DriveSense^{TM} Final Project Report

Document Version: 2.2

Author: Keith Tran
Author E-Mail: ktran033@uottawa.ca
Author Department: Engineering
Author Division: University of Ottawa

www.eecs.uottawa.ca
2023F

Document Control Information

1.1 Document Approval

Name	Responsibility	E-Mail	Date
Saurav Guduru	*Scrum Master*	sgudu104@uottawa.ca	4/14/2025
Keith Tran	*Management Lead*	Ktran033@uottawa.ca	4/14/2025
Abdullah Ramadan	*CAD Specialist*	arama014@uottawa.ca	4/14/2025
Kevin Dang	*Web Developer*	kdang038@uottawa.ca	4/14/2025
Hajer Fguir	*Web Developer*	hfgui039@uottawa.ca	4/14/2025
Aaditya Shah	*Tech Lead*	ashah203@uottawa.ca	4/14/2025

1.2 Change History

Version	Date	E-Mail	Flag	Change Description
1.0	02/13/2025	Ktran033@uottawa.ca	INIT	Version 1.0 - Original Document
1.1	02/18/2025	Ktran033@uottawa.ca	CHG	Added to sections 3.1, 3.4, 3.7, 3.8, 3.11, 4.1 and 4.2
1.2	02/19/2025	Ktran033@uottawa.ca	CHG	Added to sections 3.6, 4.2, 4.3, 4.5, 5, 6.2, 8, 8.1.1, 8.1.2, 8.2, 9.2, 9.3, 9.4, 9.5, 9.6, 10. Merged sections 7.1 and (previously) 7.2 into 7.1
1.3	02/20/2025	Ashah2033@uottawa.ca	CHG	Added to sections 3.8, 7.1, 7.2, and 8.2
1.4	2/20/2025	Arama014@uottawa.ca	CHG	Added to sections 3.5, 3.8, and 6.1
1.6	2/24/2025	Sgudu104@uottawa.ca	CHG	Added sections 5.3 and 8.1
1.7	2/25/2025	hfgui039@uottawa.ca, kdang038@uottawa.ca, ashah2033@uottawa.ca, ktran033@uottawa.ca, arama014@uottawa.ca, sgudu104@uottawa.ca	CHG	Added to sections 3.4, 5.2, 8.1, 9
1.8	4/11/2025	Ktran033@uottawa.ca	CHG	Edited sections 5.2, 8.2, 8.3, 8.5
1.9	4/13/2025	Kdang038@uottawa.ca	CHG	Edited section 7.1
1.10	4/14/2025	Ktran033@uottawa.ca, sgudu104@uottawa.ca, kdang038@uottawa.ca, ashah2033@uottawa.ca, hfgui039@uottawa.ca	CHG	Edited sections 2.6, 5.3, 6.1, 6.2, 7, 7.1.1, 7.1.2, 7.2, 8.2
2.0	10/18/2025	Ktran033@uottawa.ca, sgudu104@uottawa.ca, kdang038@uottawa.ca, ashah2033@uottawa.ca, hfgui039@uottawa.cam,	CHG	Edited multiple sections
2.1	10/20/2025	Ashah203@uottawa.ca	CHG	Changed table of contents
2.2	12/12/2025	Ktran033@uottawa.ca	CHG	Edited sections 1.27.2, 1.30, 1.31, 1.32, 1.33

2.3	12/14/2025	Arama014@uottawa.ca sgudu104@uottawa.ca Kdang038@uottawa.ca	CHG	Edited sections 1.27, 1.28 Edited sections 1.24 Edited sections 1.23

Introduction

1.3 Project Description

This document provides an outline of the architecture and high-level design of DriveSense^{TM}, a device designed for detection of driver drowsiness, and its components. Specifically, this document describes, in detail, the roles and responsibilities of the parties involved in the project's development, the functional and non-functional requirements to be achieved by the project, the architecture of both the software and hardware modules, testing procedures, and the management of the project.

1.4 Goals, Objectives & Scope

The primary goal of the DriveSense^{TM} project is to develop a real-time system for monitoring the driver of a vehicle for signs of drowsiness, in the event of which it will either give a harsh warning or take direct action. The device will also allow a third party to monitor the driver, accessing information in real time. A key goal of the project is to keep the device small and adaptable, as to allow retrofitting in the anterior of any vehicle. The device also aims to be inobtrusive, having no impact on the driver's awareness nor their ability to pilot the vehicle. With these features, DriveSense^{TM} aims to increase driver safety and ease of access to critical information, especially for older vehicles that may lack modern safety measures and whose drivers are thus at greater risk.

The objectives of the DriveSense^{TM} project are as follows:

Design, planning, research, and project management:

- Formation of the project idea, key goals, functional and non-functional requirements, and constraints
- Designing the software and hardware architecture
- Picking and ordering components, research on drowsiness detection and relevant datasets

Software development

- Developing a web application designed for a third party to remotely monitor and interact with the system
- Developing an on-device application for interacting with the driver and/or passenger in the vehicle; serves as the interface between the driver and the system
- Integrating the system with an online database, where collected data can be uploaded to the cloud in real-time and can be accessed via the web application
- Integrating a ML model in the microcontroller that can analyze visual input and determine the presence of drowsiness
- Develop drivers for the various sensors that can store and process collected data

Hardware development

- Implement a microcontroller capable of processing visual and accessory inputs while simultaneously hosting an ML model capable of decision-making
- Implementing sensors for detecting symptoms of drowsiness, most notably a camera
- Integrating a touch-screen display that will serve as the interface between the user and the system

Integration

- Designing and creating CAD models to create a structure that will house and protect components and ensure compatibility between components and between the system and the vehicle's interior
- Ensure that individual components comply with the microcontroller and resolve any related issues that may arise
- Ensure that the system is adaptable so that it can be fitted into the interior of any vehicle without impairing the driver's awareness or ability to drive

Testing, prototyping, and feedback

- Create prototypes and perform testing on various aspects of the device with an emphasis

on reliability and safety
- Rigorous testing and debugging of the web application and to ensure smooth functionality and an accessible UI
- Collect external user feedback to gather a variety of perspectives, especially from a consumer standpoint

Documentation
- Comprehensively document every aspect of the project, including the internal design and development process, as well as detailed descriptions and user manuals concerning the product itself
- Store all deliverables, files, and documents to a central database that is shared by all members of the project and can be freely accessed on the web, with back-ups in the event of system failure

The scope of this document is to provide a basis for the design and implementation of the DriveSense™ project. This prototype stems from the set of requirements obtained and listed in Section 3.6.

1.5 Assumptions, Constraints & Risks

Assumptions:
- The ideal user regularly drives a vehicle that they have personal access to
- The user lacks a vehicle with internal safety detection measures as well as real-time remote monitoring
- The device will sustain minimal environmental resistance, as it will not encounter factors unusual to the internal environment of a vehicle
- The user has access to the IoT devices

Constraints:
- The design will have to be flexible to be able to fit into the internals of various vehicles of different dimensions
- The budget should be kept low to keep the price low as a consumer incentive

Risks:
- Network failure
- Power failure
- Microcontroller malfunction
- Overheating
- Failure to send messages

1.6 Project Deliverables

The deliverables of the project include the project proposal, this project report, the mid-term and final presentations, a working device with both hardware and software components and a web application.

1.7 Schedule & Budget Summary

Tentative schedule and budget estimate for entire project (4912 and 4913)

Component	Cost (CAD$)	Link
Nvidia Jetson Orin Nano Super Developer Kit	480.85	https://www.arrow.com/en/products/945-13766-0000-000/nvidia

PI NOIR CAMERA V2 IMX219 8MP	27.06	https://www.digikey.ca/en/products/detail/raspberry-pi/SC0024/6152811?gQT=2
4.3inch HDMI LCD 800x480 IPS Capacitive Touch Screen	73.44	https://a.co/d/5GLenRh
DC 3.3-5V Passive Low-Level Trigger Buzzer Alarm Sound Module	11.29	https://a.co/d/fjKHHs3
30cm Camera Cable for Raspberry pi High Flexibility Ribbon FFC Flat Cable Wire for Raspberry Pi Zero V1.3 Cameras	15.36	https://a.co/d/fkIsxPF
GY-521 MPU-6050 MPU6050 Sensor Module 3 Axis Accelerometer and Gyroscope Module	18.07	https://a.co/d/gGHhDEE
Thin Film Pressure Sensor Flex Bend Sensor SF15 600 10kg	28.02	https://a.co/d/0nDhxGG
Digital ADC Module 16-Bit ADS1115 I2C 4-Channel ADC	18.07	https://a.co/d/h63XINY
Heart Rate Sensor Module MAX30102 Pulse Detection Blood Oxygen Concentration	43.93	https://a.co/d/ftirkyS
4 Channels IIC I2C Logic Level Converter Bi-Directional Module 3.3V to 5V Shifter	14.45	https://a.co/d/1hsF3Fc
Waveshare Binocular Camera Module Dual IMX219 8 Megapixels for AI Vision	90.16	https://a.co/d/hT3ZUZ2
Thin Film Pressure Sensor, 4pcs, 20g-2Kg, High Sensitivity	29.18	https://a.co/d/goZWs3v
Miniature Loudspeaker 5 Watt 8 Ohm Passive Enclosed Audio/Woofer Speaker Compatible	22.01	https://a.co/d/2AgNpzs
3.3V 5V Power Supply Module	18.07	https://a.co/d/a0hRpd0
eSUN PLA Filament 1.75mm	20.33	https://a.co/d/75VbWkC

Lexar 128GB Micro SD Card	21.46	https://a.co/d/55V5tDI
150W Power Inverter DC 12V to 110V AC Converter	29.82	https://a.co/d/gHVZSVV
Total Expenses: $956.61		

1.8 Acronyms, Terms and Definitions

Terms, acronyms, abbreviations	Definitions
AI	Artificial intelligence
ML	Machine learning
EEF	Engineering Endowment Fund
CAD	Computer-aided design
FPS	Frames per second
IMU	Inertial measurement unit
QA	Quality assurance
CRUD	Create, read, update, and delete
PWM	Pulse width modulation
UI	User interface
SBC	Single board computer
ADC	Analog-to-digital converter
LTE	Long term evolution
GSM	Global system for mobile communications
API	Application programming interface
OBD	On-board diagnostics
IoT	Internet of things
IR	Infrared
PPG	Photoplethysmography
REST	Representational state transfer
JSON	JavaScript Object Notation
SEG	Software engineering
CNN	Convolutional neural network
EAR	Eye aspect Ratio

Table 1: Table of Definitions, Acronyms and Terms

1.9 References

This document contains no references to outside material.

1.10 Limitations, Issues and Concerns

Some limitations, issues, or concerns that have been or are anticipated to be experienced are:

- Availability of an optimal microcontroller: There was difficulty in finding a microcontroller capable of meeting the requirements of the project while remaining affordable relative to the project's budget, as well as available on the market. Eventually, the Nvidia Jetson Orin Nano was found, and was obtained after direct contact with a supplier at a price amicable with the project's budget.
- Implementing AI model training: Implementing AI model training: Collecting and relevant datasets for drowsiness detection is time-consuming and requires many different data points such as different lighting conditions, facial features, and angles. Additionally, optimizing the AI model to run efficiently on edge devices like the Nvidia Jetson Orin Nano is a challenge. The Nvidia Jetson Orin Nano acts as an edge device because it runs the AI model locally in the car, processing video input from a camera to detect signs of drowsiness in real time.
- LTE/GSM connection: Due to the development time required for cellular technology, the prototype must use Wi-Fi in lieu of an LTE/GSM connection. An API will also be used to simulate calls.
- CAD design in relation to different car models: ensuring the system fits seamlessly into various car interiors presents challenges. Differences in dashboard layout, mounting points, and space constraints require adaptable or modular CAD designs.
- Driver hand position: The system includes a sensor that requires the driver to keep at least one hand on the steering wheel at all times.
- Working cigarette tray: We plan to power our system through the car's cigarette tray (12V socket), which presents some challenges. Some vehicles may have limited space around the socket. Additionally, older car models may not provide a stable power supply, potentially causing interruptions. Ensuring consistent power delivery and managing heat dissipation will be important considerations for system reliability.

1.11 Change of information

The information in this document is for informational purposes and may change at the sole discretion of CEG4912-13 without notice.

1.12 Confidential Information

This documentation contains confidential regarding DriveSense^{TM} design specification information and is purely intended for the University of Ottawa and not for release/disclosure in whole or in part to any other party unless agreed upon in writing by CEG4912-13, Inc.

1.13 Third Party Confidentiality Restriction

There are no third-party confidentiality restrictions concerning the DriveSense^{TM} project.

Roles & Responsibilities

1.14 Objective

The objective of the DriveSense^{TM} project is to create a device that can be fitted inside a vehicle and detect drowsiness in the driver based on stimuli measured internally to the vehicle and provide warning or emergency assistance if deemed necessary.

1.15 Project Stages

- **Initialization:** the idea of the project is created, and a project proposal is submitted and approved. Research is conducted to produce clear objectives for the project.
- **Design:** The design of the project is created and finalized, and the necessary components

are identified and requisitioned. Functional and non-functional requirements are identified. Funding for the project is secured.

- **Management:** roles and responsibilities are identified and assigned to project members. Project objectives are broken down into tasks, which are scheduled for completion throughout the project period.
- **Development, testing, and prototyping:** completion of tasks involving development, testing, and prototyping of software and hardware components is carried out in SCRUM development cycles.
- **Finalization and presentation:** Completed components are assembled and tested for compatibility. A presentation and demonstration of the final product is prepared, along with detailed and extensive technical documentation logging the development process and the product itself.

1.16 Clients

The following is a list of clients being satisfied by the project, accompanied by background information and their vested interest in the project:

- Professor Dan Ionescu: the professor of the course CEG4192, and the one responsible for grading the project and the students responsible based on their efforts.
- Course teacher's assistants: assist the professor in tracking the project's progress, giving feedback, and marking deliverables.

1.17 Participants

Person	Org.	*Contact*	Role	Contribution	CEG4912-13 Liaison
Saurav Guduru	uOttawa	sgudu104@uottawa.ca	*Scrum Master*	Project development	Ahmed Hassanein
Keith Tran	uOttawa	Ktran033@uottawa.ca	*Management Lead*	Project development	Ahmed Hassanein
Abdullah Ramadan	uOttawa	arama014@uottawa.ca	*CAD Specialist*	Project development	Ahmed Hassanein
Kevin Dang	uOttawa	kdang038@uottawa.ca	*Web Developer*	Project development	Ahmed Hassanein
Hajer Fguir	uOttawa	hfgui039@uottawa.ca	*Web Developer*	Project development	Ahmed Hassanein
Aaditya Shah	uOttawa	ashah203@uottawa.ca	*Tech Lead*	Project development	Ahmed Hassanein

Table 2: Project Participants

1.18 Market Space and Industry Sector

This project aims to target the smart assistive technology market. It targets consumers who are concerned about safety but for whom it is inconvenient or out of their price range to purchase a new vehicle with similar pre-existing technology. It also finds its place in the burgeoning industry of AI, using an ML model to help determine drowsiness and make decisions based on gathered data in real time. This industry is one that is disruptive and constantly changing and will require adaptability for the product to stay relevant and competitive in the market.

Requirements

1.19 Functional Requirements

- The system must detect when the driver is drowsy.

- The system must alert the driver if drowsiness is detected.
- The system should notify a third party (emergency contact or healthcare provider) if it detects that the driver is drowsy or in distress.
- The system should store all logs/event details in the cloud.
- The system should retrieve logs from the cloud for third-party access upon request.
- The system should display information about the driver to any third parties.
- The user must authenticate the system to their user management account.
- The user-interface must contain an embedded video for every major event that occurs.
- The system must allow user to immediately reach out to the emergency contact.
- The user must be able to dismiss alerts.

1.20 Non-Functional Requirements

- The internal system should be protected and hidden from the user.
- The system should be easy to install.
- The software UI should be simple to navigate.
- The system's UI should use O-Auth to allow users to login.
- The system emergency contact feature should notify the contact within 2 seconds.

1.21 Constraints

- The driver must hold the steering wheel at the 10 and 2 position.
- The system cannot obstruct the driver's view of the windows or dashboard.
- The sensors must not be obstructed from detecting the driver (e.g. face coverings).
- The system must have Wi-Fi connectivity to upload events/logs.

Software Architecture

The end product will feature a website accessible on the internet for external management and monitoring of the device, with a database storing user information and event logs. An application for the display screen interacts with the user directly, giving information and warnings based on gathered data.

1.22 High Level Design: Architectural Views

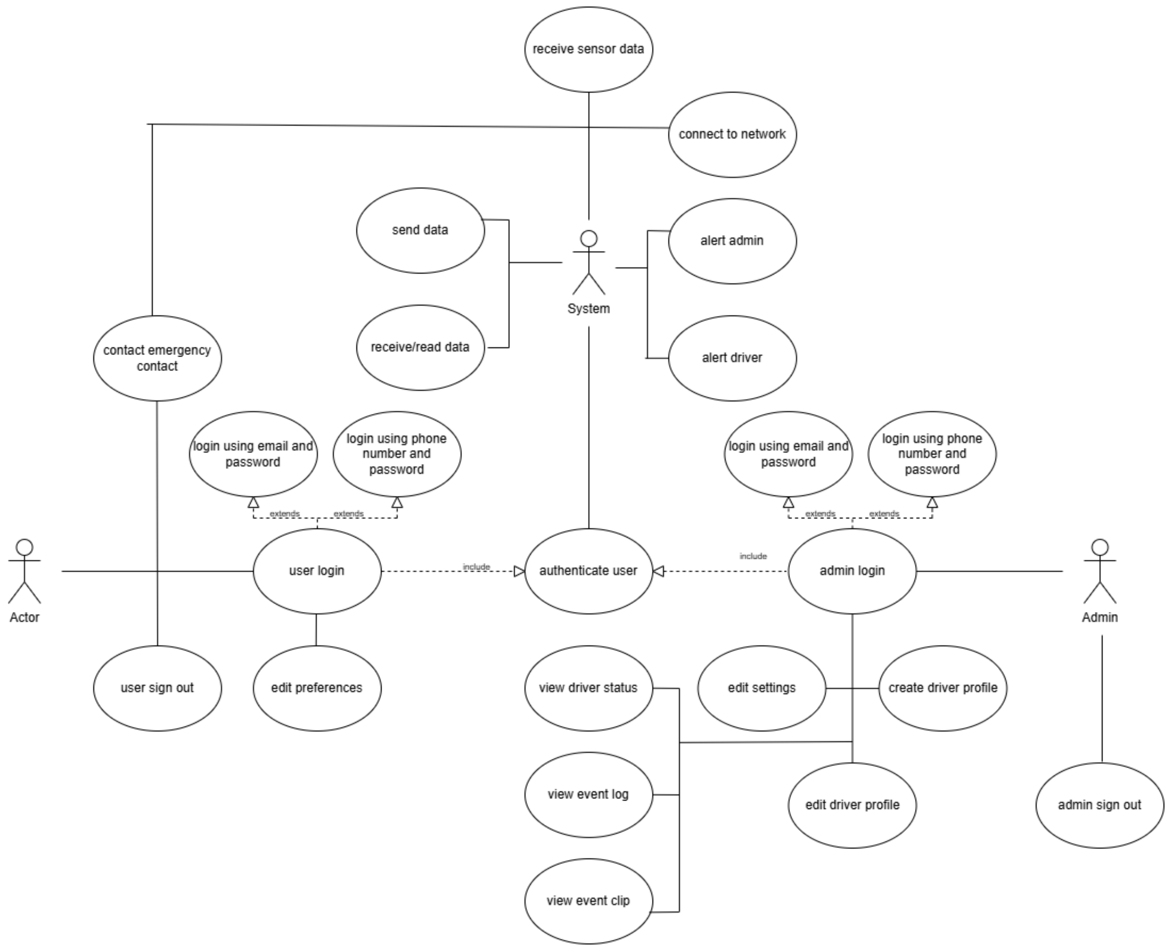


Figure 5.1: UML use-case diagram

The architectural design of the system is represented by using UML diagrams to illustrate system interactions and functionalities. The Use Case Diagram (as shown in the provided image) highlights key actors, including the User, Admin, and System, and their respective interactions. The system receives sensor data, processes it, and performs critical functions such as alerting the driver and administrator in case of drowsiness detection. User authentication is facilitated through email or phone-based login, enabling access to features like viewing driver status, event logs, and recorded event clips. Admins can manage driver profiles, edit system settings, and oversee event records. Additionally, the system integrates network connectivity for data transmission and emergency contact alerts.

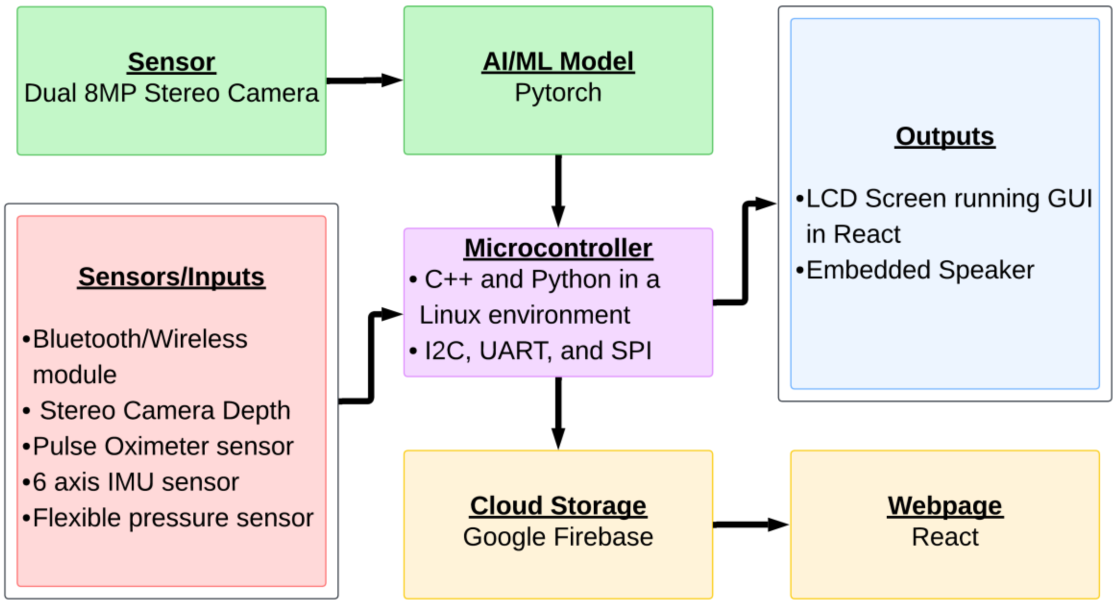


Figure 5.2: Software Architecture

The software architecture of the system is designed to help with interaction between sensors, AI processing, cloud storage, and user interfaces. As illustrated in the provided software architecture diagram, the system consists of multiple components working together.

1.23 Front-End (Graphical User Interface)

- ***The Website:***

The web-based dashboard of DriveSense^{TM} was developed to provide a comprehensive cloud-accessible platform for administrators to monitor and manage drivers in real-time. Built with React.js and styled with CSS, the website offers an intuitive and responsive interface for desktop browsers. The architecture takes advantage of React Router for seamless navigation between the various pages while maintaining a consistent user experience.

The dashboard integrates with Firebase FireStore through secure REST API endpoints, enabling real-time synchronization of driver metrics and event logs. When the model detects drowsiness, sensor data including heart rate, blood oxygen levels, and vehicle speed are automatically captured and stored as an event in the database. The driver's status is continuously polled from the backend and is reflected through color-coded indicators on the dashboard. By clicking on a specific driver card, administrators can view live sensor data of a driver. On the same page, each event can be exported as a PDF report containing the event timestamp and associated driver metrics.

To improve useability, the dashboard features a dynamic search bar and a sorting menu which allows drivers to be organized by alphabetical order, chronological order, current status, and driving activity. These sorting options can be combined with the search functionality to quickly locate a specific driver or identify drivers who need assistance.

User authentication is implemented through Firebase; each account has its own unique ID which allows users to view and manage their own associated drivers. The platform includes account management features such as editing profile information, update driver details, manage emergency contacts, and reset passwords. The driver management supports CRUD operations enforced with user ID-based authorization, preventing unauthorized access to sensitive data.

Communication features include a contact form that allows users to send inquiries, feedback, or questions directly to the DriveSense team via EmailJS, ensuring users have access to staff for assistance without leaving the platform. Overall, the website delivers a complete cloud-based management solution that connects hardware sensor integration, live data visualization and incident reporting - providing users with the tools to monitor drivers and reduce accident risk.

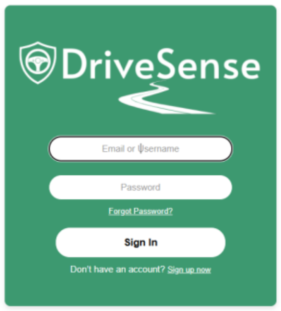


Figure 5.3: Login page. Entering an email/username and password, then clicking “Sign In” will sign the user into their account if the provided information is correct. Clicking “Forgot Password” will take the user to a page to create a new password. Clicking “Sign up now” will take the user to a page to create a new account.

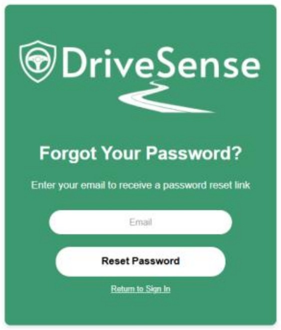


Figure 5.4: Forgot password page. The user can enter their email address to receive a link to

reset their account password.

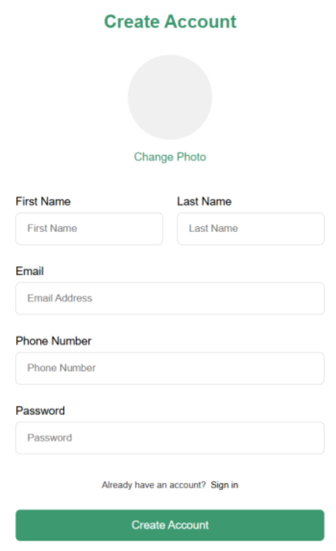


Figure 5.5: Create account page. The user can create a new account here.

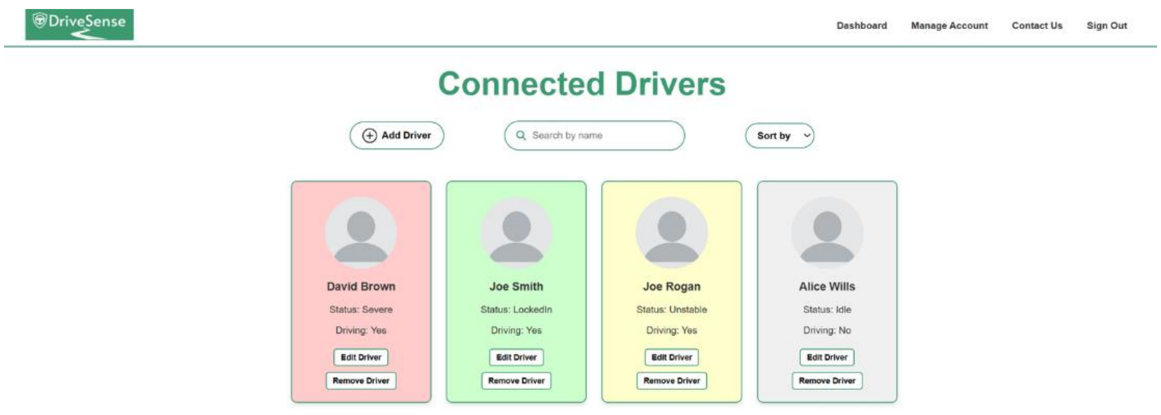
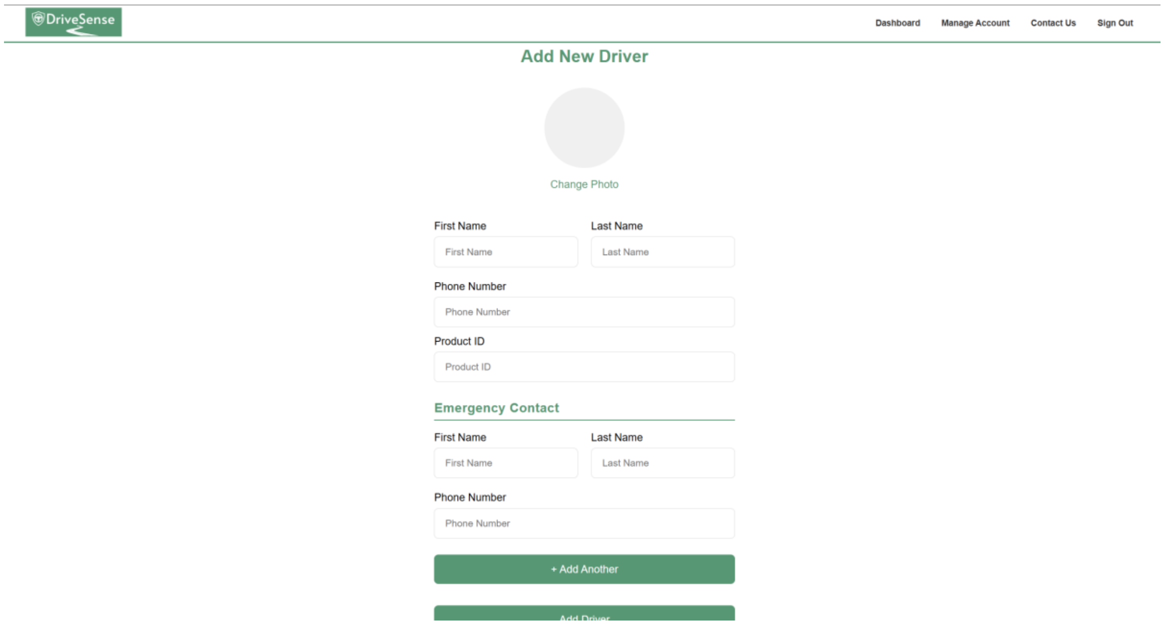


Figure 5.6: Connected drivers page. Clicking “Add Driver” will take the user to the “Add New Driver” page. Entering keywords into the search will filter profiles based on user input. The dropdown will sort profiles based on a chosen attribute. Clicking the options in the navigation bar will navigate to each respective page. Clicking “Edit Driver” button will take the user to the Edit driver page. Clicking “Remove Driver” will remove the corresponding driver.



Add New Driver

Change Photo

First Name | Last Name
First Name | Last Name

Phone Number
Phone Number

Product ID
Product ID

Emergency Contact

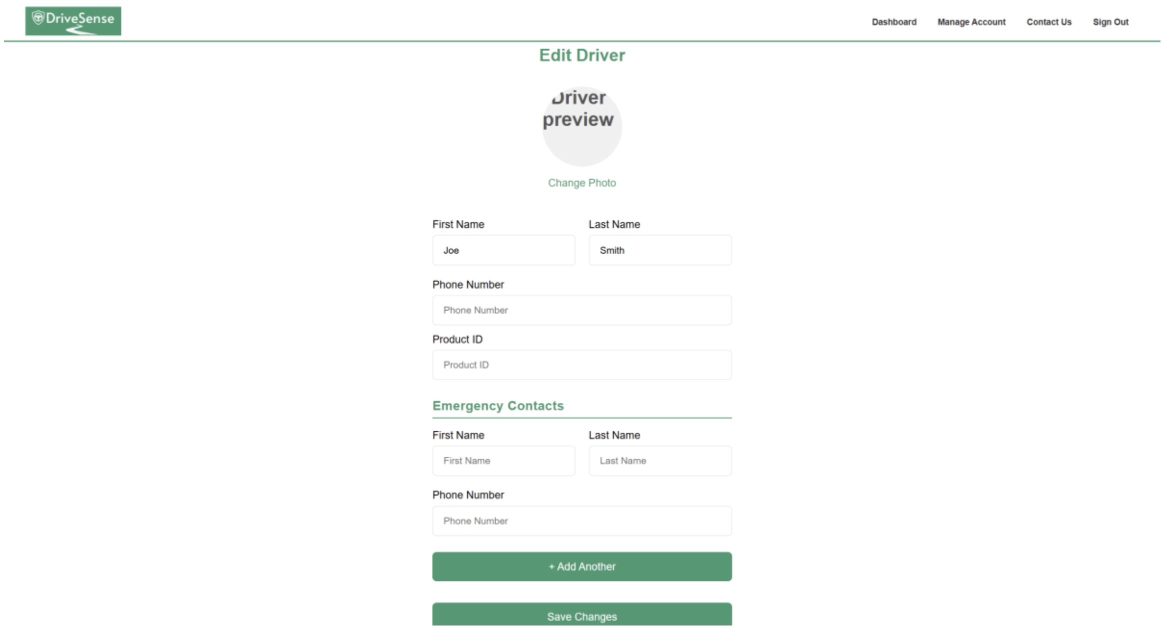
First Name | Last Name
First Name | Last Name

Phone Number
Phone Number

+ Add Another

Add Driver

Figure 5.7: Add new driver page. The user can enter a driver's personal informatiom. Clicking "Add Driver" will save the driver profile to the database and display its corresponding card on the dashboard.



Edit Driver

Driver preview

Change Photo

First Name | Last Name
Joe | Smith

Phone Number
Phone Number

Product ID
Product ID

Emergency Contacts

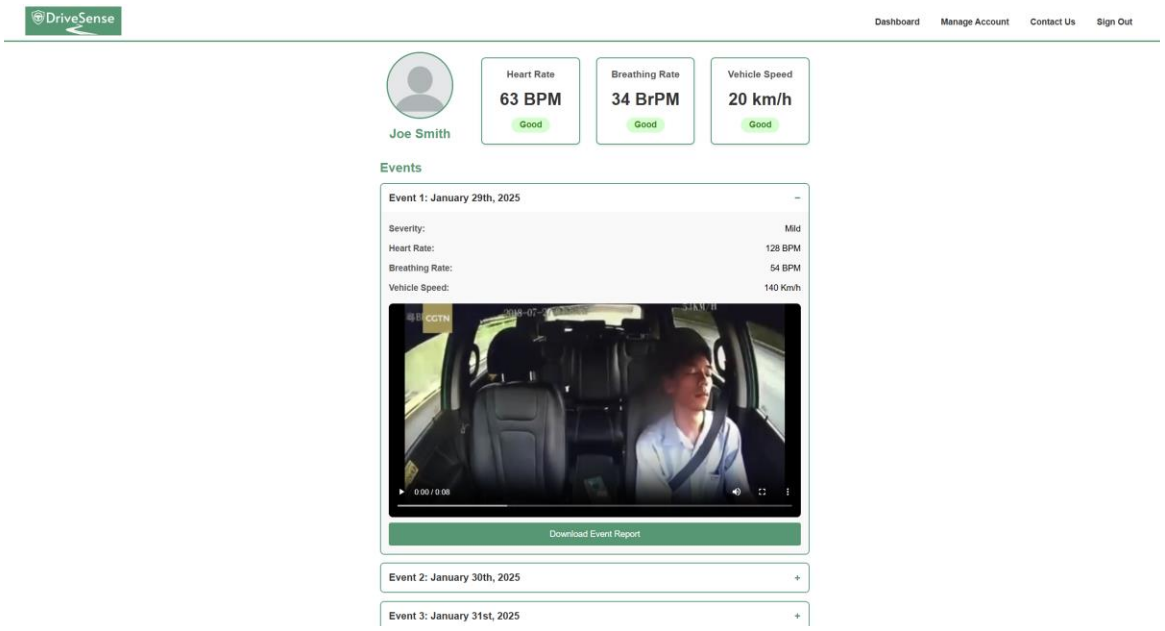
First Name | Last Name
First Name | Last Name

Phone Number
Phone Number

+ Add Another

Save Changes

Figure 5.8: Edit driver page. The driver's preexisting details populate the input fields, and the user can make edit to these fields, which will save after clicking "Save Changes."



DriveSense

Dashboard Manage Account Contact Us Sign Out

Joe Smith

Heart Rate	Breathing Rate	Vehicle Speed
63 BPM	34 BrPM	20 km/h
Good	Good	Good

Events

Event 1: January 29th, 2025 –

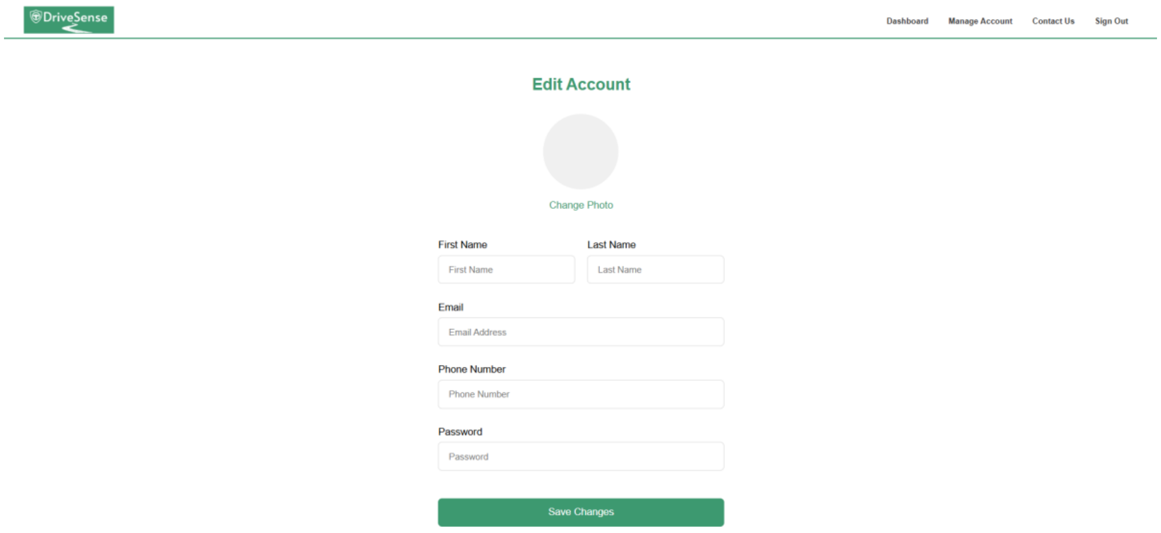
Severity:	Mild
Heart Rate:	128 BPM
Breathing Rate:	54 BPM
Vehicle Speed:	140 Km/h

Download Event Report

Event 2: January 30th, 2025 +

Event 3: January 31st, 2025 +

Figure 5.9: View driver card page. This shows the current status information of the driver, along with a list of event logs and their recorded details.



DriveSense

Dashboard Manage Account Contact Us Sign Out

Edit Account

Change Photo

First Name: First Name

Last Name: Last Name

Email: Email Address

Phone Number: Phone Number

Password: Password

Save Changes

Figure 5.10: Edit account page. The user can make changes to their own account details.

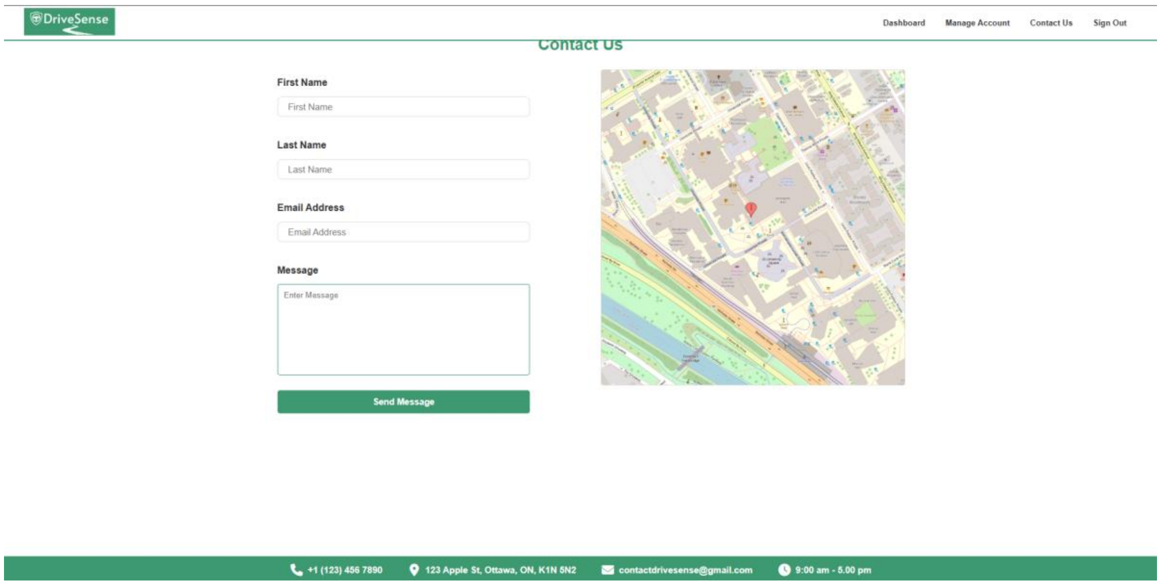


Figure 5.11: Contact us page. The user can submit a form to the team, which will send an email containing the information and message provided in the input fields. A confirmation email will be sent to the address confirming the reception of the message.

- ***MOCK UI FOR THE SMALL EMBEDDED SCREEN:***

The new embedded user interface of DriveSense^{TM} was developed to deliver a robust, responsive, and driver-focused experience for the DriveSense system. Built with React.js and TypeScript, the interface provides a lightweight and modular architecture optimized for deployment on embedded hardware such as the Jetson Nano. The migration from Electron.js to React significantly reduced resource usage while improving maintainability and performance.

The interface integrates real-time driver state monitoring through secure REST API communication with backend services connected to Firebase Firestore. Driver status (Stable, Mild, Critical) is stored directly in the drivers collection and continuously retrieved by the frontend, enabling immediate UI updates when the driver's condition changes. This ensures that critical alerts are displayed without delay and remain synchronized with live system data.

System customization features include light and dark mode switching, as well as screen brightness and alarm volume controls. Brightness is managed through a dynamic overlay rendered at the UI level, allowing real-time visual adjustment without requiring direct hardware display control. Audio alerts are implemented using the HTML5 Audio API, with volume settings centrally managed via a React Context and automatically triggered during critical driving states.

The interface also integrates Twilio-based emergency communication, enabling automatic SMS notifications and phone calls when a critical alert is detected. User authentication and session handling are secured through Firebase Authentication, ensuring that each user only accesses their associated drivers and data. Overall, the completed UI provides a scalable, real-time, and safety-oriented solution that seamlessly connects sensor data, backend services, and the embedded driver display.

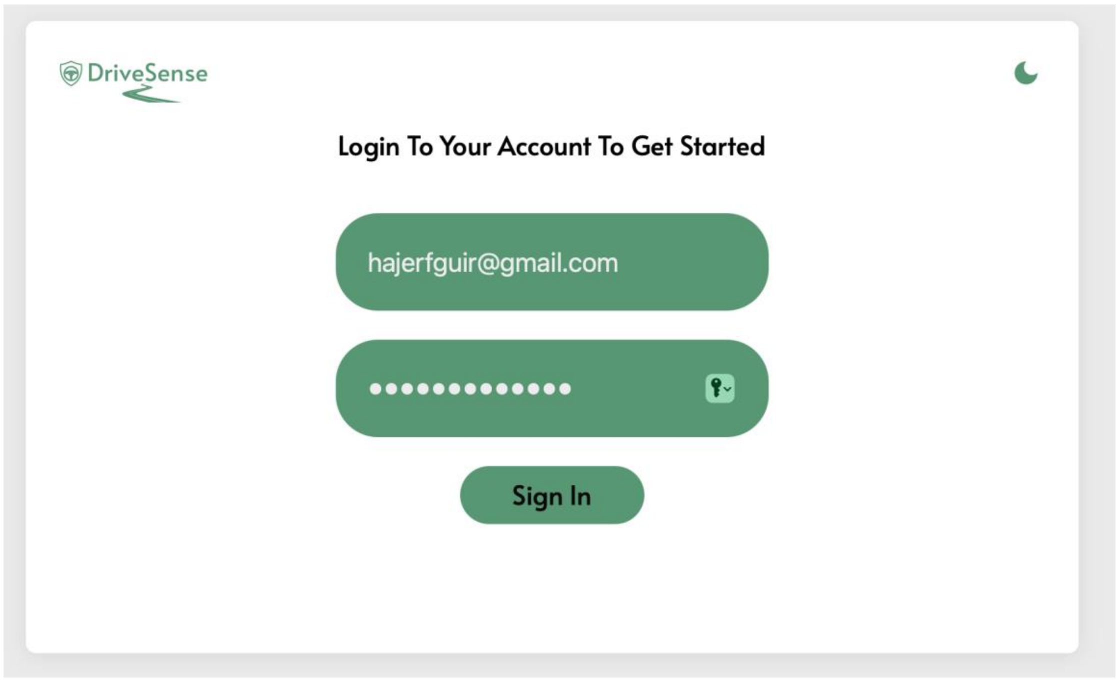


Figure 5.12: Login interface
Simple login interface that initializes driver authentication and transitions to driver selection.

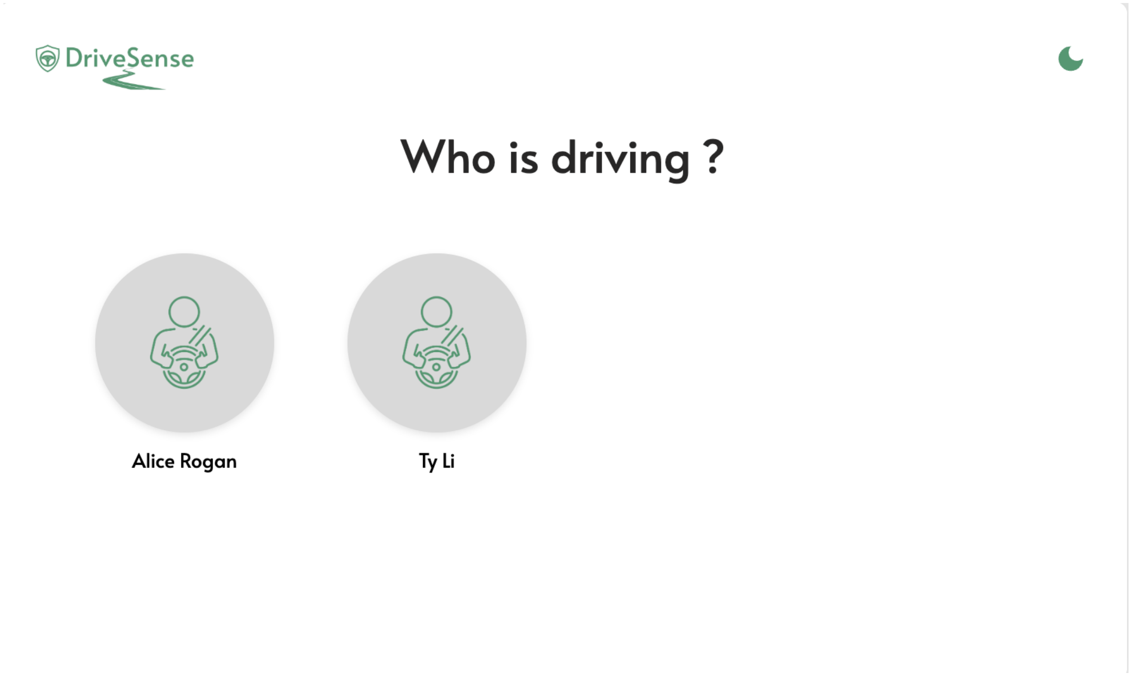
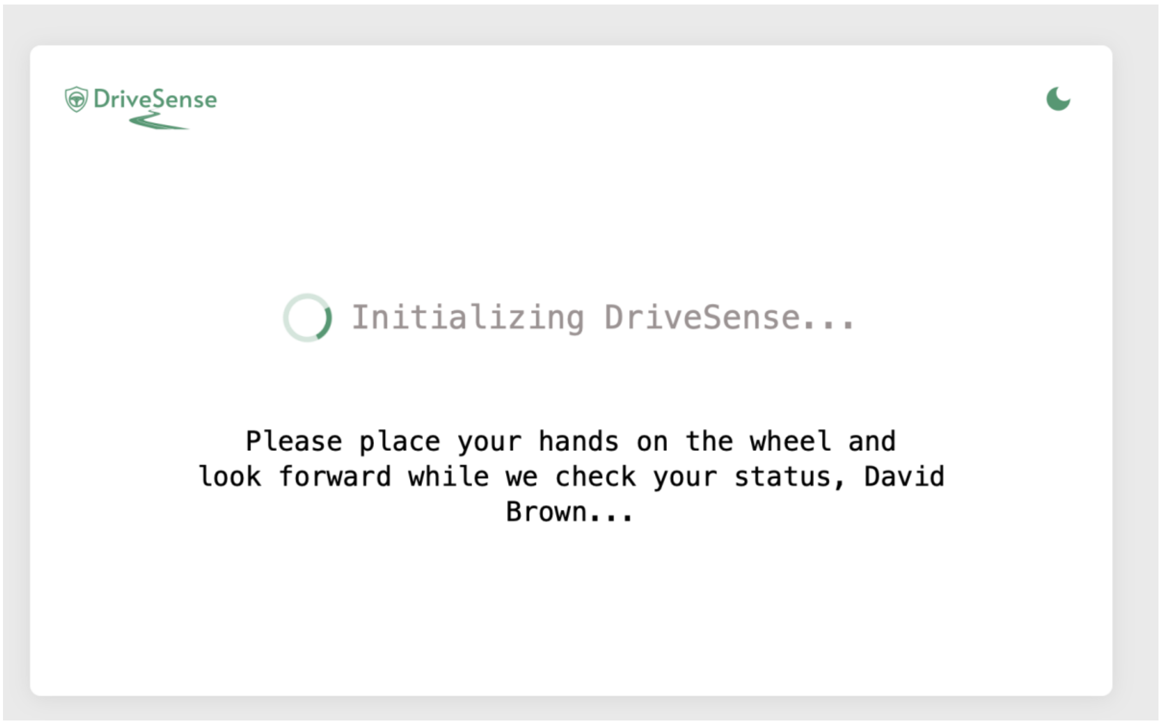


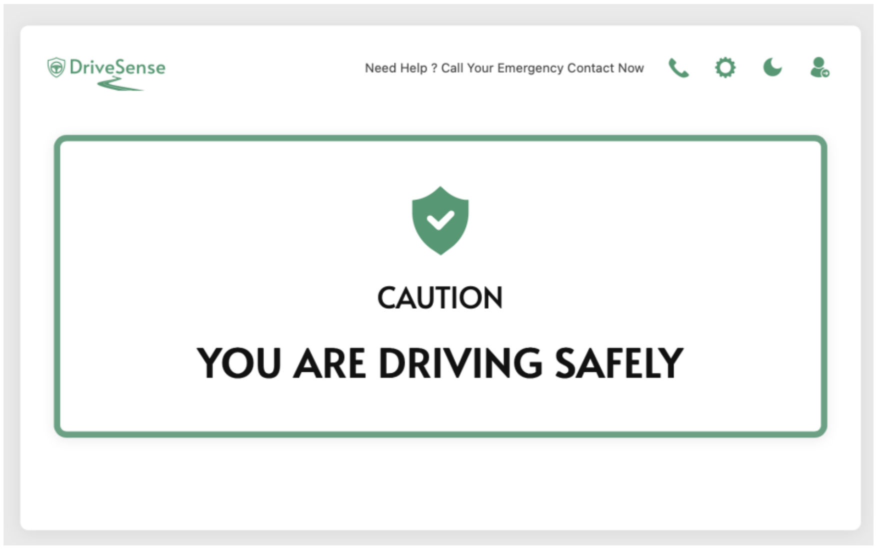
Figure 5.13: Driver Selection Page
Lists available drivers and starts the monitoring session once a driver is selected.



Initializing DriveSense...

Please place your hands on the wheel and look forward while we check your status, David Brown...

Figure 5.14: Processing Page
Displays system initialization or active monitoring states before alert mode.



CAUTION

YOU ARE DRIVING SAFELY

Figure 5.15: Normal State
Indicates safe driving conditions with no signs of fatigue detected.

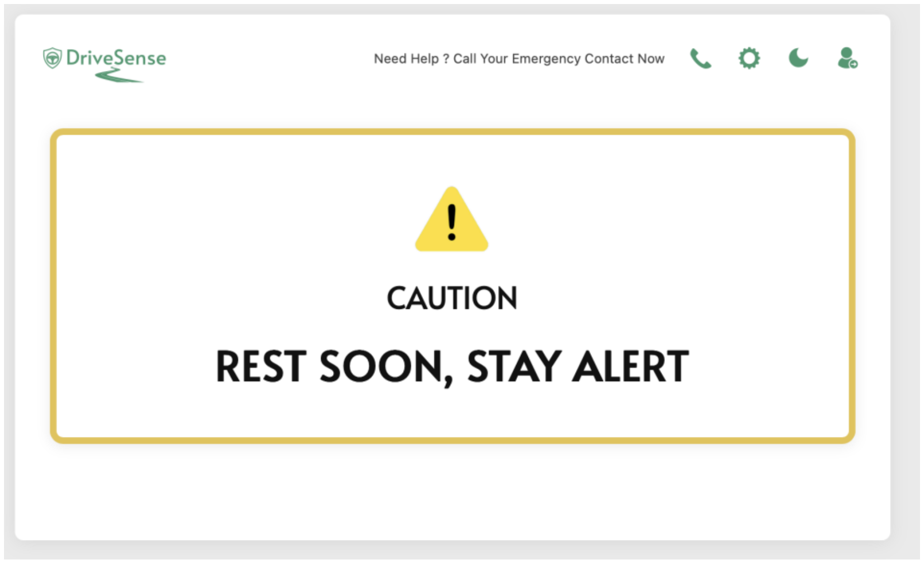


Figure 5.16: Drowsy State
Displays a yellow warning advising the driver to rest soon and stay alert.

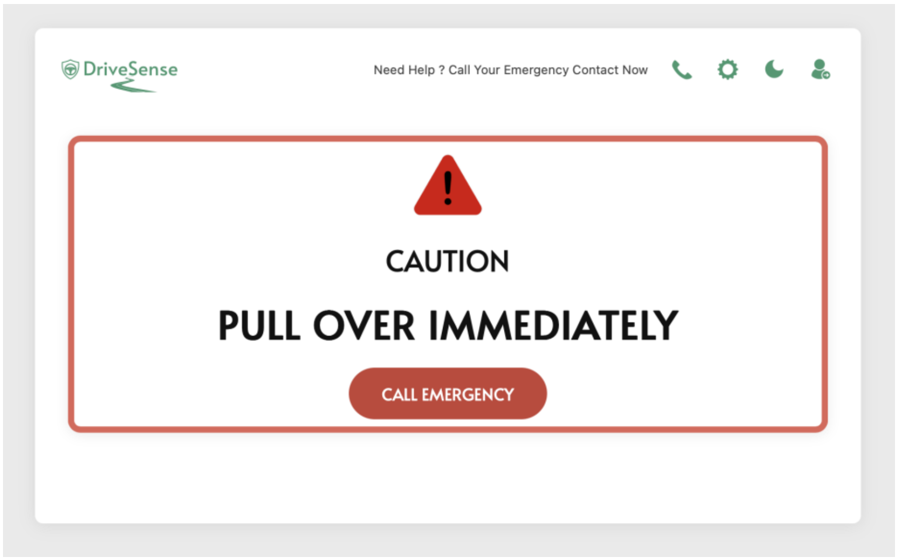


Figure 5.17: Critical State
Issues a red alert instructing the driver to pull over immediately and Call Emergency option

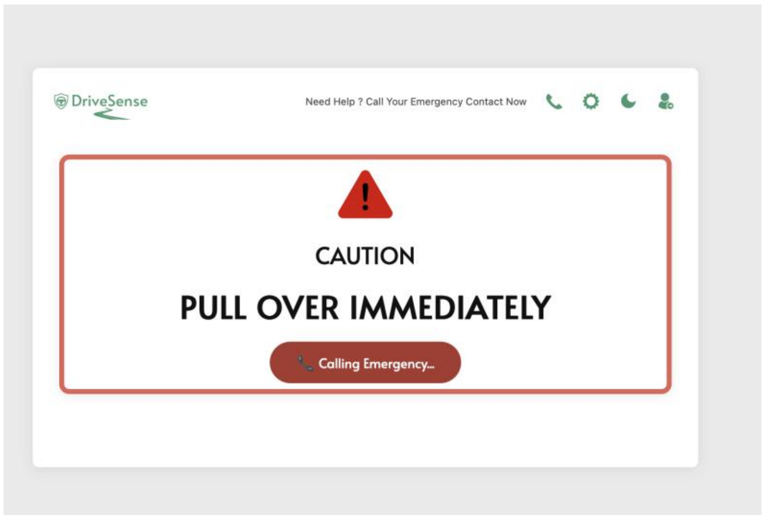


Figure 5.18: Emergency Call in Progress
Confirms that the system is contacting the emergency contact.

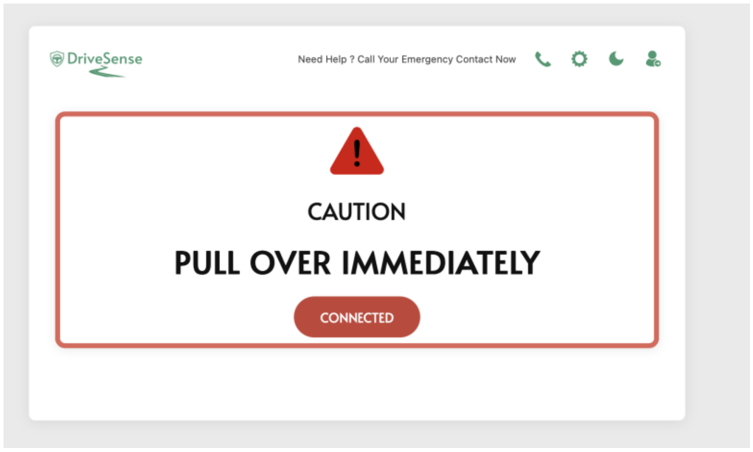


Figure 5.19: Emergency Connection Established
Shows that the emergency contact has been successfully reached.

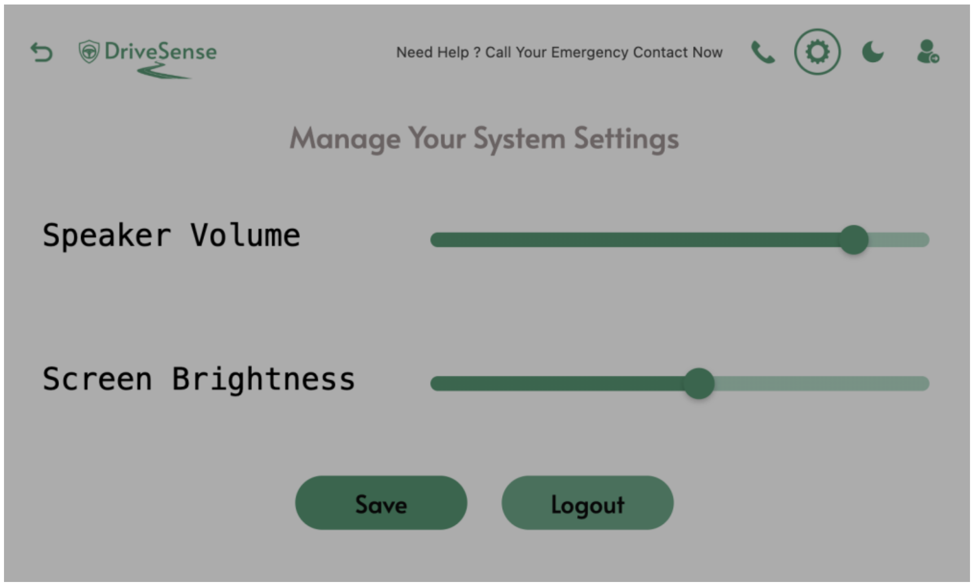

DriveSense
Need Help ? Call Your Emergency Contact Now
Manage Your System Settings
Speaker Volume
Screen Brightness
Save
Logout

Figure 5.20: Settings Page
Allows adjustment of speaker volume and screen brightness for driver comfort.

1.24 Back-End

- ***Cloud Integration- Firebase:***

For DriveSense™, we utilize Firebase as our backend service due to its scalability, real-time capabilities, and seamless integration with mobile and embedded applications. Firebase also provides a NoSQL cloud database (Firestore) that efficiently manages and synchronizes data across multiple devices in real-time. The system's embedded hardware is powered by a Jetson Nano, which processes sensor data locally before transmitting relevant information to Firebase Firestore. We decided to store logs containing relevant sensor data to display on our Web UI in events when we detect drowsiness.

In order to improve communication between our system components, we developed custom REST APIs for interacting with our database. After their implementation, we integrated these APIs into our architecture, allowing external applications to securely add, modify, and remove data. This API layer enhances modularity by decoupling the frontend, backend, and hardware, while providing better control, security, and performance.

We opted to go with Firebase's Firestore and Firebase Cloud Storage instead of other databases for the reasons as follows:

1. Firebase Cloud storage is optimized for handling multimedia files, which makes it the ideal choice for storing short video clips that were captured during the drowsiness detection event. The logs consist of sensor data (such as heart rate, ppg, speed) are stored in Firestore, allowing for structured retrieval and analysis of the data.
2. Firebase provides out of the box solutions that allow direct communication from the Jetson Nano, which helps simplify the process of uploads and retrievals. Any other alternatives such as AWS or DynamoDB would require additional setup and IAM configurations, increasing the development overhead.
3. Firestore ensures that event logs and metadata are updated instantly across devices. This real-time capability is crucial as it allows for immediate updates to occur on our web UI, notifying the user immediately after the event.

4. Firebase automatically scales based on usage, ensuring reliable performance even with large video storage needs. Additionally, Firebase Authentication enables secure access to stored logs and video files.
5. Firebase comes with built-in authentication services (e.g., Firebase Auth), allowing secure access control without requiring a custom authentication implementation.

The Jetson Nano continuously monitors driver behavior using onboard sensors and machine learning models. When drowsiness is detected the following sequence of events in relation to the database and the system carry out:

1. Sensor data (heart rate, PPG, speed) along with camera input is processed by Jetson using onboard AI models, and if drowsiness is detected a short video clip recording begins. This recording will continue for up to one minute. All sensor data during period is saved locally initially in a JSON file format (log rotation and auto-detection policies will be set in place to prevent excessive storage consumption).
2. This locally saved data is then transmitted to the cloud after the one-minute period has elapsed. The sensor data is sent to Firebase Firestore via a Firebase REST API that will be developed by us.
3. The driver dashboard on our website will be automatically updated, each driver's status is updated and any data related to their events is available to view.
4. Firebase cloud functions will be used in addition to manual API updated to our web dashboard, we will trigger alerts and send notifications automatically to the emergency contact registered depending on the severity of the driver's data.

Our team has prior experience using Firebase in past courses (SEG 2105), making it an ideal choice for DriveSense^{TM}. This familiarity allows us to efficiently develop and optimize the backend infrastructure of our software without a steep learning curve. Also, Firebase has a very large developer community with extensive documentation, making troubleshooting and feature development manageable compared to other cloud service options.

- ***<u>AI Model:</u>***

As part of the backend processing for DriveSense™, a refined AI-powered drowsiness detection system was developed and deployed to analyze live facial video streams and assess the driver’s alertness in real time. The system classifies the driver’s state as “Normal,” “Drowsy,” or “Critical,” and communicates this information to the application layer to trigger alerts, emergency functionality, and event logging. Compared to the preceding implementation, the final system introduces a more robust deep learning architecture and improved facial feature extraction optimized for embedded deployment.

MediaPipe Face Landmarker v2 was used to extract precise facial landmarks from live camera input. Rather than relying on isolated facial features, the final approach extracts a focused facial region that includes both the eyes and mouth—areas most indicative of drowsiness-related behavior such as prolonged eye closure and yawning. Key landmarks spanning the forehead, eyes, cheeks, mouth, and chin are used to generate a padded region of interest, which is resized to a fixed resolution to preserve fine facial details while reducing background noise.

Drowsiness classification is performed using a YOLOv8 Nano (YOLOv8n) image classification model, fine-tuned on a 10000 image dataset from Kaggle, an online collection of ML datasets. The dataset contains two image sets of faces displaying alertness and drowsiness. The dataset was preprocessed by extracting landmark-based facial regions and cropping the image to just contain the faces to ensure that the model makes accurate predictions. The model was trained for 40 epochs using data augmentation and early stopping (a technique that prevents overfitting by stopping training when validation performance starts to degrade), achieving reliable performance while maintaining low-latency inference suitable for real-time use on embedded hardware. This approach replaces the earlier standalone CNN model used by the previous prototype, providing improved efficiency and scalability.

The output of the YOLOv8 model is passed to a meta-level event detection algorithm that aggregates predictions over time to reduce false positives and identify sustained fatigue patterns. This temporal reasoning enables the system to distinguish between normal behaviours, such as blinking, and critical drowsiness events. The finalized model runs locally on the Jetson Nano as a backend service and uses a publisher–subscriber architecture to distribute detection results. Integration with Firebase and REST APIs ensures synchronization across embedded and web-based interfaces.

The model demonstrates strong and stable performance, as reflected in **Figure 5.24** and **Figure 5.25**. The normalized confusion matrix shows a high true positive rate for both classes, with 96% of no-yawn (alert) samples and 98% of yawn (drowsy) samples correctly classified. Misclassifications are minimal and primarily occur between adjacent behavioural states, which is expected given the subtle visual differences between brief mouth movements and full yawns. This indicates that the model generalizes well and maintains a low false-alarm rate, which is critical for real-world driver monitoring systems.

Training and validation loss decrease consistently over the 40 training epochs, converging to low final values with no significant divergence between training and validation loss. This suggests effective learning without severe overfitting. The top-1 accuracy (the accuracy of the model's best prediction) rapidly increases during early epochs and stabilizes at approximately 97%, while top-5 accuracy remains at 100%, further confirming confident class separation.

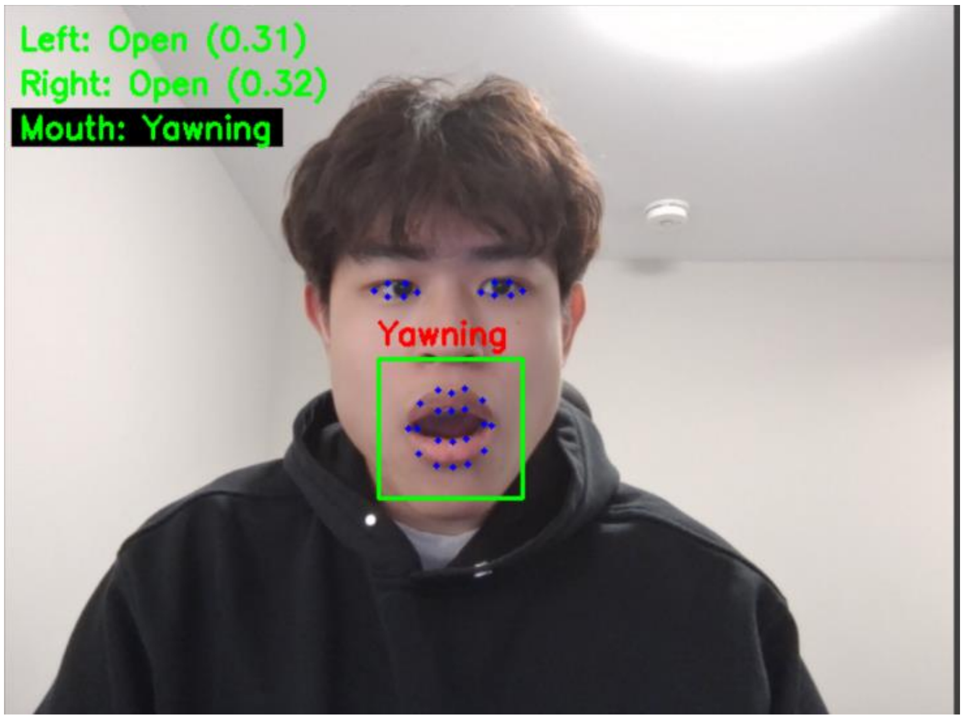


Figure 5.22: Model identifying Yawning

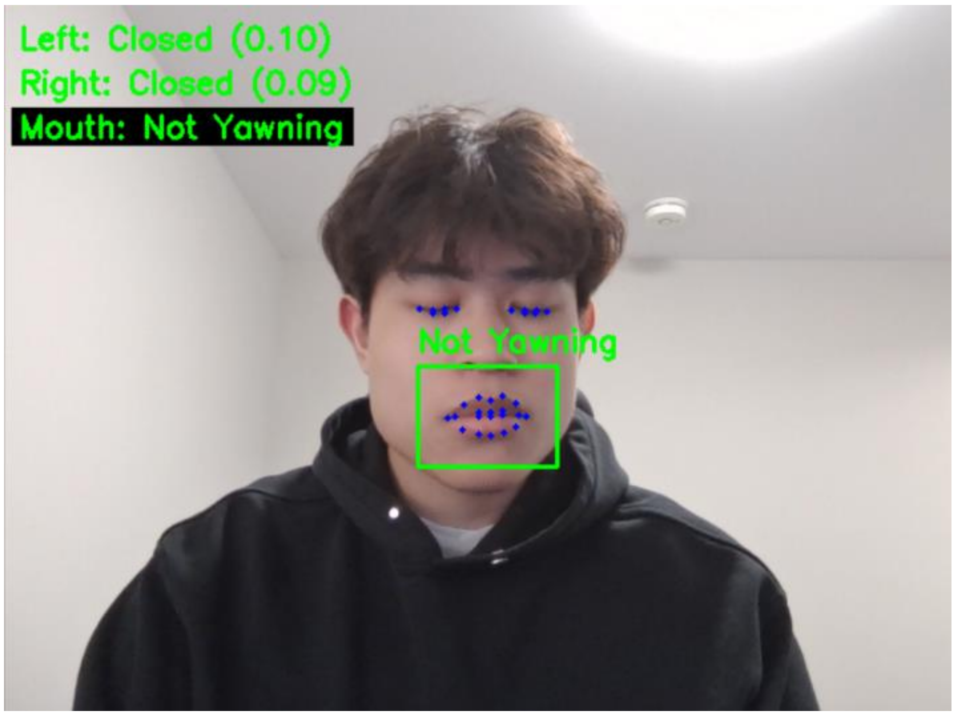


Figure 5.23: Model identifying closed eyes

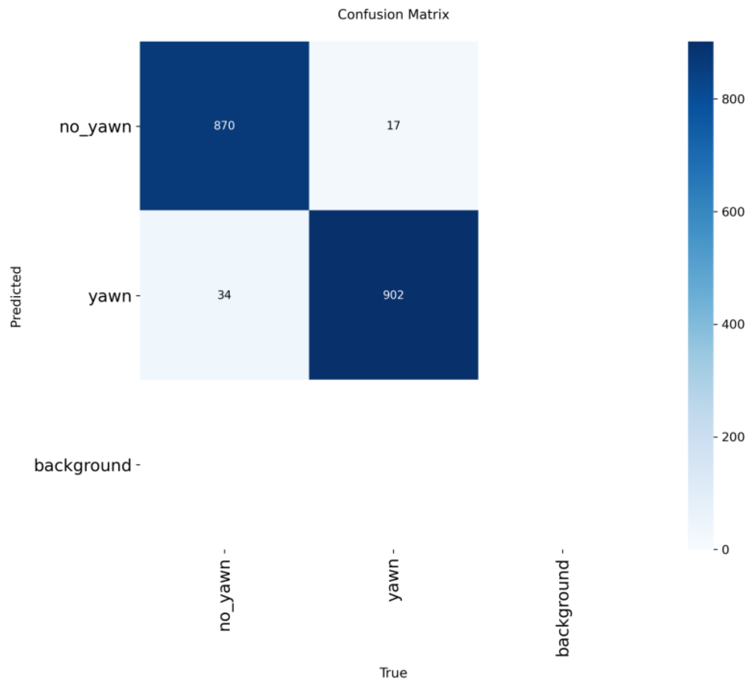


Figure 5.24: Confusion matrix of the trained model

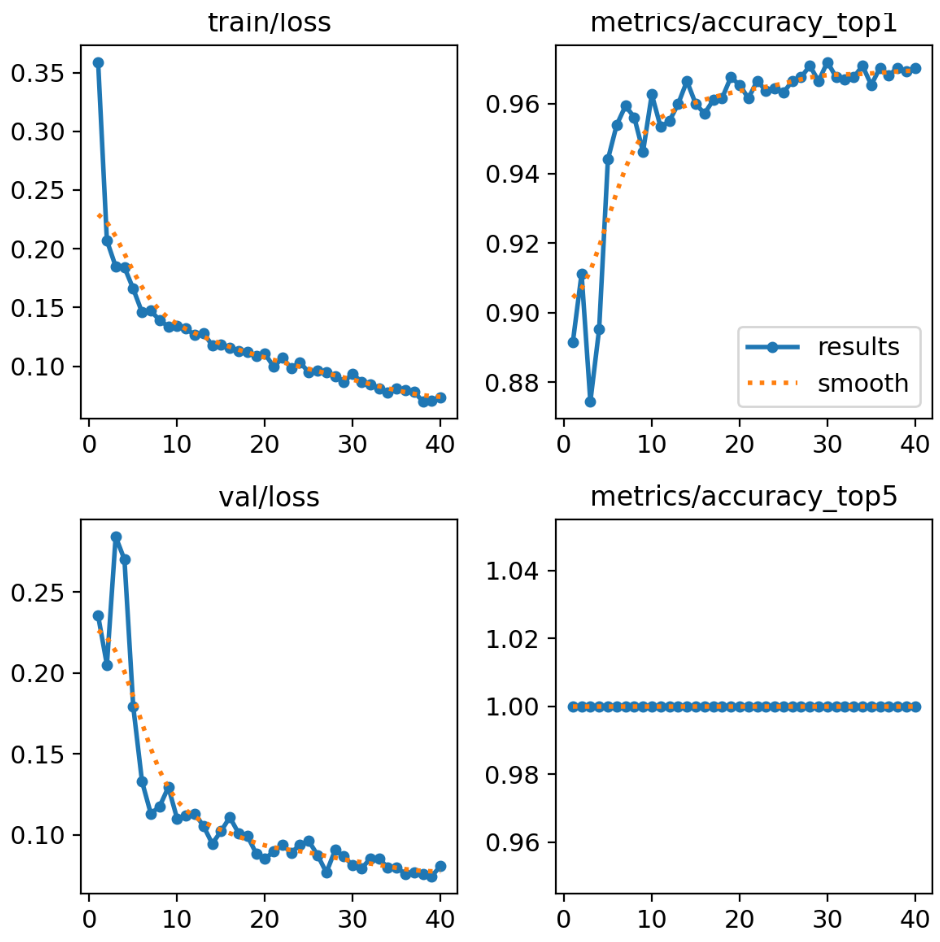


Figure 5.25: Training and validation loss curves

- ***CI Pipeline:***

To ensure code quality, maintainability, and reliability across the project, we implemented a Continuous Integration (CI) pipeline using GitHub Actions. This pipeline automates critical checks every time code is pushed to the repository or a pull request is made, helping us catch issues early and reduce errors in the development process for our repositories.

Included CI Features:

- Branch Naming Enforcement
- Build and Test Automation
- Syntax Validation

- ***Dockerization:***

A Docker container was created to emulate the Jetson Nano environment, enabling remote development and testing of all Jetson-related programs, including the screen UI and C++ driver code. The container replicates the Jetson setup, ensuring that all code changes compile and run correctly before deployment to the hardware. During development, this approach provides every team member with a consistent, error-free foundation, eliminating environment discrepancies. Once the code is verified in the container, integration tests are performed directly on the Jetson to confirm full functionality. This process greatly improves development efficiency by streamlining testing, maintaining consistent builds, and ensuring that only stable, validated code progresses to the integration stage.

- ***Twilio Phone Calls:***

Twilio was used to facilitate phone calls to a driver's emergency contact. A free trial account was created, and functions for calling and messaging that make use of Twilio's API were defined. A back-end server was established to facilitate the connection between the front-end function calls and the back-end connection to Twilio to ensure that sensitive information was not used in the front-end for security and privacy reasons.

- ***Image Capture and Upload Automation:***

To extend the backend capabilities beyond Twilio communication, a Python-based automation script was developed to handle real-time image capture and cloud upload. This module accesses the camera (using OpenCV to interface with the device), captures a frame when a specific event flag is triggered, such as drowsiness detection or emergency activation, and compresses it using the cv2.imencode() function with adjustable JPEG quality parameters to minimize data size. The compressed image is then converted to a byte stream and securely uploaded to Firebase Cloud Storage under an event-specific directory, using the Firebase Admin SDK and timestamp-based file naming. The image's download URL is then stored in the corresponding Firestore document, linking each event log to its visual record.

Hardware Design

Our project is using many different hardware components, with the most important being the SBC (Nvidia Jetson Orin Nano), Camera, and integrated touchscreen. There is a conglomeration of sensors that interface with the SBC and provide data that is to be used on top of the data extracted from the camera system. This entire system works locally/autonomously and connects to the internet to allow remote management

1.25 Hardware Components and Specifications

Component	**Specifications**	**Reasons to select this component**
Nvidia Jetson Orin Nano Super Developer Kit	CPU: 6-core ARM Cortex-A78AE 64-bit, GPU: 1024-core NVIDIA Ampere, Storage: microSD (Expandable), RAM: 8GB LPDDR5, Interfaces: 2x USB 3.2, 1x HDMI, 1x M.2 Key M, 1x M.2 Key E, Camera: 2x MIPI CSI-2.	• Run images through model quickly • Push data to cloud without compromising system performance • Has CUDA acceleration and dedicated GPU cores for our workload
IMX219-83 Stereo Camera	Sensor Type: Dual IMX219 CMOS, Resolution: 8 MP (per lens), Field of View: 75° (per lens), Interface: MIPI CSI-2, Frame Rate: up to 60 fps, Focus: Fixed, Synchronization: Hardware-synced dual cameras,	• Provides accurate depth perception using stereo imaging • Compatible with Jetson's dual MIPI CSI-2 camera inputs

	Power: 3.3V–5V.	• High-quality Sony sensor ensures clear and low-latency image capture for driver monitoring
4.3-inch HDMI LCD 800x480 IPS Capacitive Touch Screen	Resolution: 800 × 480, Display Type: IPS LCD, Touch: Capacitive, Interface: HDMI + USB (for touch), Power: 5V.	• Small screen that isn't intrusive when driving • Touch for easy interaction (dismissing alerts) • Compatible with Jetson over DisplayPort (manufacturer supported)
HD Speakers	Wattage: 5W, Low-level signal.	• Easy to play tunes using any software • Allows for modularity in sound selection, potentially even custom alarms
GY-521 MPU-6050 3-Axis Accelerometer and Gyroscope	Sensor: MPU-6050, Interface: I2C, Axes: 3-Axis Accelerometer + 3-Axis Gyroscope, Voltage: 3.3V-5V.	• To collect vehicle metrics to determine the state of the car and driver • Industry-standard and highly compatible six-axis IMU providing all position vectors needed to find out information about the driver
Thin Film Pressure Sensor Flex Bend Sensor SF15 600 10kg	Type: Thin Film Pressure Sensor, Range: Up to 10kg, Interface: Analog.	• Used to measure pressure applied to steering wheel as a metric for determining drowsiness • Thin and flat, able to fit on steering wheel without impeding driver
Digital ADC Module 16-Bit ADS1115 I2C 4-Channel ADC	Resolution: 16-bit, Channels: 4, Interface: I2C, Voltage: 2V - 5.5V.	• Interface between pressure sensor and Jetson because Jetson doesn't have an onboard ADC • Highly configurable chip
Heart Rate Sensor Module MAX30102	Measures: Heart Rate, SpO2 (Blood Oxygen), Interface: I2C, Voltage: 1.8V - 3.3V.	• Used to measure heart rate of driver as a metric for determining drowsiness

		• Small and compact, allowing easy integration without impeding driver and other components

Table 3: List of hardware components and specifications.

1.26 Hardware Block Diagram

Block diagram showing connections between components and direction of data flow.

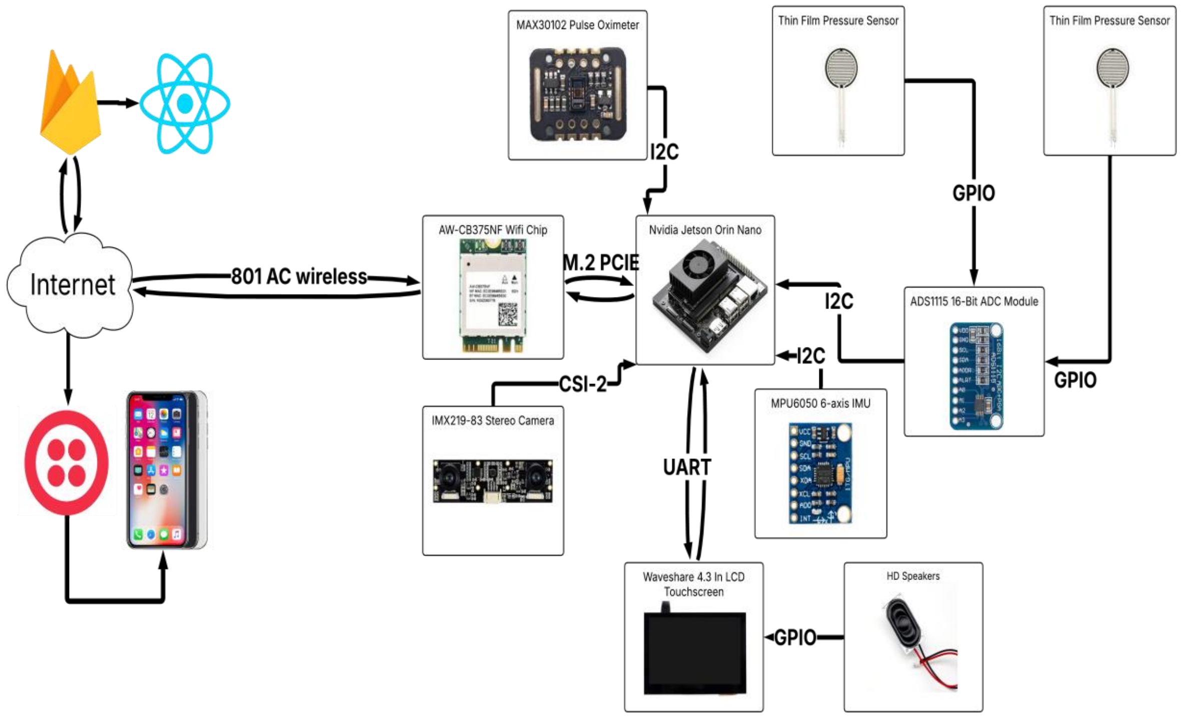


Figure 6.1: Array of components with labelled connections and data flow

1.27 CAD Design and 3D Printed Model

To physically integrate DriveSense into the vehicle, we designed and 3D-printed a simple CAD enclosure whose main purpose is to hold and organize the core components (Jetson, touchscreen, camera wiring, speakers, and other electronics) in one protected assembly. The enclosure was built to keep the system compact and secure while maintaining compatibility between parts and supporting future iterations where the form factor may need to change depending on interior space constraints across different vehicles.

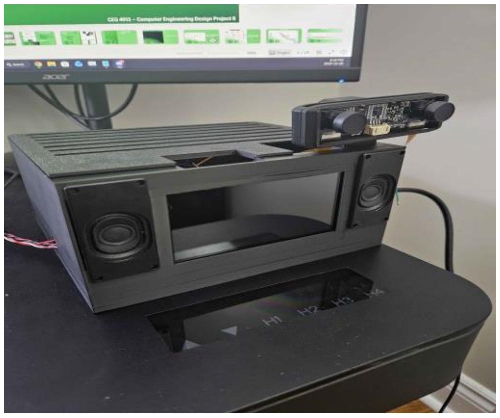


Figure 6.2: Part Enclosure

1.28 Mock Wheel Sensor Integration

A key sensing location is the steering wheel, where our pressure sensor is mounted so it can measure applied hand pressure as an additional heartrate indicator. Because the sensor is thin and flexible, it can sit on the wheel surface without significantly affecting grip comfort, while still capturing meaningful changes (i.e. reduced pressure or inconsistent contact). This aligns with our driving constraint assumptions (hands on the wheel during operation) and allows the steering-wheel sensors to assist the camera-based detection for more accurate drowsiness detection.

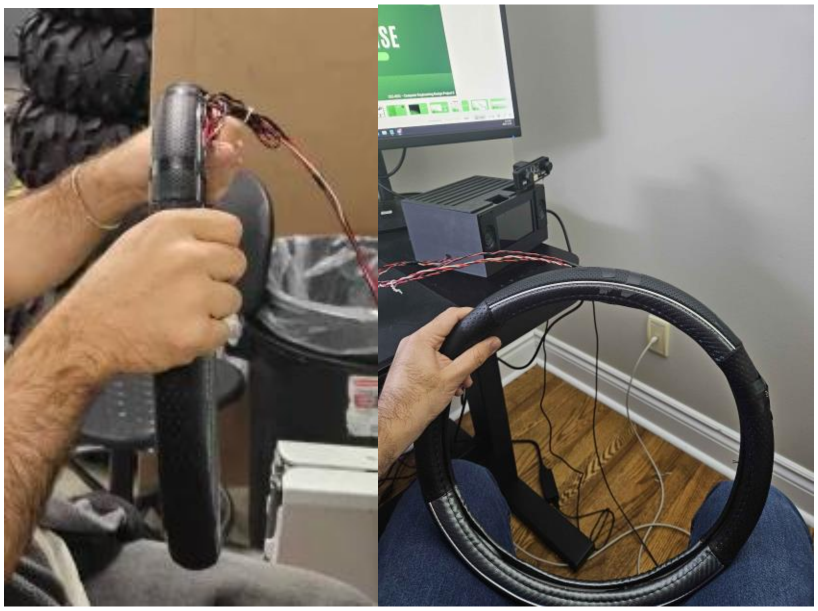


Figure 6.3: Steering Wheel Sensor Integration

Testing

Put here tables of tests to perform such that the project can be tested by people outside your

Project Group. You may list the tests in one column and the results in another column.

Test	Result
Create an account	On the website, user clicks “create an account” button, after which they will be directed to a new page that prompts their name, email, password and phone number. If the email is not already taken, the account will be created. Otherwise, the user is prompted to use a different email address.
Log in	On the website, the user enters their email address and password associated with their account and clicks “log in”. If the information correctly corresponds to an account, they are signed in and can proceed to the next page.
Reset password	On the website, the user clicks “Forgot password,” which takes them to the corresponding page. The user then enters their email address in the input field, after which an email is sent to them with a link to reset their password.
Create an account	On the website, the user clicks “Create an account” which takes them to the create account page. After entering their account details in the input fields, they click “Submit.” A pop-up confirms that the details were saved and that the account was created, and they are taken back to the login page.
Add driver profile	On the website, user clicks “add driver,” which takes them to the new driver page. After entering the driver’s details, the user clicks “Submit,” which creates a new driver with the saved details.
Edit driver profile	On the website, user clicks “edit driver” and is taken to a new page where they can edit the details of the chosen driver.
Delete driver profile	On the website, user clicks “delete driver” and the respective driver profile is deleted.
View driver profile	On the website, the user clicks a driver card, which takes them to a status page of the driver. The page shows the status of the driver, as well as an event log involving the driver.
Contact the team	On the website, the user clicks “Contact Us” in the navbar, which takes them to the contact page. After entering their name, email address, and desired message and clicking “Submit,” the user receives a confirmation email informing them that their message was received.
Edit account	On the website, user clicks “my account,” which takes them to the account page. The user then edits account information and clicks

	“save” to save the changes.
Sign out	On the website, the user clicks “Sign Out” in the navbar. They are taken back to the login page.
Sort driver profile cards	On the website dashboard, the user can change the order in which driver profile cards are displayed by selecting different options from the dropdown menu.
Filter driver profile cards	On the website dashboard, the user can filter out driver cards by entering into the search bar. Only drivers with names relevant to the search input will populate the dashboard.
Warning dismissal	On the touchscreen display, the user clicks a warning message to dismiss it.
Call emergency contact	On the touchscreen display, the user clicks “CALL EMERGENCY,” which initiates a phone call to the driver’s set emergency contact.
Speaker sound	Speaker produces sound.
Camera	Camera records and gives video feedback, identifies human face and eyes.
IMU sensor	Records and displays positional information on all 6 axes (Acceleration and gyroscopic data on X, Y, and Z axes).
Pressure sensor	The pressure of holding the sensor with either your palm or an individual finger has a different reading. Pressing the sensor with a finger results in a far smaller analog-to-digital converted value in comparison to applying pressure with your palm.
AI model	The AI model consists of a yawning detection model and an eye open and close model. The yawning model can successfully identify when someone is yawning. Even if someone is talking or laughing it does not pick up any false positives. The position of the camera affects the results, and the model will be tuned to increase this accuracy.
Firebase Realtime Database	Sensor readings can be uploaded, fetched, updated and removed from the database using REST APIs.

Table 4: Tests and their corresponding results.

1.29 Software

1.29.1 Verification

Comparison of requirements listed at the beginning of the project with the currently implemented requirements.

Software Requirement	Description	Implementation
User authentication	The user inputs username and	Fully implemented.

	password, system verifies.	
Machine Learning Model	The system should detect when drowsiness occurs.	Fully implemented.
Management of user profiles	The user can add, delete, and edit driver profiles.	Fully implemented.
Display interface	The system provides visual warnings when drowsiness is detected. The user can dismiss warnings when they occur.	Fully implemented.
Emergency call	The system calls the emergency contact when the button is pressed.	Fully implemented.
Events log	The system populates a list of events recorded by the sensors in the order that they occurred, along with their timestamps.	Fully implemented.
Account management	The user edits details of their account.	Fully implemented.

Table 5: Software Requirements Verification

1.29.2 Validation

Detailed testing to ensure how well the software components are functioning.

Software Requirement	Description	Test	Result
User authentication	The user inputs username and password, the system verifies.	Try valid and nonvalid username/password combinations.	The system correctly verifies valid username/password combinations.
Machine Learning Model	The system detects objects with 95% accuracy.	Present face in front of camera, while changing facial expressions (blinking, yawning, closing eyes).	Model can accurately identify eyes and mouth on the face. Model can identify when mouth and/or eyes are closed. The model can predict whether someone is yawning or not, and if their eyes are closed or open. The model was measured to be 97% accurate according to sel benchmarks
Management of user profiles	The user can add/edit/delete driver profiles.	Try clicking the buttons for adding, deleting and editing profiles. Try editing information belonging to a specific user.	Profiles are added and deleted when their respective buttons are pressed.
Display interface	The system gives a visual warning when drowsiness	Prompt system to give warning, then remove the prompt.	The warning is shown when prompted, and is dismissed some time after the prompt

	is detected, which is dismissed when drowsiness is no longer detected.		ends.
Emergency call	The system presents a button to call the emergency contact when in emergency mode, and sends a call to the contact when the user presses the button.	Set the system in emergency mode, and press the button when it is presented.	The system presents a button to call the emergency contact in emergency mode, and a call is established with the contact when the button is pressed.
Events log	The system populates a list of events recorded in chronological order.	Stimulate events to be recorded by the system.	Events are added in chronological order with their respective information and video of the event.
Edit account	The user edits details of their account.	Edit account information and save changes.	Account details are modified after saving changes.
Password reset	The user receives a link to reset their password.	Click "Forgot password" and enter an email address, then click "Submit."	An email is sent to the user's email address with a link, which takes them to the reset password page. The password for the user is updated in the database.
Contact organization	The user sends a message to the team.	Click "Contact Us," enter name, email address and message, and click "Submit."	An email is sent to the user confirming that their message was sent.
Sign out	The user signs out of their account.	Click "Sign Out."	The user is taken back to the login page.
Sort driver profile cards	The user sorts the driver profile cards by certain criteria.	Select options from the dropdown menu.	The driver profile cards are arranged in order according to the option selected in the dropdown.
Filter driver profile cards	The user filters out driver profile cards by name.	Type into the search bar.	Only drivers with names related to the search input will populate the dashboard.
Firebase real-time database	The database stores sensor readings, drivers, and event logs.	Upload dummy data, fetch it, edit it, fetch again and then delete and try to fetch again.	The REST APIs for the database can successfully upload, edit and remove different types of data like sensor readings.

Table 6: Software Validation Tests & Results

1.30 Hardware Testing

Making sure all hardware components are functioning as required. How well you met the requirements promised at the beginning of the project.
E.g. you claimed you can detect objects at 30 meters, do you actually? Is the range still 30 meters, or is it 25?

Hardware Component	Test	Result
Stereo Camera	Capture dual 1920*1080 video streams at 60 fps, able to record in low light.	Using OpenCV, camera successfully records frames at specification, resolution lowered to achieve faster processing times when running machine models
IMU sensor	Testing positional data by moving the sensor to different locations and orientations.	Successfully tested reading values using acceleration due to gravity, all 6 axes (Gyroscope and acceleration in x, y, and z planes) are correct
Speaker	Play Sound clips	Successfully played sounds, and we were able to play both high and low pitch tones to ensure that the audio alerts to the driver are alerting.
Pressure sensor and ADC	Apply pressure using a person's hands.	Results indicate that the analog measurement is in the range of the ADC (Analog-to-Digital converter) when the sensor is pressed with a finger versus a person's palm. The measurement is on the lower range of the ADC when pressed with the finger, whereas, when pressed against the palm, the value is on the higher end of the ADC range.
Heart rate sensor	Measure heart rate, pulse, blood oxygen levels, and breathing rate.	Results matched closely with the Apple Watch built-in heartbeat sensor with a small margin of error.

Table 7: Hardware Tests & Results

Project Management

1.31 Protocols and Procedures

1 - Development Protocols

- **Object-Oriented Programming (OOP) Standards:**
  - Driver code must follow OOP principles, including encapsulation, inheritance, and polymorphism, to enhance modularity and maintainability.

    - Each class should serve a distinct function, with clear separation between hardware control, data processing, and logging.
- **Branching and Version Control Protocol:**
    - The team will use Git with a structured branching strategy (e.g., main, develop, feature, bugfix branches).
    - All feature development must occur in separate branches and follow a merge request (MR) template before merging.
    - Code reviews will be mandatory before merging into the develop branch to ensure quality.
- **Conflict Resolution Protocol:**
    - Developers must pull the latest changes before making modifications.
    - In case of conflicts, developers must attempt resolution locally and document the changes.
    - If unresolved, a peer review session or an in-person meeting will be scheduled to resolve the issue collaboratively.
- **Continuous Integration / Continuous Deployment (CI/CD) Protocol:**
    - All commits will trigger automated unit tests and linting checks in a CI/CD pipeline before code integration.
    - Builds must pass all predefined tests before merging to the main branch.
    - Any failed pipeline must be addressed before proceeding with further development.
- **Documentation Protocol:**
    - All code must be documented following docstring standards (e.g., Doxygen for C++ or Sphinx for Python).
    - Each module should include a README file detailing its purpose, dependencies, and usage.
    - Major design decisions should be logged in the project documentation repository for future reference.

2 - Development Procedures

1. Driver Code Development Procedure
    a. Who: All developers
    b. What: Implementing and testing driver code while adhering to OOP principles
    c. Where: Development will take place in the repository (feature branches) and tested on local machines or Raspberry Pi setups
    d. When: During the development phase before integration with the main system
    e. Why: To ensure a structured, maintainable, and modular codebase

2. Code Review and Merge Procedure
    a. Who: Developers and reviewers (designated team members)
    b. What: Conducting peer reviews before merging code into develop
    c. Where: Code reviews will occur through GitHub/GitLab pull requests
    d. When: Before any major merge into develop or main
    e. Why: To maintain code quality, prevent bugs, and ensure adherence to OOP and security standards

3. CI/CD Enforcement Procedure
    a. Who: All developers
    b. What: Ensuring the CI/CD pipeline successfully validates each commit
    c. Where: GitHub Actions/GitLab CI/CD or an equivalent automation tool
    d. When: Every time a new feature branch is pushed or a pull request is created

   e. Why: To prevent integration issues and ensure a stable development environment

4. Conflict Resolution Procedure
   a. Who: Developers involved in the conflicting code
   b. What: Identifying, resolving, and documenting code conflicts
   c. Where: Resolved locally, with discussions in team meetings if necessary
   d. When: When a merge conflict occurs
   e. Why: To ensure a smooth integration process and avoid loss of work

5. In-Person Meeting Procedure
   a. Who: All team members
   b. What: Discussing project progress, blockers, and major design decisions
   c. Where: STEM building (makerspace) or designated team workspace
   d. When: Weekly or as needed when major issues arise
   e. Why: To ensure alignment among team members and resolve issues efficiently

6. Hardware Damage Reporting Procedure
   a. Who: Any team member who identifies an issue
   b. What: Reporting, diagnosing, and replacing damaged hardware components
   c. Where: Issues logged in a shared hardware status document, replacement handled in the makerspace
   d. When: Immediately upon detecting hardware malfunction
   e. Why: To prevent delays in development and testing due to faulty components

1.32 Tasks and Timelines

Date	Milestone	Deliverable
1/17/25		Project proposal
1/22/25	EEF form submitted	
1/28/25	Requirements and constraints identified	
1/29/25	Software UI Mockups created	
1/29/25	Parts picked and ordered	
2/5/25	Hardware architecture designed	
2/7/25		SCRUM and Sprint checkpoint Alpha
2/9/25	Login page created, add/delete drivers function implemented	
2/9/25	Sensors and camera initial implementation complete	
2/10/25	Screen and Nano CAD design files made and printed	
2/11/25		Mid-term presentation and demonstration – Term 1
2/24/2025		Mid-term progress report – Term 1
3/24/2025	ML model made functional	
3/30/2025	All website pages implemented	
3/30/2025	Firebase Realtime database implemented	
4/1/2025		Final presentation and demonstration – Term 1

4/14/2025		Final Progress Report – Term 1
9/03/2025		First Class presentations
9/14/2025	EEF form submitted (Term 2)	
9/29/2025	Database and Docker container testing environment created	
10/07/2025	Display UI redesigned and backend integrated	
10/09/2025		Midterm presentation and demonstration – Term 2
10/20/2025		Midterm Progress Report – Term 2
11/9/2025	Twilio API integration	
11/17/2025	Screen UI database integration	
11/25/2025	CAD complete	
11/26/2025	AI model completed	
11/27/2025	Website database integration	
11/27/2025	Final project integration	
11/28/2025		Final presentation and demonstration – Term 2
12/12/2025		Final Progress Report – Term 2

Table 8: Dates and Milestones

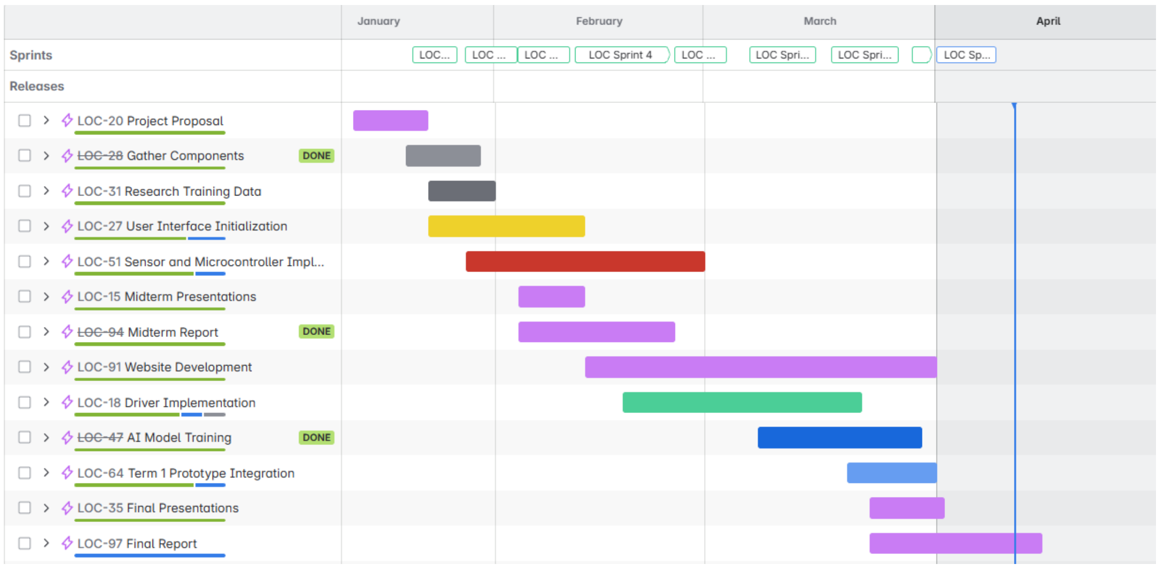


Figure 8.1: Gantt chart for term 1.

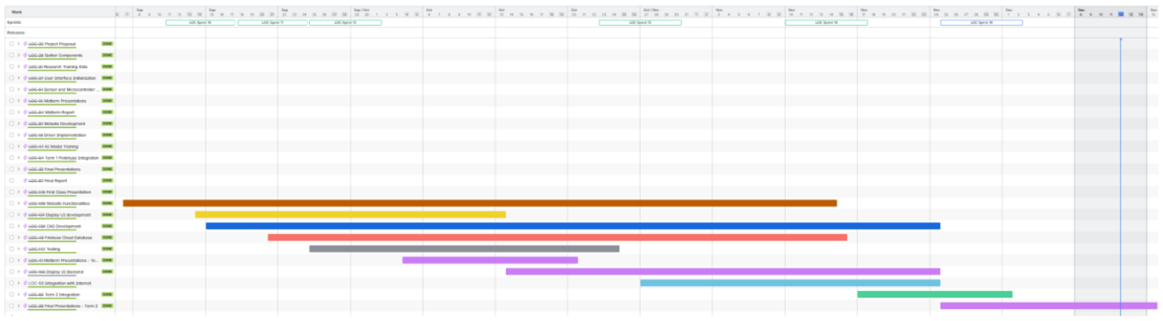


Figure 8.2: Gantt chart for term 2.

1.33 Work Force

Completed Tasks	**Members**
Project Proposal	All members
Agile management (Jira)	Keith, Abdullah, Saurav
Work Breakdown Structure	Saurav
Gantt Chart	Keith
AI model and dataset research	Aaditya, Kevin, Saurav, Keith
Selecting hardware components	Aaditya, Keith
Ordering hardware components	Aaditya
EEF form	Aaditya, Keith
Website development	Kevin, Keith
Display UI development	Hajer, Keith
Hardware CAD	Abdullah
Hardware programming	Aaditya
Hardware architecture	Aaditya
Software UML design	Abdullah
CI/CD pipeline	Abdullah
Progress Report	All members
Presentation slideshow	All members
Hardware assembly	Aaditya, Abdullah
Database development	Saurav, Kevin

Table 9: Completed tasks, along with their respective members.

1.34 Work Breakdown Structure

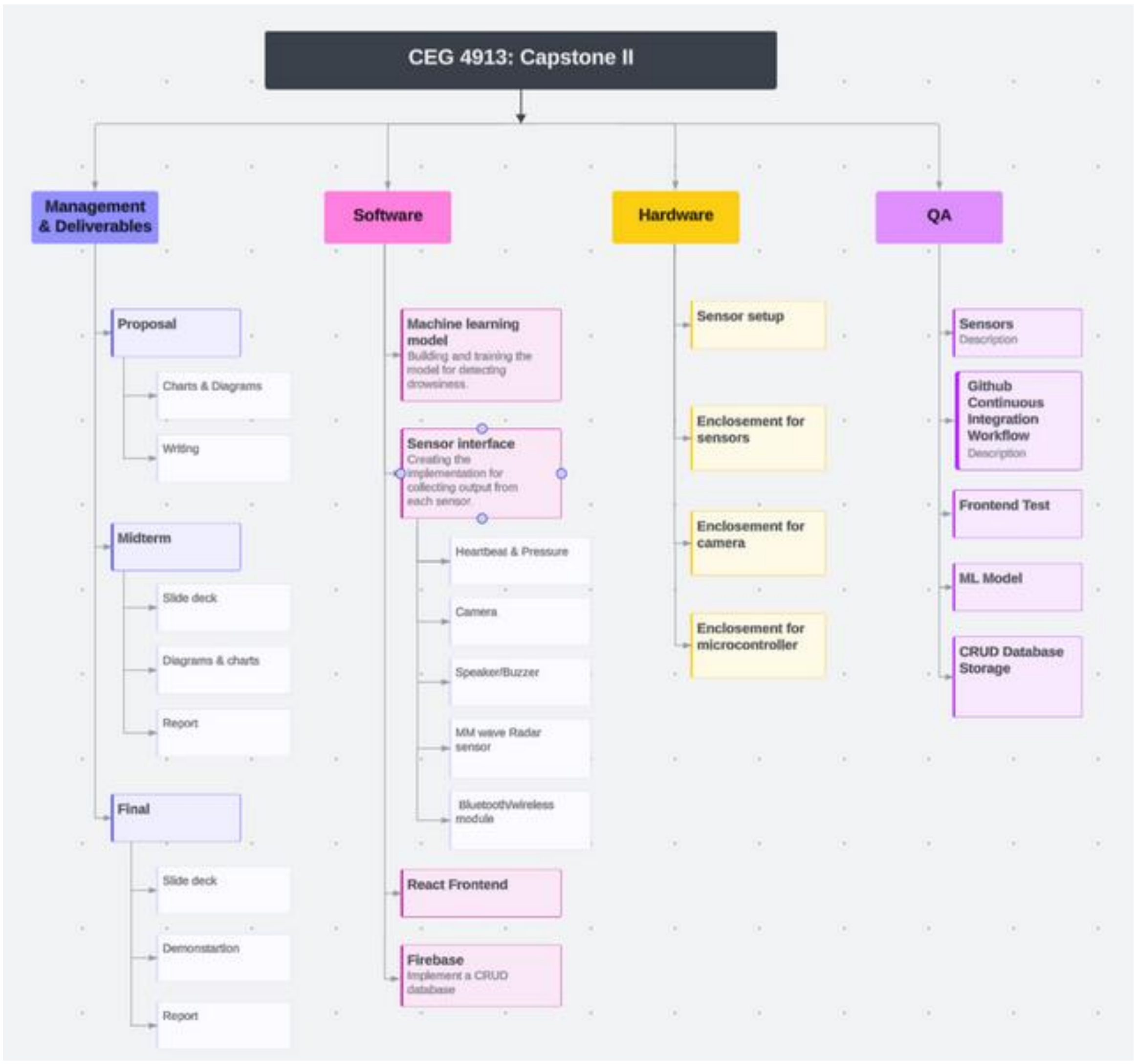
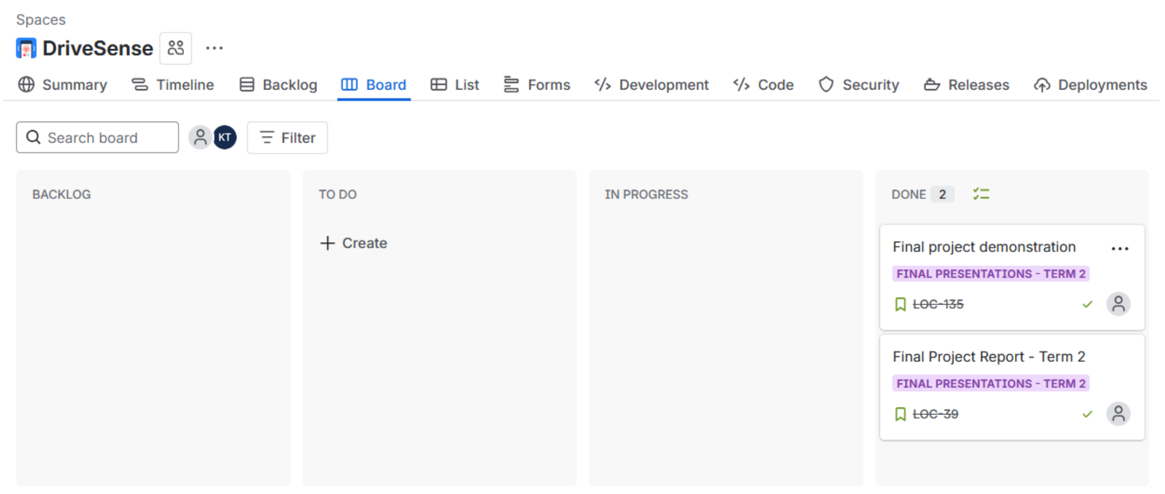


Figure 8.3: Current Work Breakdown Structure

1.35 Jira Board



Final project demonstration
FINAL PRESENTATIONS - TERM 2
~~LOC-135~~

Final Project Report - Term 2
FINAL PRESENTATIONS - TERM 2
~~LOC-39~~

Figure 8.4: Jira Board for Sprint 16

1.36 Risk Management Plan

Risk	Prob	Impact	Type	Mitigation/Solution	Priority	Responsibility
System overvolts	Medium	High	Internal	Always keep the stress on the system to a minimum. Ensure that cooling fans Implement a temperature cutoff point in which an emergency shut off is triggered.	High	Aaditya
Absence of a team member	Medium	Low	Internal	Ensure that work is equally divided among members, temporarily redistribute responsibilities temporarily	Medium	Team leader
Jetson Nano stops working	Low	High	Internal	Keep a backup, order a new one	High	Aaditya
Merge conflict on Git	Medium	Low	Internal	Use the in-built git extension in Visual Studio Code to help resolve merge conflicts. Also, ensure to pull changes from GitHub prior to pushing.	Low	Software team
Jira stops working	Low	Low	External	Use alternative applications, use other methods for management (e.g. calendar reminders)	Low	Keith

Jetson OS issues from installing packages	Medium	High	Internal	Use a virtual environment space to install and manage all packages	Medium	Aaditya
Twilio stops providing API SMS messaging service	Low	High	External	Use another service that offers SMS messaging with an API	High	Keith
Sensors or power supply shorts out	Low	High	Internal	Run a voltmeter through components and identify failures. Ensure components are properly grounded.	High	Aaditya
The Jetson cannot save data locally prior to pushing to database.	High	High	Internal	Use an external memory device to temporarily save data, and when resources are available save to cloud.	High	Saurav
Image quality degrades through processing	Medium	High	Internal	Minimize compression and test rescaling and resizing images to be processed by our model. Standardize image resolutions being used to send to the model.	Low	Saurav
Camera model face detection issues in the dark	Medium	High	Internal	Find different models for the face detection library and test them	Medium	Abdullah
Delays in emergency alert notifications	Medium	High	Internal	Optimize network requests and implement priority queuing for emergency alerts.	High	Hajer
Inaccurate detection due to poor sensor calibration	Medium	High	Internal	Continuously refine the detection model, apply real-world calibration tests, and adjust threshold values based on feedback.	High	Hardware Team
Data loss due to unexpected shutdown	Low	High	Internal	Implement periodic auto-saving of data to Firebase or local storage and introduce failover mechanisms.	Medium	Backend Team

Inclement weather on day of meeting or presentation	Low	Medium	External	Reschedule event, communicate remotely online	Low	Team Leader

Table 10: Table of potential risks, along with their probabilities, level of impact, type, solution, priority, and responsible member.

Future Works & Closing Remarks

The DriveSense^{TM} project emphasizes reliability and safety, warning drivers of potential drowsiness and providing real-time remote monitoring by a third party. However, many adjustments and additions can be implemented to further increase user safety and ease of access. The following are some possible improvements to the base project that can be implemented in the future.

- **Enhanced power dissipation:** a decrease in power dissipation can increase the efficiency of the device, as well as improving system reliability by decreasing the chance of overheating or power failure
- **Voice activation and control**: device control via voice commands will improve accessibility. It also increases the safety of the system, as a lone driver need not divert their attention to interact with the system
- **Web monitoring on other platforms**: porting the web application to other platforms such as mobile will increase system accessibility, allowing users to monitor the device outside the home via their mobile phone
- **OBD port compatibility**: allowing the system to connect to the OBD port will increase its adaptability, allowing it to be fitted into a greater range of vehicles.

APPENDIX A: PROJECT CODE

The source code for the website can be accessed here.

The source code for the display UI can be accessed here.

The source code for the hardware components can be accessed here.

The source code for the ML model can be accessed here.

APPENDIX B: PROJECT DIARIES