



Driver Drowsiness Detection Using Onboard Camera

Final Year Project by:

Shaheer Ahmed

Noor ul Huda Ajmal

In Partial Fulfilment of the Requirements
for the Degree Bachelors in Software Engineering
(BESE)

School of Electrical Engineering and Computer Science,
National University of Sciences and Technology,
Islamabad, Pakistan

(2023)

DECLARATION

We hereby declare that this project report entitled "Driver Drowsiness Detection Using Onboard Camera" submitted to the "Department of Software Engineering", is a record of original work done under the guidance of our Supervisor "Dr. Hasan Ali Khattak" and Co advisors "Mr. Maajid Maqbool" and "Dr. Bilal Ali". No part of the report has been plagiarized without citations. Also, this project work is submitted in the partial fulfilment of the requirements for the degree of Bachelor of Software Engineering.

Team Members

Shaheer Ahmed

Noor ul Huda Ajmal

Signatures:

Advisor:

Dr Hasan Ali Khattak

Co Advisors:

Mr. Maajid Majbool

Dr. Bilal Ali

Signatures:

Table of Contents

1. INTRODUCTION
 - 1.1. Problem
 - 1.2. Proposed Solution: Driver Drowsiness Detection
 - 1.3. Core Functionalities
 - 1.4. Summary
2. LITERATURE REVIEW
 - 2.1. Accidents Due to Drowsiness
 - 2.2. Existing Solutions
 - 2.3. Why Driver Drowsiness Detection System
3. PROBLEM DEFINITION
 - 3.1. System aim
 - 3.2. Audience Of The Project
 - 3.3. Impact
4. SYSTEM FEATURES
 - 4.1. Functional Requirements
 - 4.2. Use cases
 - 4.3. Tools and Technologies
5. DETAILED DESIGN AND ARCHITECTURE
 - 5.1. SYSTEM ARCHITECTURE
 - 5.1.1. Architecture Design Approach
 - 5.1.2. Architecture Design & Subsystem Architecture
 - 5.1.3. Dataflow of the system
 - 5.2. DETAILED SYSTEM DESIGN
 - 5.2.1. Drowsiness Detection
 - 5.2.2. Accident Detection
 - 5.2.3. Website
6. IMPLEMENTATION AND TESTING

6.1. Workflow

6.1.1. Research and overview

6.1.2. Compiling Dataset

6.1.3. Training Model

6.1.4. Testing Model

6.1.5. Setting up system (edge device)

6.1.6. Developing the website

6.2. Testing

6.2.1. Unit Testing

6.2.2. System Testing

7. RESULTS AND DISCUSSION

8. CONCLUSION AND FUTURE WORK

8.1. Conclusion

8.2. Impact on Society

8.3. Future Work

9. REFERENCES

Table of Figures

Figure 1 Drowsiness Detection and Alert Driver Use case.....	12
Figure 2 Accident Detection Use case.....	12
Figure 3 Website Use case.....	13
Figure 4 IR Camera.....	15
Figure 5 MPU6050 Accelerometer.....	16
Figure 6 Raspberry Pi 4 b.....	17
Figure 7 High level architecture Block diagram.....	20
Figure 8 UML Diagram for Website architecture.....	21
Figure 9 Edge Device Dataflow.....	23
Figure 10 Website Dataflow.....	24
Figure 11 System Activity Diagram.....	25
Figure 12 Dlib Facial Landmarks.....	26
Figure 13 CNN model architechture.....	27
Figure 14 Drowsy Driver Event.....	28
Figure 15 Accident Detection Event.....	29
Figure 16 Home Page.....	30
Figure 17 Services Page.....	31
Figure 18 Application Page.....	32
Figure 19 Subscription Page.....	33
Figure 20 Login View.....	33
Figure 21 Admin Panel.....	34
Figure 22 User Panel.....	34
Figure 23 Models View.....	35
Figure 24 Complaints View.....	35

INTRODUCTION

Driving is an indispensable part of commuting. Many people rely on it as a means to perform daily tasks, for some it is even a source of living. However, driving for an extended period of time or driving at odd hours of the day can cause the driver to be inattentive to the road. This is especially dangerous since it not only puts the passengers at risk, but other commuters and pedestrians as well.

It is estimated that about 2.5 million car crashes involve distracted drivers globally every year. ^[1] Many of these accidents are simply due to driver inattentiveness and as such can be curtailed by simply monitoring negligent behaviour.

1.1 PROBLEM STATEMENT

Driver Drowsiness systems are still a niche product despite being a considerably mature technology. Although some proprietary and aftermarket systems exist, they are either out of reach or simply too unreliable to be used.

With more people taking to the roads every year, there is unprecedented market potential for a product that can deliver the utility for the said market segment. The target demographic would be anyone eligible to drive a vehicle, irrespective of their gender and age group (18 years or older).

1.2 Proposed Solution

The driver drowsiness detection system will be a reliable, cost effective aftermarket system that can keep track of driver's attentiveness to help prevent accidents due to driver inattentiveness, keeping the other commuters and pedestrians safe from easily avoidable accidents.

The system will consist of an edge device along with a camera. A neural network model utilizing computer vision would be deployed on the edge device that would keep check of the driver's behaviour, making sure that the driver is attentive to the road. It will use a buzzer to either the vehicle's stereo system to alert the driver.

A novel feature of this system would be a subscription based service. The users would be provided with monthly system updates designed to improve the safety and performance of the system. The users would use the companion website to download system updates.

Furthermore, in an event of an accident, the user's footage can be recorded by the system. The users can choose to share this footage with the developers to improve device safety. Hence the developers can implement Continuous Quality Improvement (CQI), a feat that has not been achieved in this product segment.

1.3 Core Functionality

The main functions of this project include:

Drowsiness detection

The system will detect and capture the drowsiness behaviour of the driver while driving, following the image input from the camera. It will be accompanied with getting model predictions in real time. This feature is of high priority to the system.

Alerting Driver

The system shall alert the driver upon detecting drowsiness by the application of buzzer. This feature is of high priority to the system.

Accident Detection

The system shall use accelerometers attached to the board to detect an accident. The system shall capture the last few seconds of the driver's footage in case of an accident. This footage can then be used by the developers to improve the Neural Network of the system in case the accident was caused due to product error.

1.4 Summary

Driving can lead to inattentiveness, putting passengers, other commuters, and pedestrians at risk. Current driver drowsiness detection systems are unreliable or inaccessible. This project will utilize computer vision and neural networks to detect driver behaviour and alert them if they become

inattentive. The system will offer monthly updates and allow for footage recording in the event of an accident, which can be used to improve system safety through Continuous Quality Improvement (CQI).

LITERATURE REVIEW

2.1 Accidents Due to Drowsiness

Accidents caused by drowsiness are a significant global issue that can have severe consequences. According to a report by the World Health Organization (WHO), drowsiness is responsible for up to 20% of road accidents worldwide ^[2]. In the United States alone, drowsy driving causes an estimated 90,000 crashes, 50,000 injuries, and 800 deaths annually ^[3].

Despite the alarming statistics, many vehicles do not come equipped with driver drowsiness detection systems. These systems are designed to monitor a driver's level of alertness and provide warnings when signs of drowsiness are detected. However, according to a study conducted by the European Transport Safety Council (ETSC), only 15% of new cars sold in Europe in 2020 came equipped with a driver drowsiness detection system ^[4]. This, coupled with the lack of any aftermarket systems makes this an important issue.

2.2 Existing Solutions

While some manufacturers provide similar systems on their luxurious offerings, these vehicles are still out of reach for most of the population. The aftermarket offerings are not as reliable as they claim to be.

Examples of systems from a few manufacturers

BMW: Active Driving Assistant with Attention Assistant analyses driving behaviour and, if necessary, advises the driver to rest ^[5].

Ford: Driver Alert, introduced with 2011 Ford Focus. The Driver Alert system comprises a small forward-facing camera connected to an on-board computer. The camera is mounted on the back of the rear-view mirror and is trained to identify lane markings on both sides of the vehicle ^[6].

Examples of aftermarket products

The Speedir state-of-the-art driver monitoring system (DMS) can detect drowsy drivers by accurately measuring eye and head position, driver attention and fatigue ^[7]. It was a promising crowdfunded project that had managed to achieve 67% of its funding goal, however a high retailing price of \$249 failed to capture the wider audience.

2.3 Why Driver Drowsiness Detection System

A casual glance at the statistics and reports highlight that the lack of reliable aftermarket drowsiness systems is still an unresolved issue. With the rise of and strides made in Artificial Intelligence, such a system is more feasible now more than ever, thanks in part to the wide adoption of smart Internet of Things (IoT) devices, hence lowering component costs.

PROBLEM DEFINITION

3.1 System aim

The lack of aftermarket drowsiness detection systems is a significant concern for drivers worldwide. These systems are designed to monitor a driver's level of alertness and provide warnings when signs of drowsiness are detected. However, not all vehicles come equipped with this feature, leaving drivers vulnerable to the dangers of drowsy driving.

This project aims to provide an additional layer of safety by alerting the driver when signs of fatigue are detected. This can help prevent accidents caused by drowsiness, which can be fatal in some cases.

Additionally, this detection system can be installed in any vehicle, regardless of the make or model. This means that drivers do not have to purchase a new car to access this important safety feature. It can also be a cost-effective solution compared to buying a new car with built-in driver drowsiness detection systems.

3.2 Audience Of The Project

Drivers

Drivers who frequently travel long distances, work night shifts, or have sleep disorders are more likely to experience drowsiness while driving. This system will benefit these drivers the most as it can provide an additional layer of safety by alerting them when signs of fatigue are detected.

Passengers and Pedestrians

Passengers in the vehicle and pedestrians will also benefit from the implementation of the drowsiness detection system. In case the driver becomes drowsy or loses control of the vehicle due to fatigue, the system can help prevent accidents and save lives.

3.3 Impact

The implementation of the drowsiness detection system can have a positive impact on society as a whole. Road accidents caused by drowsy driving can result in significant economic costs in terms of medical expenses, property

damage, and lost productivity. By preventing such accidents, the introduction of these systems can help save lives, reduce economic costs, and improve overall road safety.

SYSTEM FEATURES

4.1 Functional Requirements

The functional requirements for the embedded system are listed as:

Drowsiness Detection

- The system should detect the face of the driver.
- The system should trigger an alarm upon detecting a drowsy driver.
- The buzzer alarm should not turn off until the driver isn't drowsy.

Alert Driver

- The system should alert the driver upon detecting drowsiness, through a buzzer alarm.
- The buzzer should not be turned off until the driver is not drowsy.

Accident Detection

- The accelerometer detects deceleration due to accident.
- The last 5 frames of driver footage is saved.

The functional requirements for the web portal will be described as:

Updates Portal:

- The website should allow the user to download the latest version of the updated models

Complaints Portal:

- The user should be able to upload accident footage recorded by the embedded system
- The user should be able to add comments to add context to the footage and the accident

4.2 Use Cases

The embedded system features two main use cases as illustrated below:

Drowsiness Detection and Alert Driver:

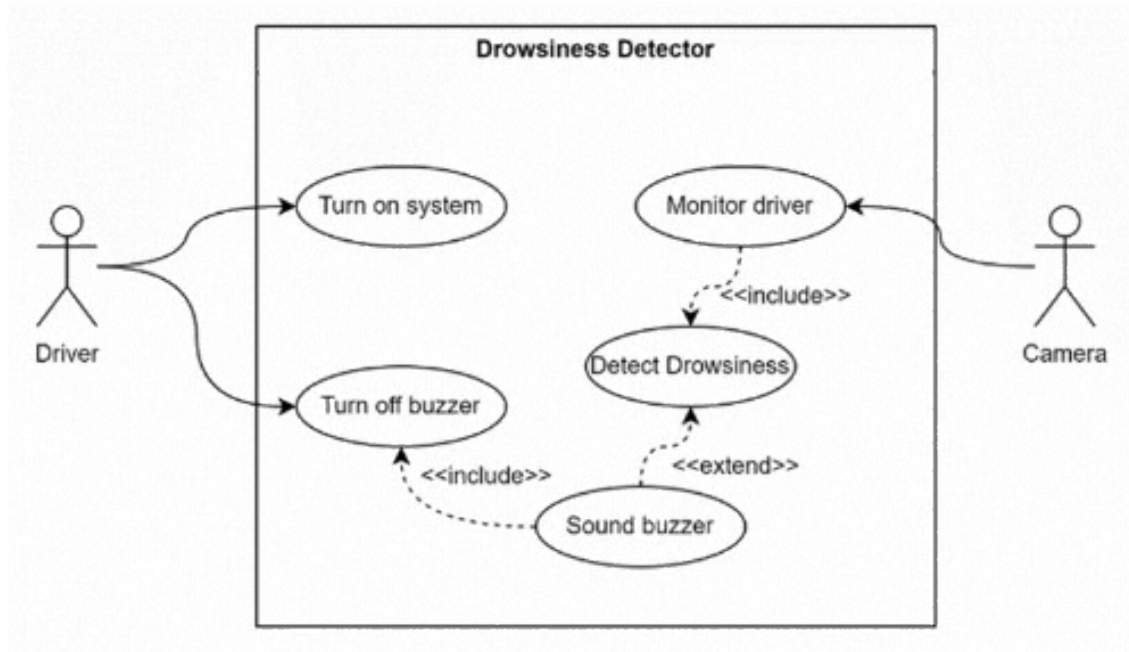


Figure 1 Drowsiness Detection and Alert Driver Use case

Accident Detection:

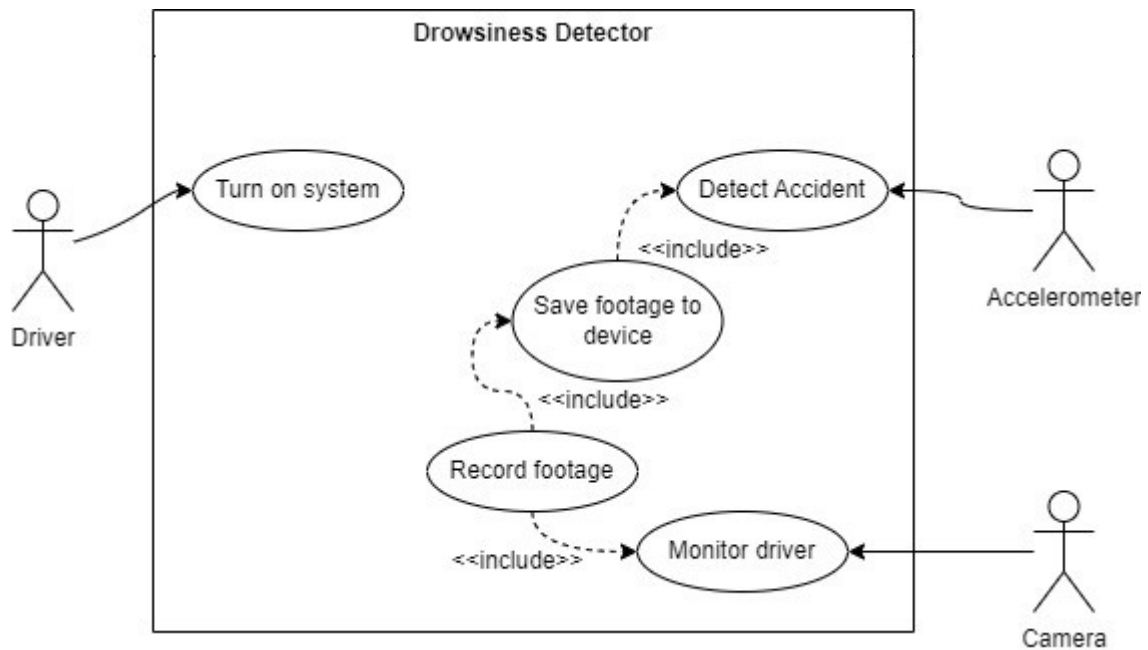


Figure 2 Accident Detection Use case

The use case for the website is illustrated as follows:

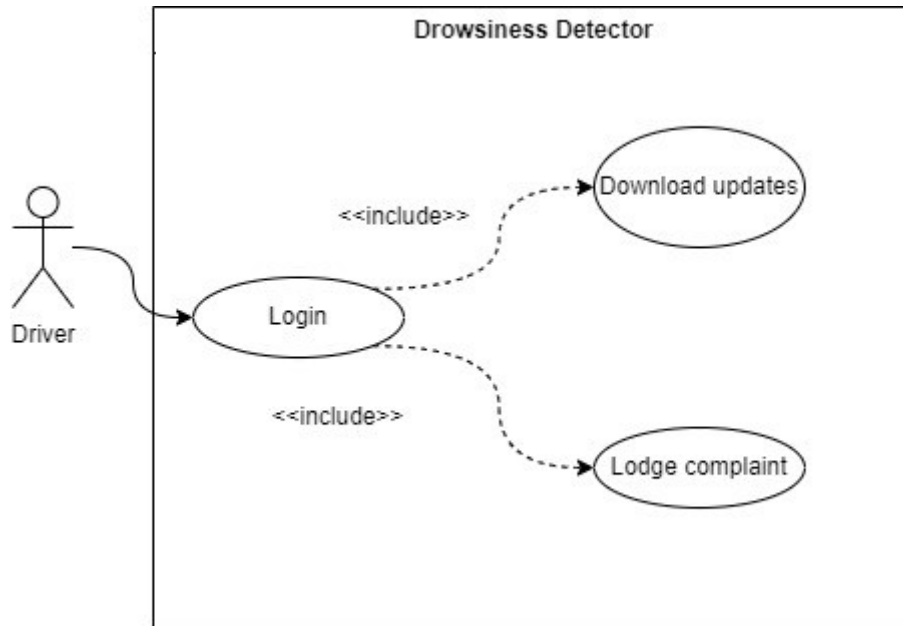


Figure 3 Website Use case

4.3 Tools and Technologies

This section delves into the tools and technologies that will be used to construct the project. It covers both the hardware and software as well as any languages, libraries or frameworks that will be used to develop the modules of the project.

Python

Python is a high level programming language that has quite mature libraries for machine learning and data science.

Tensorflow

TensorFlow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is used for building and training machine learning models, particularly deep neural networks.

TensorFlow also provides several high-level APIs, such as Keras and Estimator, that simplify the process of building and training deep learning models. The library also includes tools for visualizing and debugging

models, distributed computing, and deploying models to different platforms, such as mobile devices or the cloud.

We used tensorflow due to the simplified API as well as its compatibility with libraries specifically designed for Machine learning on low end devices such as TensorFlow Lite.

TensorFlow Lite

TensorFlow Lite is a lightweight version of the popular open-source machine learning framework TensorFlow. It is designed specifically for mobile and embedded devices, making it possible to run machine learning models on devices with limited resources, such as smartphones, IoT devices, and microcontrollers.

TensorFlow Lite allows developers to deploy machine learning models on mobile and embedded devices without the need for a server or internet connection. It provides a set of tools for optimizing and compressing machine learning models, making them smaller and more efficient for deployment on resource-constrained devices.

In addition to model deployment, TensorFlow Lite also provides tools for model training and conversion. This library was vital in order to optimize the convolutional neural networks for the edge device, since we began development of the models before the procurement of the edge device.

OpenCV

OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and machine learning software library.

OpenCV provides a wide range of algorithms for image and video processing, such as object detection, face recognition, feature detection, image filtering, and more. It also includes tools for camera calibration, stereo vision, optical flow, and machine learning. The library supports various image and video file formats and can read and write to different types of cameras and video streams.

We mainly used OpenCV to extract frames, resize them and to convert them to grayscale to reduce the intensive need for compute resources.

Dlib

The Dlib Python API provides a simple and intuitive interface to access powerful tools and algorithms for image processing, including facial detection and landmark detection.

shape_predictor_68_face_landmarks.dat is a pre-trained model file for facial landmark detection in Dlib. It contains a machine learning model that has been trained to identify 68 specific points on a face, including the corners of the eyes, nose, mouth, and chin.

Dlib uses the shape_predictor_68_face_landmarks.dat file in its facial detection and recognition algorithms. By detecting and locating these specific landmarks, Dlib can accurately identify and track faces in an image or video. This information can then be used for a wide range of applications, including facial recognition, emotion detection, and gaze tracking.

Infrared Camera

It features a 5-megapixel sensor and can capture still images and video at a resolution of 2592 x 1944 pixels.

What sets this camera module apart from others is its infrared night vision capability. It has a built-in infrared (IR) LED that illuminates the surrounding area with invisible IR light, enabling the camera to capture images in complete darkness. The IR LED is controllable, allowing users to adjust the brightness and turn it on or off as needed.



Figure 4 IR Camera

picamera

picamera is a Python library that provides a way to capture images and video using the Raspberry Pi camera module.

It allows you to set various camera parameters, such as resolution, frame rate, and exposure, and provides a way to capture images and video to files or streams. The library also supports various image and video formats, including JPEG, PNG, and H.264.

MPU6050 Accelerometer

The MPU6050 is a commonly used accelerometer and gyroscope sensor module. It is designed to measure acceleration, rotation, and tilt of a device or object. The module consists of a MEMS (Micro-Electro-Mechanical Systems) accelerometer and a MEMS gyroscope, along with a digital motion processor (DMP) that performs complex calculations on the sensor data to provide accurate readings of orientation, acceleration, and other motion-related parameters.

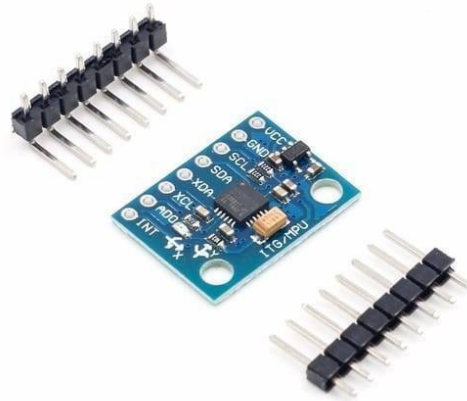


Figure 5 MPU6050 Accelerometer

For our project, we will only concern ourselves with the accelerometer. The accelerometer measures linear acceleration along three axes (X, Y, and Z) and provides acceleration data in units of meters per second squared (m/s^2).

The sensor was chosen as it communicates with a microcontroller or other devices through an I2C interface, as our chosen edge device does not support communication through analogue means. I2C allows the sensor to communicate with other devices using digital signals rather than analogue signals.

The second reason was the mpu6050 raspberry pi python library.

The MPU6050 Raspberry Pi Python library is a Python library that allows developers to interact with the MPU6050 accelerometer and gyroscope sensor using the Raspberry Pi. The library provides a simple and easy-to-use interface for reading data from the MPU6050 sensor and processing it within a Python program.

The MPU6050 Raspberry Pi Python library can be installed using the pip package manager, which makes it easy to integrate into existing Python projects. Once installed, the user can access the MPU6050 sensor by importing the library and using its provided functions and classes.

These features made the integration and calibration of the accelerometer much easier.

Raspberry Pi 4

The Raspberry Pi 4 is a powerful and versatile single-board computer that offers a lot of features and capabilities in a small form factor and at an affordable price point. It is powered by a Broadcom BCM2711 quad-core Cortex-A72 64-bit processor clocked at 1.5 GHz and features ample connectivity in the form of dual-band 802.11ac wireless networking, Bluetooth 5.0, Gigabit Ethernet.

These factors made it a very good candidate for the development and integration of the Driver Drowsiness Detection system.

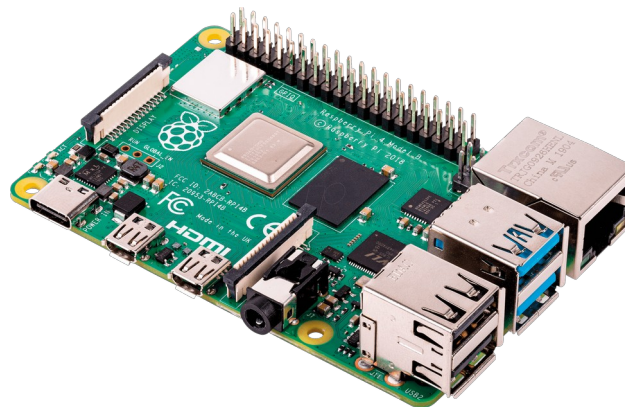


Figure 6 Raspberry Pi 4 b

For web development, we used the following tools:

Django (for backend):

Django is a high-level Python web framework that is used for rapid development of secure and maintainable websites and web applications. It was first released in 2005 and is open-source software. It follows the Model-View-Controller (MVC) architectural pattern and focuses on the reusability and "pluggability" of components, which allows for faster and more efficient development of web applications. This was the main reason we chose this library for the backend.

React (for frontend):

React is a JavaScript library that is used for building user interfaces. It was developed by Facebook and first released in 2013. React allows developers to build reusable UI components that can be used to create complex and

interactive web applications. We chose to use react for the frontend due to its simplicity, efficiency, and the vast community support and resources available.

For testing, we used the pytest library.

Pytest

Pytest is a popular testing framework for Python programming language. It provides a simple and easy-to-use interface for writing and executing tests. It supports a wide range of testing options and features, including fixtures, parameterization, test discovery, and test coverage analysis. It is designed to be flexible and customizable, allowing developers to write tests in a variety of styles and formats.

As both of the sub modules, the Drowsiness Detection System and the website will be developed using libraries and frameworks of python, the use of pytest streamlines the testing process.

DETAILED DESIGN AND ARCHITECTURE

5.1 SYSTEM ARCHITECTURE

The system consists of an edge device along with a camera. A machine learning model would be deployed on the edge device that would keep check of the driver's behaviour, making sure that the driver is attentive to the road. It will use a buzzer to either the vehicle's stereo system to alert the driver.

5.1.1 Architecture Design Approach

The program logic has been developed using Procedural Design Methodology.

Procedural Design Methodology is a systematic and structured approach to designing and developing computer programs or systems. It involves defining a series of procedures to achieve a specific outcome, such as solving a problem or completing a task.

There is no specific "procedural design methodology" that is commonly used in the development of Machine Learning (ML) based real-time applications. However, as Neural Network models can exclusively use functions it can be said to be following the procedural programming paradigm.

5.1.2 Architecture Design & Subsystem Architecture

The higher-level architecture diagram of the system is as follows:

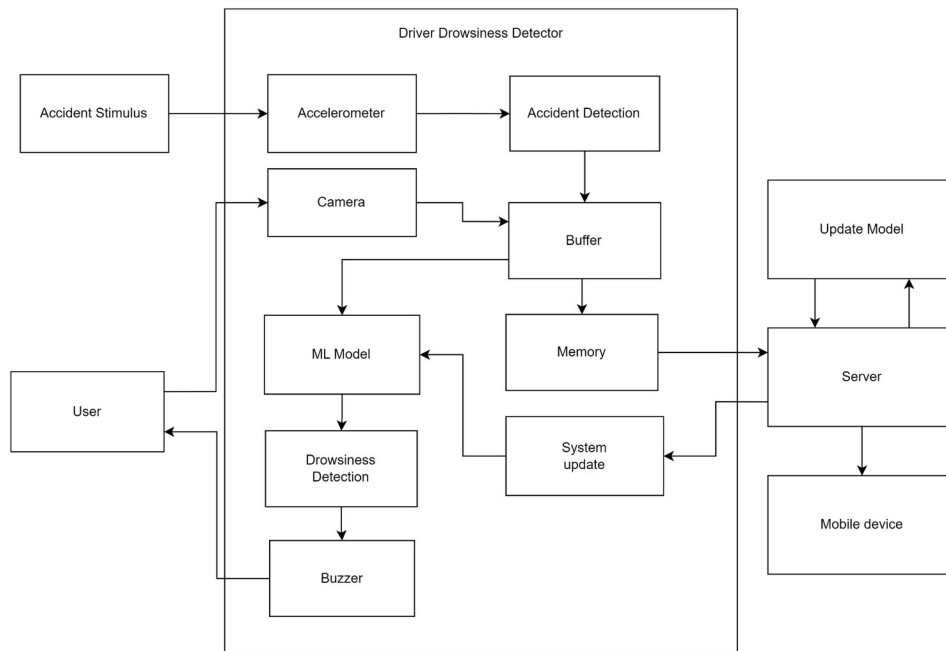


Figure 7 High level architecture Block diagram

The camera records the driver's footage and sends it to the buffer after pre-processing. The model then classifies the image and the drowsiness detection module then determines whether the driver is drowsy or not. The user is alerted to wake up via a buzzer. The accelerometer can detect an accident via deceleration, in which case the system can store the last few seconds of the footage in the memory. The user can then upload that footage on the server. The developers can retrain the model and upload it to the server so the users can update their systems.

The program logic on the edge device follows procedural programming, hence it does not follow a specific architecture. The website on the other hand will use the model view controller architecture.

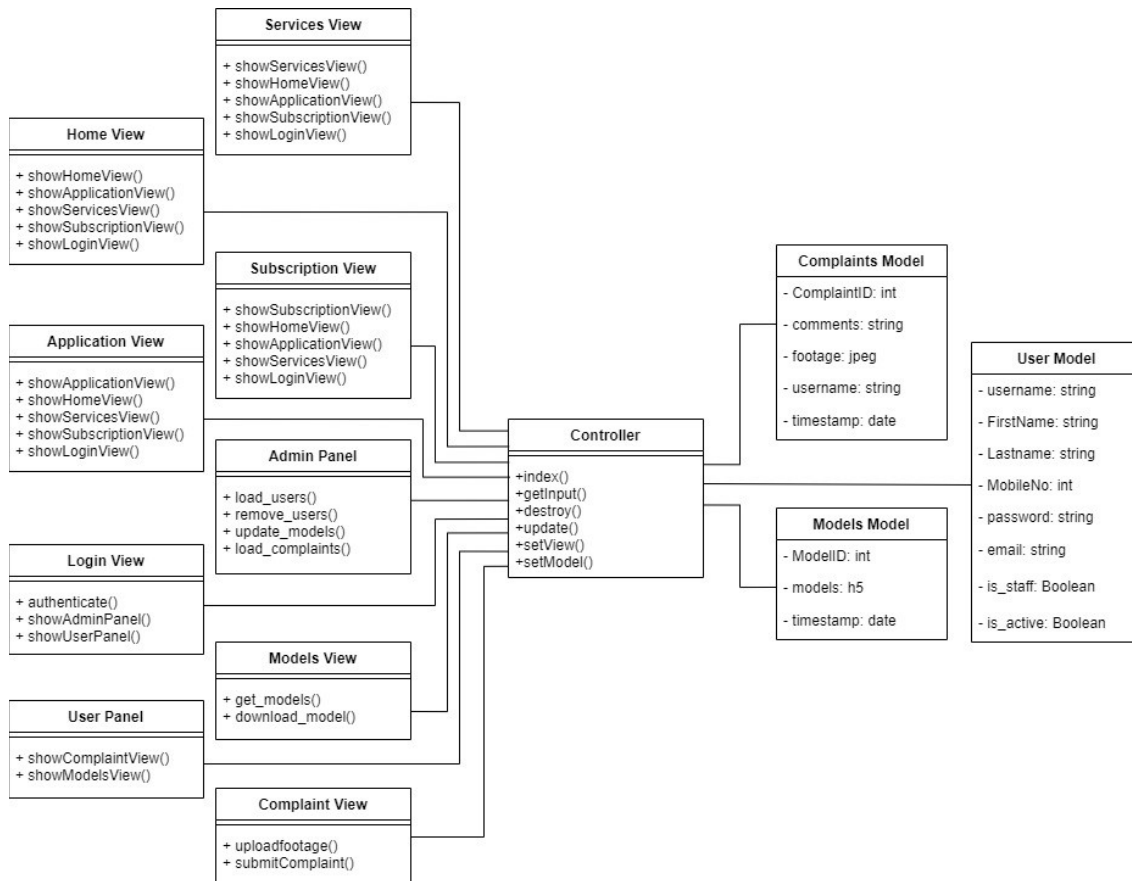


Figure 8 UML Diagram for Website architecture

The website consists of 9 views but 3 models as some views do not have any functionality. The views are what the user sees and interacts with while the controller manages how the view affects the models. It then updates the views to reflect those changes.

The Home View provides an introduction to the project to the website visitor. It links to the Services View, Application View, Subscription View as well as back to itself.

The Services View provides information about the features of our project. It links to the Home View, Application View, Subscription View and lastly back to itself.

The Application View provides information on who can benefit from the use of our project. It links to the Home View, Services View, Subscription View and lastly back to itself.

The Subscription View display subscription plans that the user can opt for the use of our project. It links to the Home View, Services View, Application View and lastly back to itself.

The Login view allows the user to login to the website. It corresponds to the user model. After authentication, it will check if the user is from staff. Upon confirmation, it will link to the admin panel. Otherwise, it will link to the user panel.

The admin panel allows the admin to load users, register new users and remove inactive users. It also allows the admin to load and review complaints as well as issue the updated models. Logging out leads the user back to the Home View.

The User Panel is what the customer would see upon successfully logging in. The user would then have the option to either lodge a complaint by linking to the Complaint View or download the updated models via the Model View.

The Complaint View allows the user to lodge a complaint. The User can upload footage and add comments to supplement their complaint. The user is taken back to the User Panel upon successfully lodging or cancelling their complaint. The Complaint View corresponds to the Complaint Model.

The Model View allows the user to download the updated models. The user is taken back to the User Panel after downloading the model or cancelling their download. The Model View corresponds to the Model Model.

The Controller acts as an intermediary between the Model and the View, handling user input, updating the Model, and updating the View as necessary.

5.1.3 Dataflow of the system

The system consists of two modules, the edge device and the website.

Edge Device:

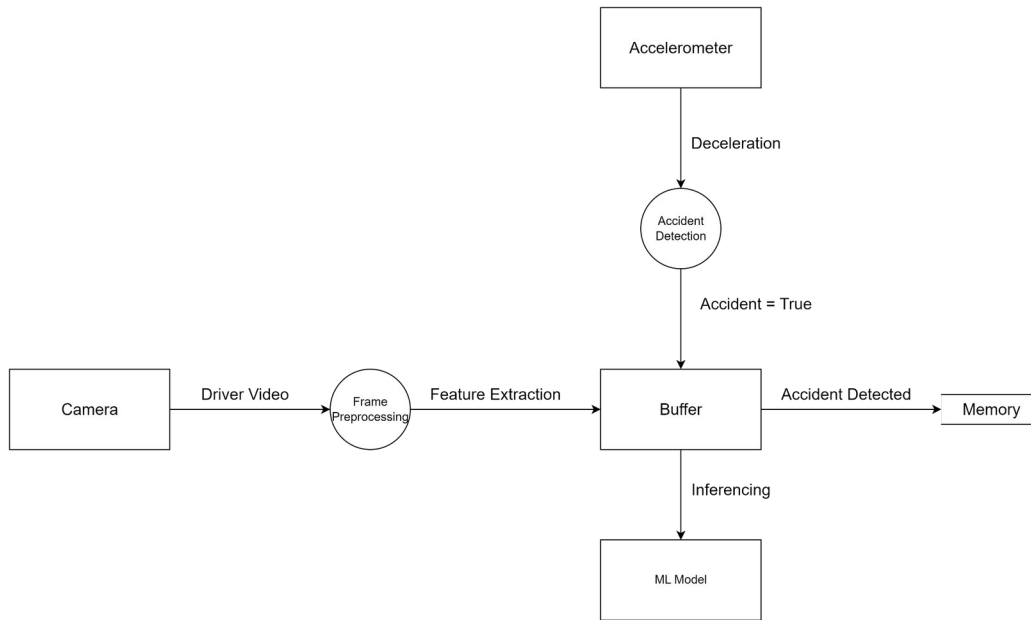


Figure 9 Edge Device Dataflow

The system takes the video stream from the camera and extracts the frames. It then pre-processes the frames to extract relevant features such as the nose and mouth of the driver and to transform these features into a suitable format that is compatible with the Convolutional Neural Networks. These frames are then fed into a buffer which is then fed into the CNN models for driver drowsiness detection. Additionally, in the event of an accident, the accelerometer can detect deceleration and hence instruct the system to write the buffer to the memory which can then be reviewed for inspecting system defects and ensuring quality control.

Website:

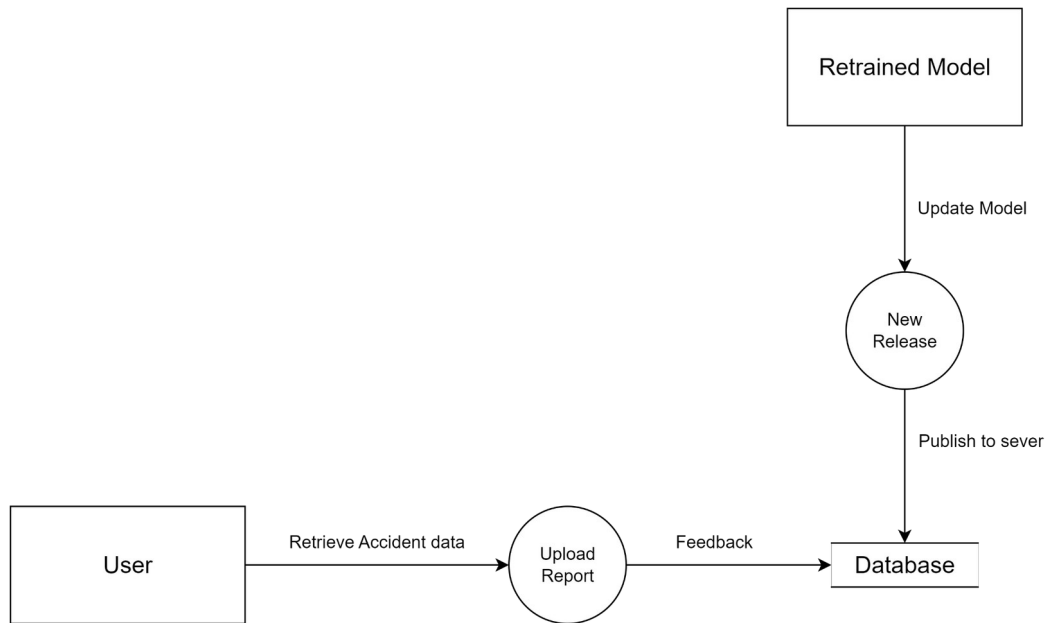


Figure 10 Website Dataflow

As the web ecosystem does not feature extensive functionality, the data flow diagram for it is quite simple. The user can retrieve the accident data from the system and upload the complaint report for product review. If the complaint is appropriate, the developers can then retrain the model and release it on the server for the users to download and update their system.

5.2 DETAILED SYSTEM DESIGN

As the system uses the Procedural Approach, the design will be illustrated by the following diagram.

Activity Diagram:

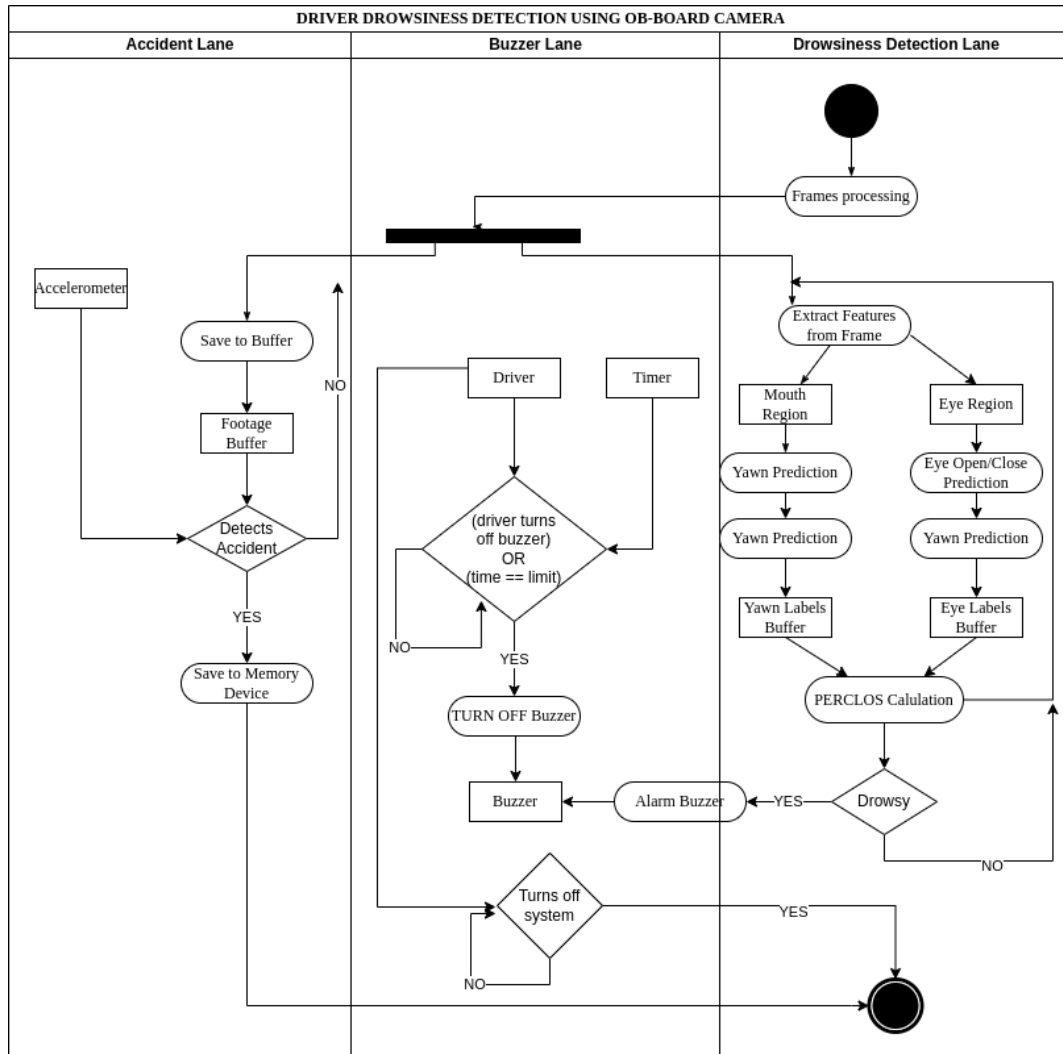


Figure 11 System Activity Diagram

The activity diagram shows the overall program logic. When the program begins, it starts capturing frames from the video using an IR camera. The frames are then further pre-processed to get three separate regions from a person's face; left eye, right eye and mouth. The extracted regions are cropped and then fed to corresponding models to get inference about the eyes being open or closed and whether there is yawning or not.

The frequency of person yawning and closed eyes is continuously calculated inside the buffer to get PERCLOS measurements for the last 5 frames, if PERCLOS value is exceeding the defined limit, then the alarm through buzzer will be generated to alert the drowsy driver till the buzzer is manually turned off by the driver.

Alongside, the frames are being saved to memory buffer and upon detecting the accident through accelerometer, the last 5 seconds footage will be saved to memory device.

As the overview describes most of the functionality, we are going to look at the behavioural design of the system.

5.2.1 Drowsiness Detection

This module uses a camera to detect the user's face. Due to the low resources available, the system records the driver's footage at the standard resolution of 480p at 24 frames per second. The picamera library is used to read the camera stream while OpenCV handles the image manipulation. Frames are extracted from the stream and then converted into a grayscale image. Then we extract the relevant features such as the eyes and the mouth of the driver.

For this, we use the dlib library along with the shape_predictor_68_face_landmarks.dat pre-trained model. To detect the right eye, we extract the landmarks L36 to L41. For the left eye we extract L42 to L47. Lastly for the mouth we extract L48 to L67. These features are then resized to 256 x 256 dimension before being used for inferencing.

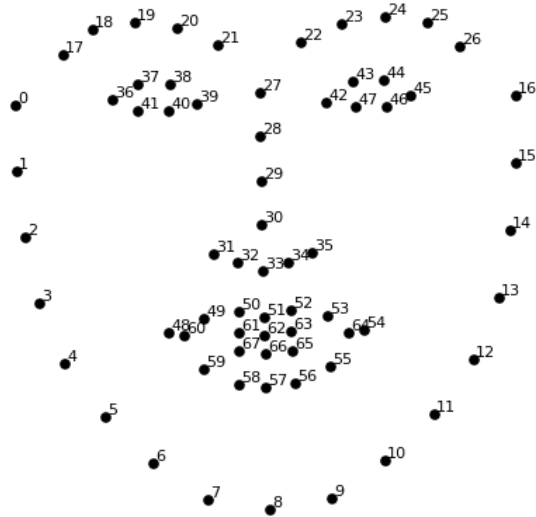


Figure 12 Dlib Facial Landmarks

The Drowsiness Detection module uses 2 convolutional neural networks. Both models follow the same architecture. They comprise of 9 layers, with 3,784,689 trainable parameters. The first layer is a 2D convolutional layer with 128 filters, followed a max pooling layer. Then the feature map passes through three more 2D convolutional layers having 64, 32 and 16 filters respectively. After Dropout, another pooling layer of depth 16 is used. Then the feature map is flatten to be fed into a dense layer of size 64. Lastly, the final layer consists of a single neuron for classification. The final layer uses the sigmoid function while the hidden layers use the ReLU activation function.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 256, 256, 1)]	0
conv2d (Conv2D)	(None, 254, 254, 128)	12880
max_pooling2d (MaxPooling2D)	(None, 127, 127, 128)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	73792
conv2d_2 (Conv2D)	(None, 123, 123, 32)	18464
conv2d_3 (Conv2D)	(None, 121, 121, 16)	4624
dropout (Dropout)	(None, 121, 121, 16)	0
max_pooling2d_1 (MaxPooling2D)	(None, 60, 60, 16)	0
flatten (Flatten)	(None, 57600)	0
dense (Dense)	(None, 64)	3686464
dense_1 (Dense)	(None, 1)	65

Total params: 3,784,689
Trainable params: 3,784,689
Non-trainable params: 0

Figure 13 CNN model architecture

The system infers the both the models and then uses a modified version of the PERCLOS algorithm to predict drowsiness detection. PERCLOS analyses the driver drowsiness level using eye states. The modified version of our algorithm incorporates the yawning metric as well.

The system observes the last five frames in the buffer for both yawning and blinking. Then it uses the formula

$$=0.75(\text{blinkpos_pred} / \text{total blinks}) + 0.25 (\text{yawnpos_pred} / \text{total yawns})$$

Blinkpos_pred is drowsy output from eye model while yawnpos_pred is drowsy output from yawn model. We assigned a higher weightage to the eye model as a drowsy person is more likely to close their eyes than yawn.

If the value is larger than 0.55, the driver is identified to be drowsy.

Behaviour Diagram

When the system is booted, the system is in the Recording state. The video is then preprocessed in the Preprocessing state. These frames are then fed to the ML model in the Prediction state. Now if the Driver is found to be drowsy, the system turns ON the buzzer, then continues inferencing from the Recording State. If the Driver is not drowsy and the buzzer is ON, then the system turns off the Buzzer and continues inferencing from the Recording state. Furthermore, if the buzzer is off and driver is not drowsy, the system will continue inferencing from the Recording State. Lastly, the system can be shut down to enter the terminal state.

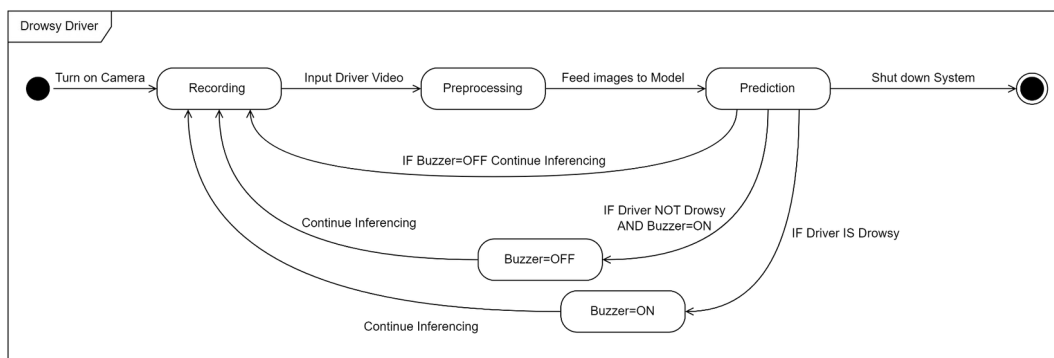


Figure 14 Drowsy Driver Event

5.2.2 Accident Detection

The system detects an accident using readings from the MPU6050 accelerometer. It records the acceleration of the driver every second. If the change in acceleration is greater than 8 meters per second squared, the system saves the footage and shuts down.

The accident threshold was determined to be 8 meters per second squared as it is the minimum deceleration required in case of an accident to deploy the airbags for most passenger cars [8].

Behaviour Diagram

For accident detection, the system can detect an accident by using the accelerometer and enter a saving state where it can save the footage from the buffer into the memory. The system can then be turned off to enter the terminal state.

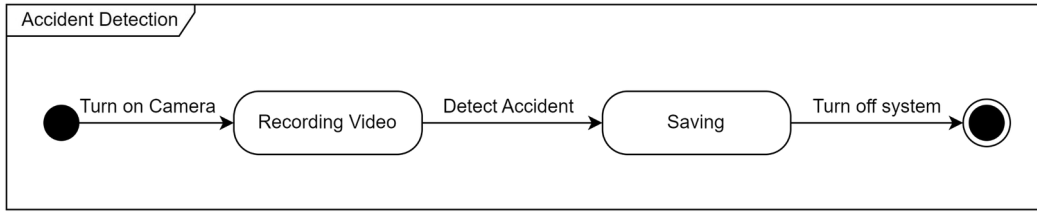


Figure 15 Accident Detection Event

5.2.3 Website

User Interface Design:

As the edge device is an embedded system, it does not have a UI for the user to interact with. The system is going to enact a plug and play approach.

Screen Objects and Actions:

The website consists of 9 views, they are listed below:

1. The Home page:

This page shows the purpose and reason for the project.



Figure 16 Home Page

2. The Services page:

This page showcases the features of the project.

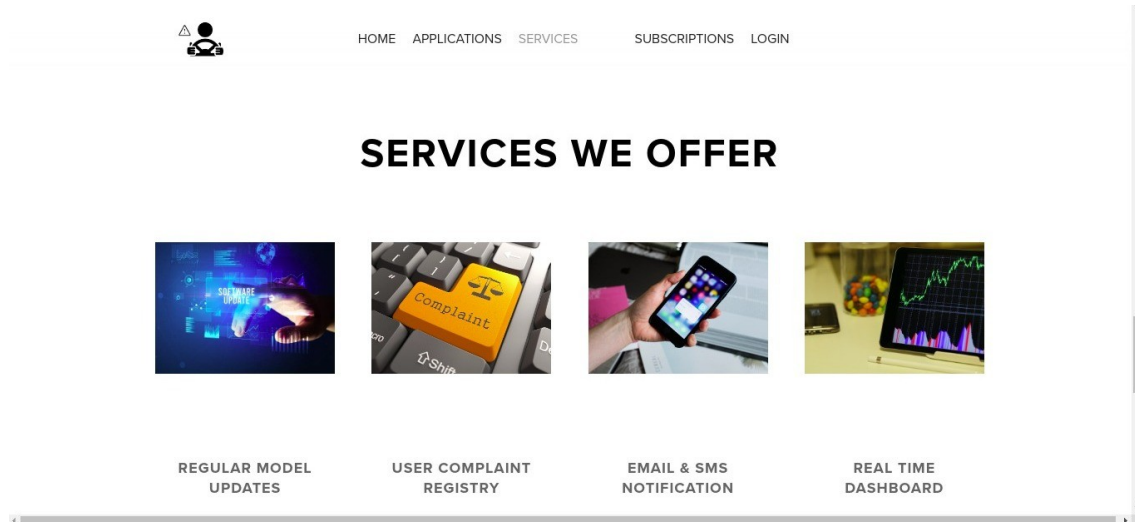


Figure 17 Services Page

3. The Applications page:

This page shows who can benefit from our services in what manner.

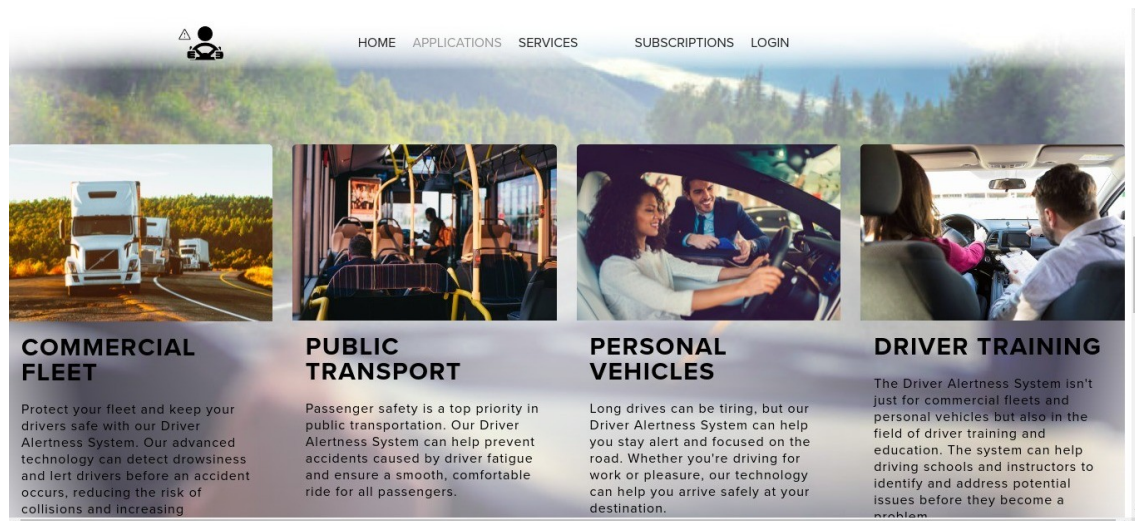


Figure 18 Application Page

4. The Subscription page:

This page allows the user to sign up for the service.

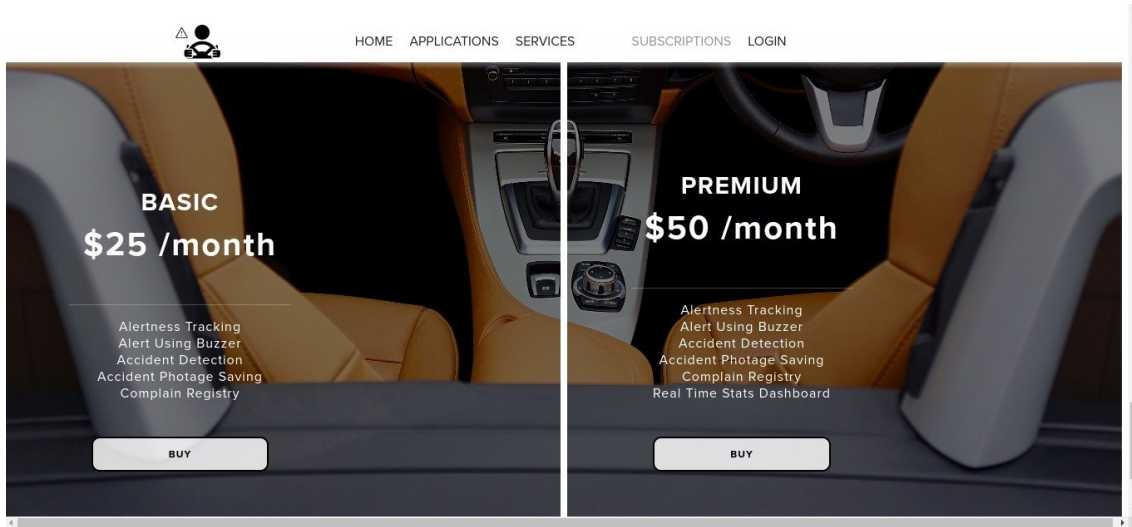


Figure 19 Subscription Page

5. The Login view:

This page allows the user to log in for the service.

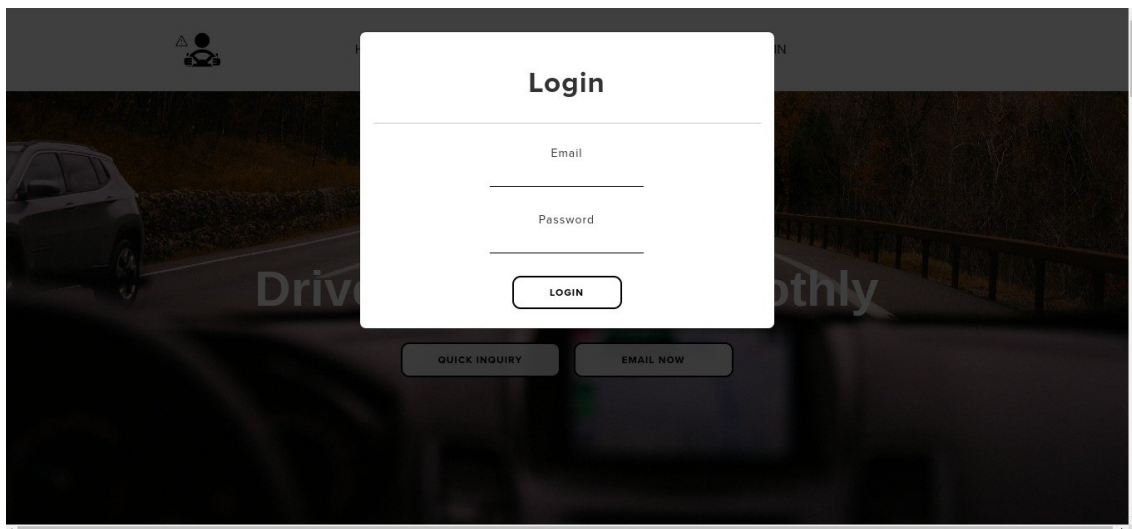


Figure 20 Login View

6. Admin Panel

If the login view detects a super user, they are taken to the admin panel. Here the admin can manage user accounts, view and manage complaints as well as model updates.

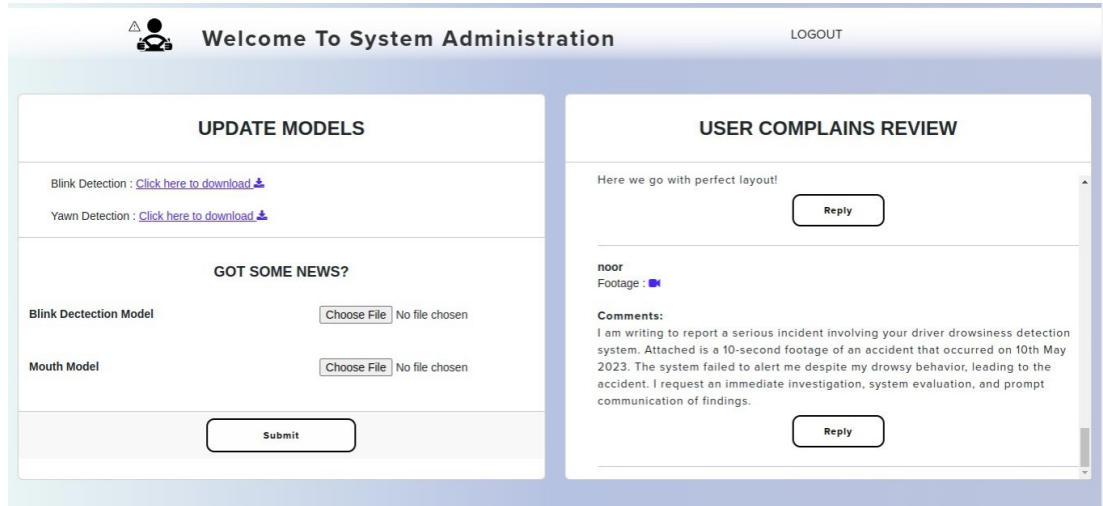


Figure 21 Admin Panel

7. User Panel

What the user sees upon logging in to the portal. Here the user is given options to either lodge a complaint or download model updates.

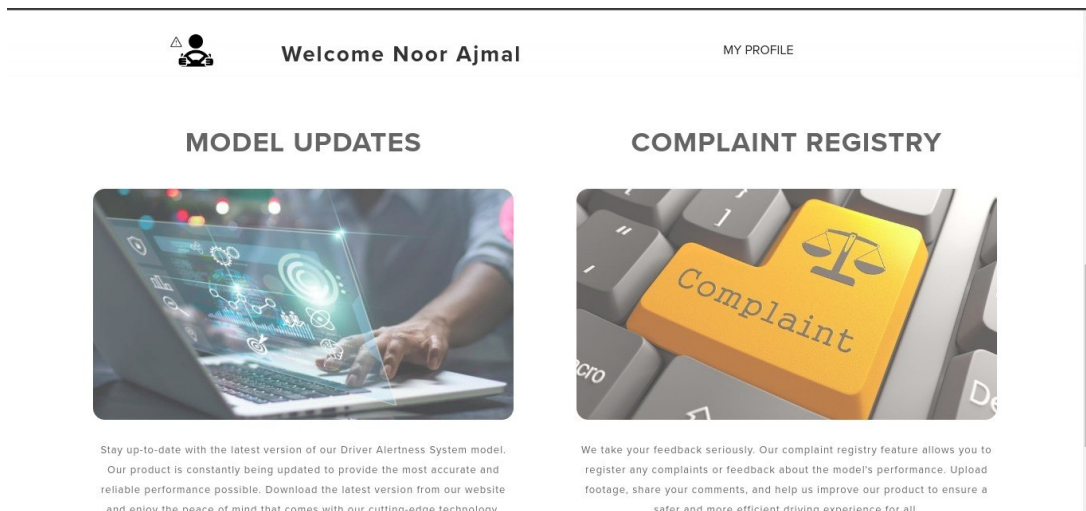


Figure 22 User Panel

8. The Download Models view:

This page contains the download links for the previous models as well as the release notes that denote the updates featured. The user can click on the link to download the new models.

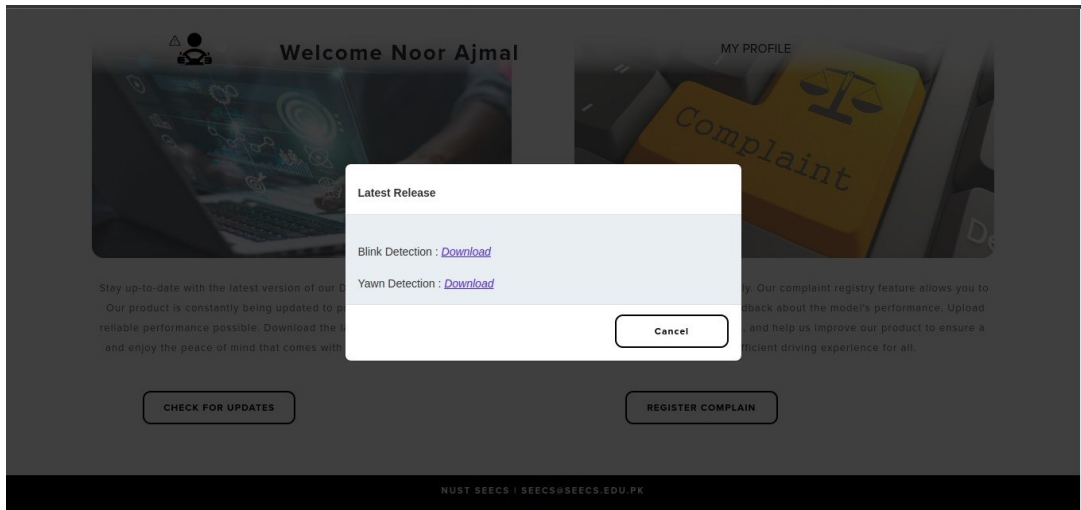
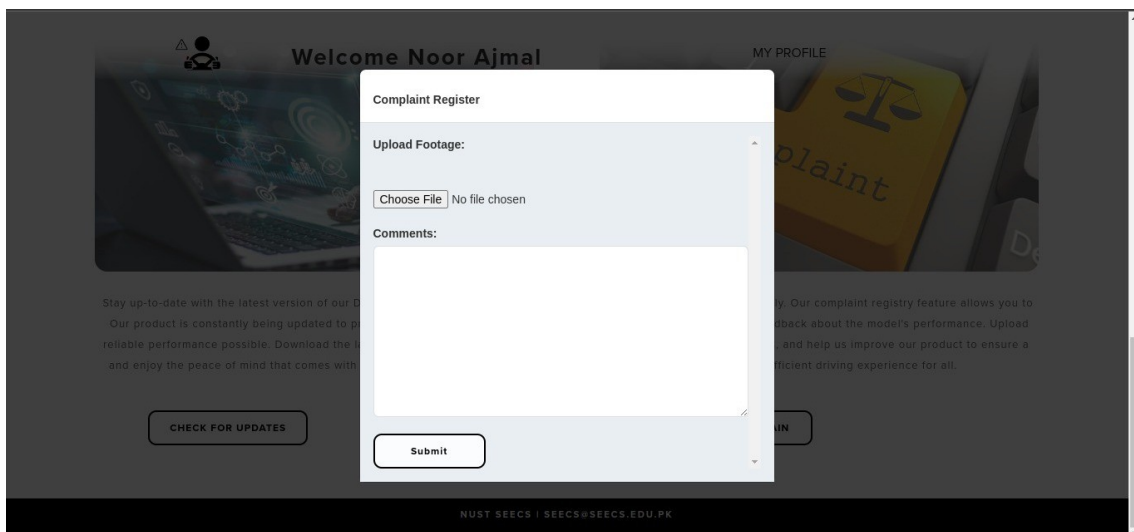


Figure 23 Models View

9. The Complaints view:

This page allows the user to upload the footage of the accident and provide



details about the accident using the comments text box.

Figure 24 Complaints View

IMPLEMENTATION AND TESTING

6.1 Workflow

The development methodology used for this project was Kanban.

Kanban is a software development methodology that emphasizes visualizing the workflow, limiting work in progress (WIP), and continuously delivering small batches of work. It originated in manufacturing, where it was used to optimize production processes, but has since been adapted for use in software development.

The goal was to limit the amount of work in progress at any given time, so that we could focus on completing current tasks before moving on to the next one. This helped to reduce bottlenecks and improve the flow of work through the development process.

The tool used for tracking progress was ASANA. We used five columns to classify tasks on our Kanban whiteboard. The columns were:

- To Do: Tasks that were identified but not yet planned for or started.
- Design: Tasks for which the structure and rubrics were being laid out.
- Development: Tasks for which coding and optimization had begun
- Testing: Modules were tested against the requirements of the project
- Done: All the tasks that had passed the preceding stages and were deemed complete.

The high level structure of our workflow comprised of Research and Overview, Compiling the Dataset, Training and Developing the Model, Integration and Evaluation of system and lastly developing the web portal. These modules are listed in a sequential order.

6.1.1 Research and overview

We read up on related literature and researched products and solutions similar to the ones described above.

6.1.2 Compiling Dataset

The first step is to collect data on how drowsiness affects a person's behaviour. This can include collecting data on eye movements, facial expressions, head movements, and other physical and behavioural indicators of fatigue.

For our system, we collected data for eyes and mouth to detect blinking and yawning. We primarily used datasets from Kaggle, however we augmented them with additional images from sources such as Pinterest, the noun project and other stock image repositories.

For eyes, we used the MRL eye dataset ^[9]. The dataset consists of 84,898 images, collected from 37 different persons, 33 men and 4 women.

For yawning, we had to search quite a bit to gather useful materials. We had to combine two kaggle datasets, Drowsiness_dataset^[10] and Yawning Dataset Classification^[11]. Furthermore, we had to augment this dataset by scrapping images from various sources such as pinterest, thenounproject and other various stock image websites.

6.1.3 Training and developing Model

Model development: Based on the extracted features, we trained a wide variety of models to achieve acceptable performance.

In the end, we settled on CNN models to detect signs of drowsiness. The details of the model are listed in section 5.2.1.

For an objective evaluation, we used yet another dataset to simulate real world driving conditions. The NITYMED dataset ^[12] features 130 videos taken of 21 subjects, 10 females and 11 males, driving at night in Patras, Greece.

6.1.5 Integration and Evaluation of system (edge device)

System integration (edge device): Next we integrate them into a complete drowsiness detection system. This involved integrating the model with hardware components such as the camera and the edge device through the use of some program logic.

6.1.6 Developing the website

Web Development: Finally we develop the companion website that would complete the user experience.

6.2 Testing

The tests were carried out using the pytest library. We prioritized the testing of the functional requirements over non-functional ones. The reason was due to the real time critical nature of our system.


6.2.1 Unit testing

The test cases are listed below along with the corresponding functional requirement.

Drowsiness Detection:

Test Case ID: Fun_11	Test Designed by: Noor ul Huda Ajmal
Test Priority (Low/Medium/High): High	Test Designed date: 12th April 2023
Module Name: Facial Landmarks Extraction	Test Executed by: Noor ul Huda Ajmal
Test Title: The <code>get_facial_landmarks</code> function correctly detects facial landmarks in an input image	Test Execution date: 12th April 2023
Description: The numpy array returned by <code>get_facial_landmarks()</code> function must have 68 rows representing face landmarks.	

Unit Code	<pre>detector = dlib.get_frontal_face_detector() predictor = dlib.shape_predictor("detectors/shape_predictor_68_face_landmarks.dat") # read an image with a face img = cv.imread("img.png") # conversion to gray scale img = get_gray(img) # detect facial landmarks using the function landmarks = get_facial_landmarks(img) # verify that the output is an array of landmarks assert isinstance(landmarks, np.ndarray) assert landmarks.ndim == 2 assert landmarks.shape[1] == 2 # verify that the landmarks are within the expected range assert landmarks[:, 0].min() >= 0 and landmarks[:, 0].max() < img.shape[1]</pre>
------------------	---

	<code>assert landmarks[:, 1].min() >= 0 and landmarks[:, 1].max() < img.shape[0]</code>
Input Data	
Expected Results	2-dimensional numpy array with 68 landmarks of the face
Actual Results	2-dimensional numpy array with 68 landmarks of the face

Test Case ID: Fun_12	Test Designed by: Noor ul Huda Ajmal
Test Priority (Low/Medium/High): High	Test Designed date: 12th April 2023
Module Name: Facial Landmarks Extraction	Test Executed by: Noor ul Huda Ajmal
Test Title: The <code>get_facial_landmarks</code> function correctly handles cases where no face is detected	Test Execution date: 12th April 2023
Description: When there is no face in the input frame/image, the function should return -1	

Unit Code	<pre><code># create an image with no faces img = np.zeros((100, 100, 3), dtype=np.uint8) # detect facial landmarks using the function landmarks = get_facial_landmarks(img) # verify that the output is -1 (indicating no face was detected) assert landmarks == -1</code></pre>
------------------	--

Input Data	
Expected Results	The function would return -1
Actual Results	The function returns -1

Test Case ID: Fun_15	Test Designed by: Noor ul Huda Ajmal
Test Priority (Low/Medium/High): High	Test Designed date: 12th April 2023
Module Name: Drowsiness Detection	Test Executed by: Noor ul Huda Ajmal
Test Title: The program could detect and print the drowsiness level	Test Execution date: 12th April 2023
Description: In case of drowsy person, the program should detect drowsiness and prints drowsiness level	

Unit Code	-
Input Data	video / <class 'cv2.VideoCapture'> with drowsy behaviour of the person
Expected Results	The drowsiness level/percentage must be above 60%
Actual Results	Drowsy Level....0.60% Drowsy Level....0.60% Drowsy Level....0.65% Drowsy Level....0.65% Drowsy Level....0.70% Drowsy Level....0.70%

Alert Driver

Test Case ID: Fun_13	Test Designed by: Noor ul Huda Ajmal
Test Priority (Low/Medium/High): High	Test Designed date: 12th April 2023
Module Name: Buzzer	Test Executed by: Noor ul Huda Ajmal
Test Title: The beep function correctly produces an audible beep	Test Execution date: 12th April 2023
Description: Upon calling beep function, the buzzer should beep for 1sec upon 1kHz frequency	

Unit Code	<pre># set up a mock system call that logs arguments to a list call_log = [] def mock_system_call(cmd): call_log.append(cmd) os.system = mock_system_call # call the buzzer function beep() # verify that the system call was made with the expected arguments assert len(call_log) == 1 assert call_log[0] == "beep -f 1000q -l 1500"</pre>
Input Data	system call
Expected Results	The beep sound of the pre-specified frequency and time
Actual Results	The beep sound of the pre-specified frequency and time

Accident Detection

Test Case ID: Fun_13	Test Designed by: Shaheer Ahmed
Test Priority (Low/Medium/High): Medium	Test Designed date: 12th April 2023
Module Name: Accelerometer	Test Executed by: Shaheer Ahmed
Test Title: Accident detection	Test Execution date: 12th April 2023
Description: The system should detect an accident is the change in deceleration is greater than 8 meters per second squared.	

Unit Code	<pre> import pytest from accident import * # import mock # from mock import patch @pytest.fixture def accel_data(): return {"x": 1, "y": 2, "z": 3} def test_changeaccel(accel_data): # Initial values of x, y, and z are set to 0 global x1, y1, z1 x1 = 12 y1 = 3 z1 = 1 # Set the acceleration data to exceed the threshold value of 8 accel_data["x"] = 2 accel_data["y"] = 3 accel_data["z"] = 1 # Check if the changeaccel function returns True assert changeaccel(accel_data) == True # Set the acceleration data to be within the threshold value of 8 accel_data["x"] = 1.5 accel_data["y"] = 2.5 accel_data["z"] = 3.5 # Check if the changeaccel function returns False assert changeaccel(accel_data) == False </pre>
Input Data	<pre> x1 = 12 y1 = 3 z1 = 1 accel_data["x"] = 2 </pre>

	<pre> accel_data["y"] = 3 accel_data["z"] = 1 accel_data["x"] = 1.5 accel_data["y"] = 2.5 accel_data["z"] = 3.5 </pre>
Expected Results	Both test cases pass
Actual Results	<p>Both test cases passed</p> <pre> Total number of tests expected to run: 1 Total number of tests run: 1 Total number of tests passed: 0 Total number of tests failed: 1 Total number of tests failed with errors: 0 Total number of tests skipped: 0 Total number of tests with no result data: 0 Finished running tests! </pre>

Test Case ID: Fun_18	Test Designed by: Shaheer Ahmed
Test Priority (Low/Medium/High): Medium	Test Designed date: 12th April 2023
Module Name: Footage	Test Executed by: Shaheer Ahmed
Test Title: Accident detection	Test Execution date: 12th April 2023
Description: The system should save the footage of the driver upon detecting an accident.	

Unit Code	<pre> def testsavefootage(accel_data): # Initial values of x, y, and z are set to 0 accident.x1 = 12 accident.y1 = 13 accident.z1 = 1 # Set the acceleration data to exceed the threshold value of 8 accel_data["x"] = 2 accel_data["y"] = 3 accel_data["z"] = 1 # Check if the changeaccel function calls save footage with patch.object(accident, 'savefootage') as mock: changeaccel(accel_data) mock.assert_called() </pre>
------------------	--

Input Data	<pre>x1 = 12 y1 = 13 z1 = 1 accel_data["x"] = 2 accel_data["y"] = 3 accel_data["z"] = 1</pre>
Expected Results	Test case is passed
Actual Results	<pre>Test case is passed Total number of tests expected to run: 1 Total number of tests run: 1 Total number of tests passed: 0 Total number of tests failed: 1 Total number of tests failed with errors: 0 Total number of tests skipped: 0 Total number of tests with no result data: 0 Finished running tests!</pre>

6.2.2 System Testing

As the website had used the Model View Controller (MVC) architecture, the components are interdependent. As such, we performed end to end testing to test the functional requirements of the website.

Updates portal:

Test Case ID: Fun_16	Test Designed by: Shaheer Ahmed
Test Priority (Low/Medium/High): High	Test Designed date: 20 th March 2023
Module Name: Models	Test Executed by: Shaheer Ahmed
Test Title: users can download updated models using the model view	Test Execution date: 26 th March 2023
Description: The user can download updated models by clicking the download links in the model view.	

Pre-conditions:

- User has provided valid username and password.
- User has navigated to model view

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to app website link.	None	The browser initiates a download.	Model update is downloaded	Pass	-
2	Click 'download'					

Post-conditions:

Updated Model is downloaded



Complaints Portal:

Test Case ID: Fun_15	Test Designed by: Shaheer Ahmed
Test Priority (Low/Medium/High): High	Test Designed date: 20 th March 2023
Module Name: Complaints	Test Executed by: Shaheer Ahmed
Test Title: users can lodge complaint using complaint view	Test Execution date: 26 th March 2023
Description: The user can lodge a complaint in complaint view by providing footage in a valid format as well as adding comments with up to 500 characters	

<p>Pre-conditions:</p> <ul style="list-style-type: none"> User has provided valid username and password User navigates to the complaints view
<p>Dependencies:</p>

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to app website link.	Uploaded footage.jpeg				-
2	Upload a file with a valid file format.	Entered "Accident occurred at 9 th Avenue on 11 th March. I don't understand why the buzzer did not work".	Complaint is submitted.	User is directed to user panel after complaint is submitted.	Pass	
3	Enter comments.					
4	Click 'submit'.					

<p>Post-conditions: Complaint is submitted</p>

RESULTS AND DISCUSSION

The eyes model has an accuracy of 87% while the yawn model has an accuracy of 88%. As there is nothing similar available on the market, we cannot make any direct comparisons to any aftermarket systems. The comparison with systems from BMW and Ford would be unfeasible as those vehicles are not available in Pakistan.

The project delivers on all the functional requirements we set out to fulfil. There were a number of challenges faced while making this project.

One of the primary challenges was the lack of good quality public datasets. Although some datasets are available, they may not be representative of real-world scenarios and may have limited data variability. This can affect the accuracy of the model. Curating a dataset manually would not only be time consuming but also expensive, considering the amount of participants and equipment that would be required to collect data that has good class distribution and is usable.

In addition to the lack of good quality public datasets, the lack of availability of edge devices was another challenge. Edge devices such as Nvidia Jetson Nano and Raspberry Pi 4 are suitable for running machine learning models in real-time on low-power hardware. However, these devices are expensive and are not easily accessible, making it difficult to develop and test models for drowsiness detection.

Furthermore, stability issues may arise when running machine learning models on low-power systems. The system may freeze or crash due to the limited processing power, which negatively impacts the availability of the system. Therefore, addressing stability issues is an important factor that needs to be taken into consideration while training a model for drowsiness detection.

Lastly, optimizing neural networks for low-power systems is another challenge in training a model for drowsiness detection. The model needs to be designed to operate within the constraints of low-power hardware while still providing accurate results. This is a complex task as one would need in-depth knowledge about the hardware capabilities of the system as well as the operating conditions that the system would normally experience. This can be a complex task that requires extensive knowledge of machine learning and computer vision algorithms. Further research is required to

develop better methods for optimizing neural networks for low-power systems.

CONCLUSION AND FUTURE WORK

8.1 Conclusion

The Driver Drowsiness detection system was developed to provide drivers with a plug and play aftermarket system that detects drowsiness and prevents road accidents. This is achieved through the marriage of Internet of Things (IoT) and Machine Learning. Furthermore, the implementation of Continuous Quality Improvement (CQI) breathes an air of innovation to this previously stale field.

8.2 Impact on Society

The implementation of drowsiness detection systems in vehicles has the potential to significantly impact society in various ways. Primarily, such systems can lead to an increase in safety on the roads by detecting when a driver is getting drowsy and alerting them or taking corrective action. Drowsy driving is a major cause of accidents, injuries, and fatalities on the roads. Therefore, the use of drowsiness detection systems could potentially prevent many accidents, leading to a reduction in the overall burden of injury and loss of life due to traffic accidents.

Furthermore, drowsiness detection systems can contribute to the improved well-being of drivers. By alerting drivers when they are getting drowsy, these systems can encourage drivers to take breaks or stop driving altogether, thereby avoiding the physical and mental strain of driving while fatigued. This, in turn, can lead to a reduction in stress, improved mental health, and an overall enhancement of the quality of life for drivers.

In addition, the implementation of drowsiness detection systems could also potentially lead to increased productivity. By allowing drivers to drive for longer periods without risking drowsiness-related accidents, these systems can help improve efficiency and reduce costs for businesses, especially for commercial drivers such as truck drivers who need to cover long distances in a short amount of time.

The potential economic benefits of drowsiness detection systems should also be considered. The reduction of the number of accidents and fatalities on the roads can help reduce the burden on healthcare systems, insurance providers, and other sectors that are affected by traffic accidents.

Additionally, the increased safety and well-being of drivers can lead to increased productivity, which can help boost economic growth.

8.3 Future Work

As with any project, this system has room for improvement. In the future, we would like to:

- Implement the system as a self-contained system with added functionality, for example lane assist and blind spot monitoring.
- Improve program optimization so we can use even lower power hardware to reduce system costs
- Provide automatic updates using Over The Air (OTA) nodes for areas with good connectivity

REFERENCES

- [¹] Milenkovic, D. (2022, February 23). Distracted Driving Statistics & Facts for 2022. Carsurance. Retrieved from <https://carsurance.net/insights/distracted-driving-statistics/#:%7E:text=%E2%80%93%20Every%20year%2C%20there%20are%202.5,driver%20in%20the%20United%20States>.
- [²] World Health Organization. (2011). Global status report on road safety 2013: supporting a decade of action. <https://www.who.int/publications/i/item/9789241564564>
- [³] National Highway Traffic Safety Administration. (n.d.). Drowsy Driving. <https://www.nhtsa.gov/risky-driving/drowsy-driving>
- [⁴] European Transport Safety Council. (2020). PIN Flash Report 43: Progress in reducing deaths and serious injuries on EU roads. <https://etsc.eu/pin-flash-report-43-progress-in-reducing-deaths-and-serious-injuries-on-eu-roads/>
- [⁵] BMW upgrade measures taking effect from summer 2013. (n.d.). BMW Group PressClub. Retrieved from, <https://www.press.bmwgroup.com/global/article/detail/T0141144EN/bmw-model-upgrade-measures-taking-effect-from-the-summer-of-2013>
- [⁶] Ford's Wake-Up Call for Europe's Sleepy Drivers. (n.d.). Retrieved from https://web.archive.org/web/20110513232258/http://media.ford.com/article_print.cfm?article_id=34562
- [⁷] Driver Fatigue and Drowsiness Monitoring System (Canceled). (n.d.). Kickstarter. Retrieved from <https://www.kickstarter.com/projects/hightech/ai-powered-plug-n-play-driver-fatigue-monitoring-system>
- [⁸] Insurance Institute for Highway Safety (IIHS). (n.d.). Airbags. Retrieved from <https://www.iihs.org/topics/airbags#:~:text=Typically,%20a%20front%20airbag%20will,up%20to%20these%20moderate%20speeds>.
- [⁹] MRL Eye Dataset. MRL. (n.d.). <http://mrl.cs.vsb.cz/eyedataset>
- [¹⁰] Drowsiness_dataset. (2020, September 27). Kaggle. <https://www.kaggle.com/datasets/dheerajperumandla/drowsiness-dataset>
- [¹¹] Yawning Dataset Classification. (2023, April 3). Kaggle. <https://www.kaggle.com/datasets/deepankarvarma/yawning-dataset-classification>
- [¹²] NITYMED. (2022, July 9). Kaggle. <https://www.kaggle.com/datasets/nikospetrellis/nitymed>

