Driver Drowsiness Detection System (DDDS)

Acceptance Test Report

Submission Date: April 14, 2025

**Submitted to:  
Prof. Emel Koç**

Prepared by:  
Khaled Alrefai  
Student ID: 220212334  
Department of Computer Engineering

Contents

[1. Introduction 2](#_Toc195564890)

[1.1 Purpose of this document 2](#_Toc195564891)

[1.2 Scope of the development project 3](#_Toc195564892)

[1.3 Summary of results 3](#_Toc195564893)

[1.4 Definitions, Acronyms and Abbreviations 3](#_Toc195564894)

[2. Performed Tests and Results 3](#_Toc195564895)

[2.1 Capture Stage 3](#_Toc195564896)

[2.2 Processing Stage 4](#_Toc195564897)

[2.3 Alarming Stage 4](#_Toc195564898)

[2.4 System-Level Tests 4](#_Toc195564899)

[3. Conclusion 6](#_Toc195564900)

[3.1 Acceptance Criteria 6](#_Toc195564901)

[3.2 Acceptance Results 6](#_Toc195564902)

[3.3 Success Performance 6](#_Toc195564903)

[4. References 7](#_Toc195564904)

# 1. Introduction

## 1.1 Purpose of this document

The purpose of this Acceptance Test Report is to document the outcomes of test cases executed on the Driver Drowsiness Detection System (DDDS). These tests verify the implementation’s alignment with the functional and non-functional requirements specified in the Software Requirements Specification (SRS) and Software Design Document (SDD). It further serves to demonstrate the progress made through simulation and partial hardware integration.

## 1.2 Scope of the development project

The Driver Drowsiness Detection System (DDDS) is a real-time intelligent monitoring system that aims to detect signs of driver fatigue through video input and machine learning inference. The project is implemented with a focus on embedded systems and TinyML, capable of operating under low power with efficient image processing. The system utilizes a camera to capture eye-region images, processes them using a lightweight trained CNN model, and triggers an alert mechanism such as a buzzer or LED when signs of drowsiness are detected. All tests were conducted in a simulated environment using OpenCV and TensorFlow, with deployment options on ESP32-S3, ESP32-CAM, Maixduino k210, or Raspberry pi planned for integration.

## 1.3 Summary of results

Tests across capture, processing, and alerting stages confirm that most core functionalities work as intended. Several tests—particularly related to power usage and long-duration runtime—remain pending final hardware integration. All components tested in simulation matched system expectations. This document captures the current verified status and test outcomes. Additional tests such as power usage and continuous uptime have been scheduled for the upcoming graduation phase.

## 1.4 Definitions, Acronyms and Abbreviations

SRS: Software Requirements Specification  
SDD: Software Design Document  
DDDS: Driver Drowsiness Detection System  
TinyML: Tiny Machine Learning  
CNN: Convolutional Neural Network  
TFLite: TensorFlow Lite  
FPS: Frames Per Second  
GPIO: General Purpose Input/Output  
UART: Universal Asynchronous Receiver/Transmitter

# 2. Performed Tests and Results

## 2.1 Capture Stage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Result | Comments | Reference |
| CAP01 | Camera initializes and streams frames | ✅ Pass | Camera feed stable and smooth using OpenCV | SRS 6.1.1 |
| CAP02 | Frame rate ≥15 FPS | ✅ Pass | Measured average: 28.61 FPS | SRS 8.2 |
| CAP03 | Camera handles low-light conditions | ⏳ Pending | To be tested in dim lighting | SRS 7.1 |
| CAP04 | Face and eye region centered | ⚠️ Partial | Face mesh works, eye region not always centered | SRS 6.1.1 |
| CAP05 | Image preprocessing (grayscale, resize) | ✅ Pass | Output clean at 24x24 for model input | SRS 6.1.2 |

## 2.2 Processing Stage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Result | Comments | Reference |
| PROC01 | Model loads and classifies input | ✅ Pass | Model output: Closed, 99% confidence | SRS 6.1.2 |
| PROC02 | Detect eye closed > 2 sec | ✅ Pass | Alerts fired after 56+ frames closed | SRS 6.1.3 |
| PROC03 | Detect head tilt or bad posture | ❌ Fail | Accuracy dropped under head angle variation | SRS 7.1 |
| PROC04 | Ignore short blinks | ✅ Pass | No false alerts for 2–3 frame blinks | SRS 6.1.3 |

## 2.3 Alarming Stage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Result | Comments | Reference |
| ALRT01 | Trigger buzzer or LED | ✅ Pass | Simulated and ready for GPIO pin integration | SRS 6.1.3 |
| ALRT02 | Reset alert after eyes open | ✅ Pass | System resets after detecting open eye | SRS 6.1.3 |

## 2.4 System-Level Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Description | Result | Comments | Reference |
| SYS01 | Run for 2+ hours | ⏳ Planned | To be tested in final hardware setup. | SRS 8.2 |
| SYS02 | Power usage <500mA | ⏳ Planned | Pending multimeter testing. | SRS 8.2 |
| SYS03 | Offline functionality | ✅ Confirmed | System does not require internet connection. | SRS 8.3 |

# 3. Conclusion

## 3.1 Acceptance Criteria

To consider the implementation acceptable, the following vital processes must be tested and validated:  
• Real-time camera integration  
• Image preprocessing  
• Inference with trained model  
• Alert triggering  
• Offline capability

## 3.2 Acceptance Results

Most modules functioned as expected and were verified either through direct testing or simulation. Although some advanced features like posture adaptation and power profiling are incomplete, the system meets the primary project goals and can be considered a working prototype.

## 3.3 Success Performance

Total Test Cases: 13

Pass: 10

Partial/Pending: 2

Fail: 1

Success Rate: 10 / 13 × 100 = 76.9%

# 4. References

* 1. Computer Engineering SRS – Driver Drowsiness Detection System Khaled Alrefai
* 2. Software Design Document – Khaled Alrefai
* TensorFlow– Tool Configuration
* 5. IEEE 829 Software Test Documentation Standard