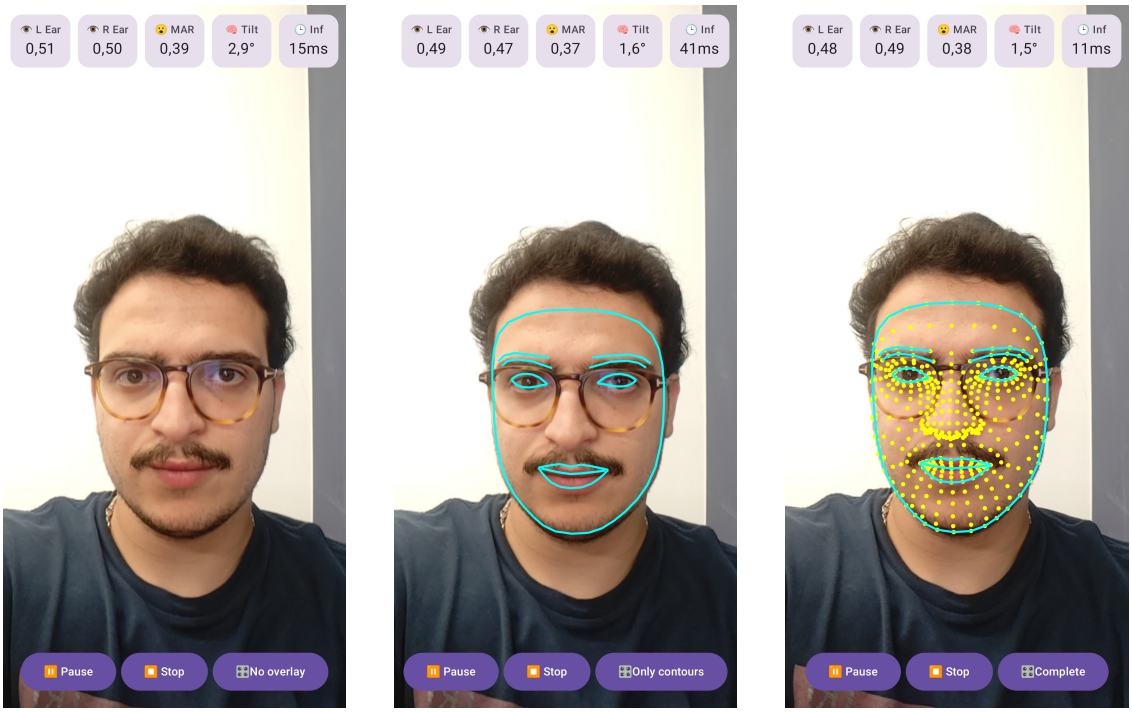
Appendix B

Screenshots of the Proof of Concept Application



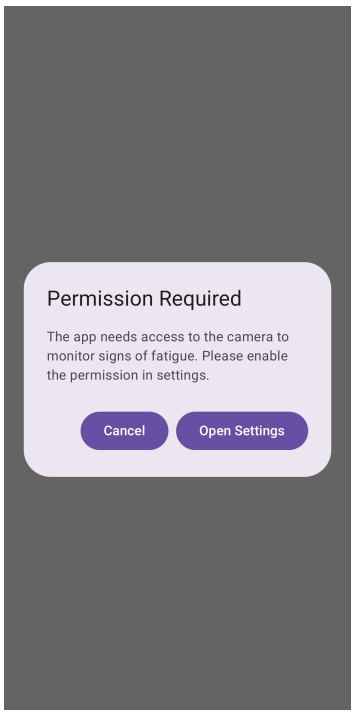
(a) Splash screen (b) Home screen (c) About section

Figure B.1: Splash, Home and About screens of the Driver Fatigue Detection Proof of Concept Application.

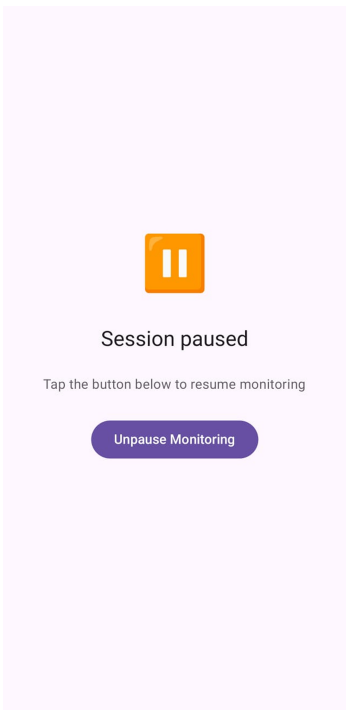


(a) Monitoring screen (Normal) (b) Monitoring screen (Contours) (c) Monitoring screen (All)

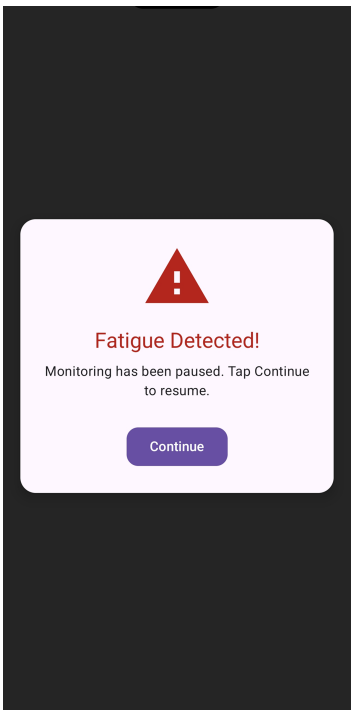
Figure B.2: Monitoring screen modes of the Driver Fatigue Detection Proof of Concept Application: normal mode, facial contours, and full landmark overlay.



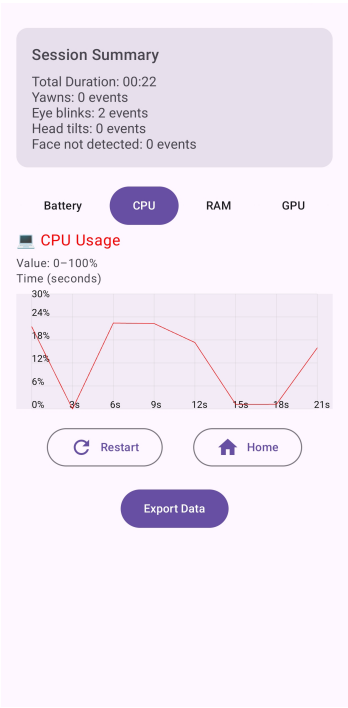
(a) Permission required screen



(b) Monitoring session paused



(c) Fatigue detected alert



(d) Session summary screen

Figure B.3: System responses before and during monitoring: permission request, session pause, and fatigue detection alert; and session summary screen.

Appendix C

Script for Real-World Testing Sessions of the PoC

C.1 Participant Briefing

Hello! Before we begin, I will explain what will happen during this test to make sure everything runs smoothly.

This test aims to validate a mobile application that detects signs of driver fatigue using the front-facing camera of the smartphone. For approximately five to six minutes, I will be supervising the process and giving you verbal instructions.

You will be seated comfortably in the car, hands on the steering wheel, simulating a driving posture. Three smartphones will be used: two are mounted in front of you, and a third one will be recording you for later comparison with the application's results.

During the test, I will ask you to:

- Close your eyes for a few seconds;
- Blink several times;
- Try not to blink for one minute;
- Simulate yawns, i.e., open your mouth wide;
- And nod your head forward as if you are dozing off.

You will follow my instructions naturally. During the session, you should try to only perform the actions indicated above. During the test, alerts may appear on the screen as the application detects and records any fatigue-detected events automatically. Please ignore them unless I explicitly ask you to press the on-screen button.

At the end, the data collected will be analyzed to determine if the application correctly detected the simulated signs of fatigue.

Do you have any questions before we begin?

C.2 Timed Action Script

Time (min:sec)	Observer Command
–	Press "Monitoring" on both phones at the same time.
0:00-0:10	Sit comfortably and look naturally at the road in front of you.
0:10	We will now start the test for Rule 1: Eye Closure.
0:15	Close your eyes and keep them closed.
0:18	Open your eyes.
0:18-0:28	[10-second buffer: Accept any alert on devices]
0:28	We will now test Rules 2 and 3: Blink Count. When I say “blink”, blink.
0:35-1:13	*(Say "Blink" repeatedly, every 2 seconds, so 20 times in the interval)*
1:13-1:23	[10-second buffer: Accept any alert on devices]
1:28-2:28	Try not to blink for 1 minute. You may blink involuntarily, but focus not to.
2:28-2:38	[10-second buffer: Accept any alert on devices]
2:38	We will now test Rules 4 and 5: Frequent Yawning. When I say “Open” or "Close", do the movement with your mouth, starting closed.
2:45	Open.
2:47	Close.
2:49	Open.
2:51	Close and let’s wait 5 seconds.
2:56	Open.
2:58	Close.
3:00	Open.
3:02	Close.
3:02-3:12	[10-second buffer: Accept any alert on devices]
3:12-3:17	Let’s continue. Open and close your mouth as instructed, starting closed.
3:17	Open.
3:19-3:21	(Say "Close and Open" twice, 2 seconds apart)
3:23	Close.
3:23-3:33	[10-second buffer: Accept any alert on devices]
3:33	We will now test Rule 6: Head Nods.
3:35	Tilt your head forward.
3:39	Return your head to normal.
3:41	Tilt forward.
3:45	Return to normal.
3:47	Tilt forward.
3:51	Return to normal.
3:53	Tilt forward.
3:57	Return to normal.
3:57-4:07	[10-second buffer: Accept any alert on devices]
4:07-4:30	Behave naturally. Look forward and relax. No further instructions.