



INFORMATICS
INSTITUTE OF
TECHNOLOGY

UNIVERSITY OF
WESTMINSTER

Informatics Institute of Technology

Module: Software Development Group Project

Module Code: 5COSC009C

Module Leader: Banuka Athuraliya

Implementation Report

Topic: Driverdroid - Driver Drowsiness Detection System

Date of Submission: 23/04/2021

Group Name: HelaDevs

Group members

1. Odhil Weerakoon - W1761129 (2019488)
 2. Rusiru Wijethilake - W1761195 (2019162)
 3. Ravindu Bandara - W1761190 (2019123)
 4. Sanduni Bhagya - W1761916 (2018434)
 5. Geethika Suharshani - W1761082 (2019293)
 6. Dilanka Wanigasekara - W1761078 (2019219)
-

Abstract

Driver drowsiness is one of the highest causes of vehicle accidents all around the world. According to worldwide reports, the rate at which accidents are caused by driver drowsiness is growing year by year and has been a critical issue. To reduce this rate different approaches have been made to action, such as introducing new laws and regulations, conducting surveys worldwide and making every person aware of this critical issue.

Surveys Conducted have indicated that driver drowsiness is caused mainly due to drunkenness, lack of sleep, depression, tiredness, and loneliness. Many companies have already produced many systems that can detect and warn the driver when they are drowsy, but those products are either cost highly or have low accuracy.

‘HelaDevs’ Driver Drowsiness Detection System is a solution to tackle this same problem but with more efficiency and with more accuracy. ‘HelaDevs’ Driver Drowsiness Detection System is not only more efficient but also uses high-end low-cost hardware to detect and alarm the user when any state of drowsiness is detected.

Table of Contents

Abstract.....	i
Table of Contents	ii
List of Tables	v
List of Figures.....	vi
Chapter 01 – Implementation	1
1.1 Chapter Overview	1
1.2 Overview of the prototype.....	1
1.3 Technology selections	1
1.3.1 Driverdroid IoT Device Components Selections.....	1
1.3.2 Software and Operating System selection	2
1.3.3 Language selection	4
1.3.4 Libraries / Frameworks selection	5
1.3.5 Third Party APIs	6
1.3.6 Summary of components in full Driverdroid product	6
1.4 Implementation of the data science component	7
1.4.1 First Stage	7
1.4.2 Final Stage	9
1.5 Implementation of the backend component	11
1.6 Implementation of the mobile application component.....	14
1.6.1 Implemented Rest Places screen.....	15
1.6.2 Implemented Driver History screen	19
1.6.3 Implemented Driver History Statistics screen	22
1.7 Deployments/ CI-CD Pipeline	24
1.8 Chapter Summary.....	25
Chapter 02 – Testing.....	26
2.1 Chapter Overview	26
2.2 Testing Criteria.....	26
2.3 Testing functional requirements	26
2.4 Testing non-functional requirements	27
2.5 Unit testing	29
2.6 Performance testing.....	29

2.7 Usability testing.....	30
2.8 Compatibility testing	35
2.9 Chapter Summary.....	36
Chapter 03 – Evaluation.....	37
3.1 Chapter Overview	37
3.2 Evaluation methods	37
3.3 Quantitative evaluation	37
3.4 Qualitative evaluation	37
3.5 Self evaluation.....	42
3.6 Chapter Summary.....	43
Chapter 04 – Conclusion	44
4.1 Chapter Overview	44
4.2 Achievements of aims and objectives	44
4.3 Legal, social, ethical and professional issues	44
4.3.1 Legal	44
4.3.2 Social	45
4.3.3 Ethical	45
4.3.4 Professional	45
4.4 Limitations of the research	46
4.4.1 Limitation of hardware	46
4.4.2 Limitation of language.....	46
4.4.3 Cost of the prototype	46
4.4.3 Limitation of the dataset	46
4.5 Future enhancements.....	46
4.5.1 Improve the accuracy.....	46
4.5.2 Decrease the cost	46
4.5.3 Increase the performance.....	47
4.5.4 Improve user experience.....	47
4.6 Extra work	47
4.7 Concluding remarks	47
References	48
Bibliography	49
Appendix Section A - Implementation	i

Appendix Section A.1 – Implementation of the mobile application component	i
A.1.1 Implemented Sign up screen	i
A.1.2 Implemented Verify Device screen	iii
A.1.3 Implemented Register Device screen	vi
A.1.4 Implemented Sign In screen	viii
A.1.5 Implemented Forgot Password screen.....	x
A.1.6 Implemented Reset Password screen	xiv
A.1.7 Implemented Home screen.....	xvi
A.1.8 Implemented About screen.....	xvii
A.1.9 Implemented Contact Us screen.....	xviii
A.1.10 Implemented Log Out screen	xx
Team contribution on the report	xxii

List of Tables

Table 1: functional requirements testing	27
Table 2: non-functional requirements testing	29

List of Figures

Figure 1: saving dataset features into bottleneck file.....	8
Figure 2: training dataset bottleneck file based on the sequential model method	9
Figure 3: accuracy of the data science model	9
Figure 4: 68 facial landmarks file	9
Figure 5: ear formula	10
Figure 6: points on the eye.....	10
Figure 7: capturing the mouth from the frames	10
Figure 8: capturing the eyes from the frame	10
Figure 9: if condition to determine the drowsiness.....	11
Figure 10: voice function to sound the alert voice.....	11
Figure 11: user api class image1	12
Figure 12: use api class image2	12
Figure 13: backend network.....	13
Figure 14: project structure of the mobile application.....	15
Figure 15: request location access image.....	16
Figure 16: nearby rest places image.....	16
Figure 17: start getting directions image	17
Figure 18: get directions to rest place image	17
Figure 19: getting directions image	18
Figure 20: code to request location permission from user	18
Figure 21: code to get directions.....	19
Figure 22: code to get nearby rest places	19
Figure 23: latest drowsiness warning image	20
Figure 24: no warnings history image	20
Figure 25: no drowsiness warnings list image	21
Figure 26: code to get warnings history data	22
Figure 27: code that extract drowsiness warnings history details from the navigation routes	23
Figure 28: driver warnings history image	23
Figure 29: code to populate list with warnings data	24
Figure 30: CICD pipeline.....	24
Figure 31: AWS CICD pipeline.....	25
Figure 32: test result 1.....	29
Figure 33: test result 2.....	29
Figure 34: usability testing form image1	30
Figure 35: usability testing form image2	31
Figure 36: usability testing form image3	32
Figure 37: usability testing form image4	33
Figure 38: questionnaire result analysis image1	34
Figure 39: questionnaire result analysis image2.....	34

Figure 40: questionnaire result analysis image3	35
Figure 41: qualitative evaluation form image1	38
Figure 42: qualitative evaluation form image2	39
Figure 43: qualitative evaluation form image3	40
Figure 44: qualitative evaluation form image4	41
Figure 45: questionnaire result analysis image	41
Figure 46: qualitative evaluation feedback	42
Figure 47: sign up screen	i
Figure 48: sign up screen input validation	ii
Figure 49: user already exist alert	ii
Figure 50: phone no. already exist alert	iii
Figure 51: verify device screen	iv
Figure 52: invalid device id alert	v
Figure 53: verify device validations	v
Figure 54: already registered device alert	vi
Figure 55: register device screen	vii
Figure 56: invalid otp alert	viii
Figure 57: register device validations	viii
Figure 58: sign in screen	ix
Figure 59: sign in screen validations	x
Figure 60: no such user account alert	x
Figure 61: forgot password screen	xi
Figure 62: forgot password validations	xii
Figure 63: invalid username alert	xii
Figure 64: temp password sent alert	xiii
Figure 65: reset password screen	xiv
Figure 66: reset password validations	xv
Figure 67: invalid temp password alert	xv
Figure 68: reset password alert	xvi
Figure 69: home screen	xvii
Figure 70: drawer navigator	xvii
Figure 71: about screen	xviii
Figure 72: contact us screen	xix
Figure 73: contact us screen validations	xix
Figure 74: sent feedback alert	xx
Figure 75: log out screen	xxi

Chapter 01 – Implementation

1.1 Chapter Overview

This chapter will discuss and elaborate on the implementation of the prototype, chapter will focus on overview of prototype, implementation of the actual prototype which consist of mobile application, backend and data science component, technology selections for the implementation and also discuss on deployment of CICD pipeline for the project.

1.2 Overview of the prototype

The raspberry pi module will detect the users face and extract the eye feature from the isolated frames of the video feed.

The graphical user interface of the mobile app provides the user with convenient access of to the history of the user's drowsiness habit.

The back end will gather the location data of the user and send it to the aws server. This allows the user to access the location and drowsiness data from the front end.

1.3 Technology selections

Technology selections for Driverdroid has detailed below.

1.3.1 Driverdroid IoT Device Components Selections

Raspberry Pi 4 Model B - 4GB variant

Driver Drowsiness Systems are depending on image processing to work. The Raspberry Pi 4 Model B has enough computing power and memory to complete those tasks, as well as enough GPIO headers for sensors and other hardware.

SIM700c 4GB NB-IoT 4G/eMTC/EDGE/GPRS/GNSS LTE Module

Many features are included in this SIM7600 model that are needed for the device to run properly and smoothly. This module offers up to 4GB LTE connectivity, with maximum download speeds of 150Mbps and upload speeds of 50Mbps.

5MP with IR Lights Camera Module for Raspberry Pi

The Raspberry Pi 4 model B board has a camera header that can be used to link the 5MP with IR Lights camera module. This module includes a Sony IMX219 camera sensor with an integrated IR sensor for use in low-light conditions that can be turned automatically.

Obstacle Avoidance Sensor IR Infrared Module

To accurately position the driver with the camera inputs, the IR Infrared module is used.

5-inch Raspberry Pi Touch Screen Display

The Raspberry Pi touchscreen control module, which is used to display and get user inputs through a GUI interface, provides simple touch inputs and a high-quality display in the same module.

1.3.2 Software and Operating System selection

Raspberry Pi OS

For regular usage on a Raspberry Pi, Raspberry Pi OS is the recommended operating system. Raspberry Pi OS is a Debian-based free operating system that is optimized for the Raspberry Pi hardware. Over 35,000 packages are included with Raspberry Pi OS, which are precompiled software bundles in a nice format for quick installation on your Raspberry Pi. Raspberry Pi OS was used as the base OS on the raspberry pi module.

PyCharm

PyCharm is a Python Integrated Development Environment (IDE) that includes a variety of important resources for Python developers that are tightly integrated to build a convenient environment for efficient Python, web, and data science development.

Anaconda

Anaconda is a Python and R programming language distribution aimed at simplifying package management and deployment in scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, and so on).

Cloud Platform - AWS - EC2, CodeDeploy, S3, RDS

CodeDeploy and codebuild within AWS code pipeline was used to maintain CICD and deployed in EC2(Elastic Compute cloud) and S3 simple storage service was used to store created jar files within deployment.

IntelliJ IDEA

IntelliJ IDEA examines the code, searching for symbols that are linked to all project files and languages. It provides in-depth coding assistance, fast navigation, clever error analysis, and, of

course, refactoring's using this knowledge. IntelliJ IDEA was used to develop the GUI of the IoT device and implementation of the backend component.

DataGrip

DataGrip is a database management system designed for programmers. It's made for querying, creating, and managing databases. Locally, on a computer, or in the cloud, databases may be used. MySQL, PostgreSQL, Microsoft SQL Server, Oracle, and other databases are supported. DataGrip software was used to access and manipulate data of the hosted database.

XAMPP

XAMPP is a free and open-source cross-platform web server solution stack kit created by Apache Friends, which consists primarily of the Apache HTTP Server, MariaDB database, and interpreters for PHP and Perl scripts. XAMPP was used to access local database of the project.

Git

Git is a software that allows you to monitor changes in any collection of files. It's typically used to coordinate work among programmers who are working on source code together during software development. Speed, data integrity, and support for distributed, non-linear workflows are among its objectives.

Postman

Postman is a powerful API client that makes creating, sharing, testing, and documenting APIs simple for developers. Users can build and save easy and complex HTTP/s requests, as well as read their replies, to accomplish this. Therefore, work is more productive and less challenging. Visual Studio Code was used to test APIs for the project.

Visual Studio Code

Microsoft's Visual Studio Code is a freeware source-code editor for Windows, Linux, & macOS. Debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git are among some of the features. Users can customize the theme, keyboard shortcuts, and preferences, as well as install plugins that add new features. Visual Studio Code was used as the IDE for the coding of React Native.

Expo CLI and Expo Go App

Expo CLI is a command-line application that serves as the primary interface between developers and Expo tools. It will be used for several activities, including the development of new projects. Running the project server, viewing logs, and running the app in a simulator are

all steps in the development process. Expo CLI was used as the key interface between a developer and Expo tools is a command line app.

1.3.3 Language selection

Python was chosen as the main programming language for the implementation of the data science component of this project, after an evaluation of the available programming languages.

Python was chosen primarily for the following reasons:

- Libraries and tools such as Tensorflow and Jupyter are available.
- Python is becoming more common in both industry and academia due to its strength and flexibility.
- Python has a large group of developers and data scientists, as can be seen by sites such as Python.org, Fullstackpython.com, and realpython.com.

Java - Spring boot was chosen as the main programming language for the implementation of the data science component of this project. And MySQL has used to create the database and manipulate data, that receive through data science component and frontend.

Java - Spring boot was chosen primarily for the following reasons:

- Reduces the amount of time spent on production and improves the development team's overall performance.
- Assists in the automatic configuration of all components for a production-ready Spring app.
- Provides a default setup for unit and integration tests, making it easier to create and test Java-based applications.

MySQL was chosen primarily for the following reasons:

- MySQL is widely regarded as the safest and most dependable database management system for common web applications.
- MySQL has unrivaled scalability, allowing you to run deeply embedded apps with a smaller footprint, even in large warehouses with terabytes of data.
- MySQL provides a variety of high-availability options, such as cluster servers and master/slave replication setups.

React and JavaScript was chosen as the main programming language for the implementation of the mobile application of this project, after following a review of the programming languages that are available.

React and JavaScript (React Native) were chosen primarily for the following reasons:

- React Native bridges the gap between web UI components and their Java/Swift native counterparts.
- To build React Native apps, JavaScript developers need less specialized knowledge.
- Can write a single codebase for both Android and iOS.

1.3.4 Libraries / Frameworks selection

NumPy

NumPy is a Python library that allows you to work with arrays. It also has functions for dealing with matrices, fourier transforms, and linear algebra. It is an open-source project that you are free to use. Numerical Python is referred to as NumPy. NumPy was included in the project for scientific computing tasks.

Dlib

DLib is an open-source C++ library that implements machine learning algorithms such as classification, regression, clustering, data transformation, and organized prediction.

Keras

Keras is a free open-source Python library for developing and testing deep learning models that is both efficient and simple to use. It wraps Theano and TensorFlow, two powerful numerical computation libraries, and allows you to define and train neural network models in just a few lines of code.

TensorFlow

TensorFlow is a machine learning software library that is free and open source. It can be used for a variety of activities, but it focuses on deep neural network training and inference. Tensorflow is a dataflow and differentiable programming-based symbolic math library.

OpenCV

OpenCV-Python is a Python bindings library for computer vision. Numpy, a highly optimized numerical operations library with a MATLAB-style syntax, is used by OpenCV-Python. Numpy arrays are used to transform all OpenCV array structures to and from.

Spring Boot

The Spring Framework is a Java platform programming framework and inversion of control jar. Any Java application may use the framework's core features, but there are enhancements for developing web apps on top of the Java EE platform. Backend component of this project has implemented using java language Springboot framework.

React Native

React Native was used for the implementation of the Driverdroid mobile application. React Native is an open-source mobile application framework created by Facebook. It allows developers to use React's framework alongside native platform capabilities to create apps for Android, Android TV, iOS, macOS, tvOS, Web, Windows, and UWP.

1.3.5 Third Party APIs

Json Web Token

JSON Web Token is a proposed Internet standard for creating data with optional signature and/or encryption, whose payload contains JSON that asserts a set of statements. A private secret or a public/private key is used to sign the tokens. JWT (Json Web Token) was used for token-based user authorization.

Java Persistence API

The Java Persistence API (JPA) is a Java specification that allows Java objects and classes to access, persist, and manage data in relational databases. As a replacement for the EJB 2 CMP Entity Beans specification, JPA was specified as part of the EJB 3.0 specification. Java Persistence API was used to persist java objects into relational database.

Google Places API

The Places API returns information about places. This API returns data using HTTP requests. Places API has used to get location details in 'Driverdroid' mobile application.

1.3.6 Summary of components in full Driverdroid product

IoT Device

Detect driver drowsiness by observing the driver's facial features, yawning, and eye posture, and sounding an alarm after analyzing the data to alert him that he is not in a good situation to drive.

Data Science Component

VGG16 convolutional neural network model was used in the project and it was retrained by using a new dataset. this produces an accuracy of 95%. The model can identify the drowsiness of the user by analyzing the mouth and head position. Also, the eye aspect ratio is calculated to determine the drowsiness.

Backend Component

Backend component has implemented using java language Springboot framework, backend contents all APIs and logics and the database that frontend needs to call over the GUI (Graphical User Interface). MySQL has used to create the database and manipulate data, that receive through data science component and frontend. CodeDeploy and codebuild within AWS code pipeline is used to maintain CICD and deployed in EC2(Elastic Compute cloud).

Mobile Application

In this app, when a user sign up for the first time, he will be asked to provide a username, telephone no & the password. In this process, all the necessary validations will be done, to validate the user inputs. On successful provision of those details, user will be directed to the verify device screen, where he needs to provide the device id mentioned in the IoT device.

After that, he will be navigated to the register device screen, where he will be prompted to enter the verification code that has sent, to his registered phone number. After successful completion of these steps, the device will be registered under current user, and the user will be navigated to the sign in screen. There he will have to sign in, using his username & password.

If user has forgotten his password, he can reset his password, using the forgot password option. When signed in, user will be directed to the home screen. There he can navigate between screens using a drawer. In rest places screen, if user provide location access, user will be able to see all the nearby restaurants on the map. When he selects one restaurant, user can get directions to that restaurant. In the driver history screen, it will show the latest drowsiness warning, that has given to the user. He can use the find out more button, to see further details of the drowsiness warnings, that has given to him, within a month.

About screen will show, a brief description about the app, while contact us screen will let user, to send his feedback about the app. If user want to log out from the app, he can choose the log out option.

1.4 Implementation of the data science component

1.4.1 First Stage

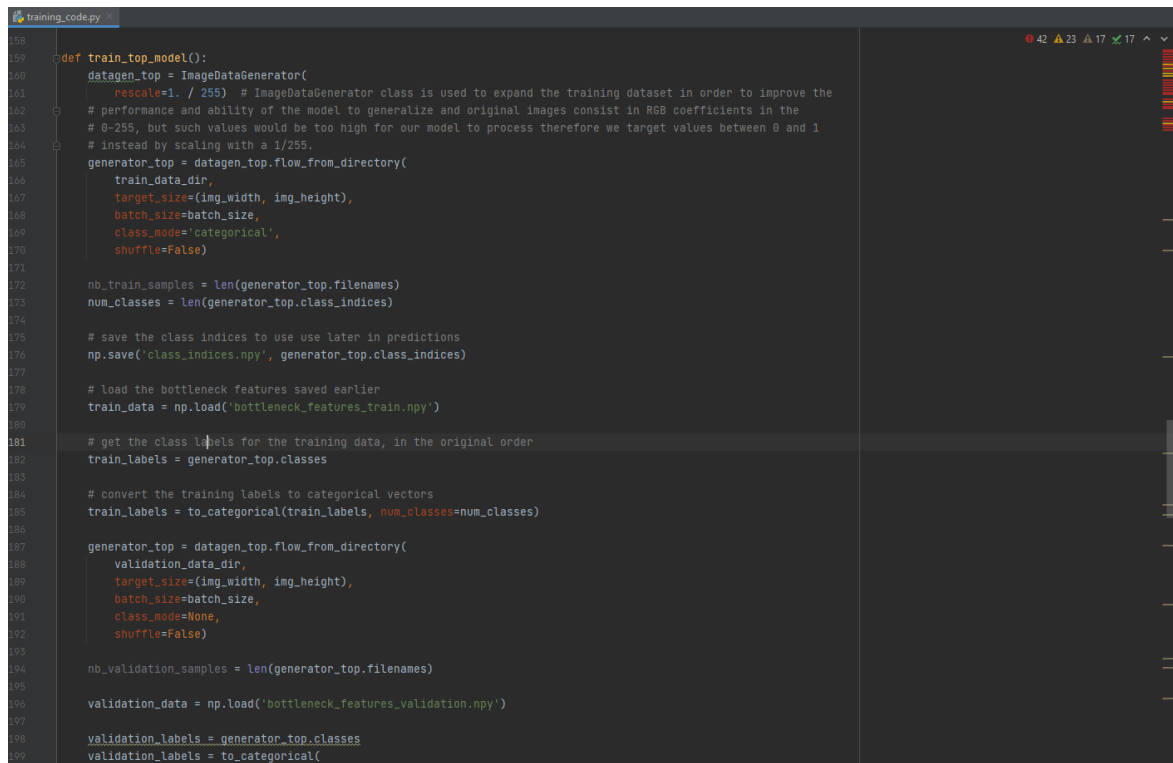
In the First Stage of Implementing the Data Science Component we used our Dataset to Train our prototype. To Train our prototype we used the VGG 16 pretrained model as our convolutional neural network for Image Classification. First, we took a fully connected layer of the VGG16 and created our own fully connected layer with our own custom dataset, using a technique called transfer learning technique. Next, we extracted the Features from our dataset and compressed it and saved it to the last bottleneck feature layer in VGG 16.

To Train our Model to identify drowsiness state during yawning, the images from the Dataset were taken and the flow_from_directory method in Image generator was used to read the

directory path, images and resize our images to the required size and finally saving the features of our dataset into a bottleneck feature file as a NumPy array. The same Procedure is done for the Alert and Drowsy Dataset.

```
training_code.py
70 def save_bottleneck_features(): # for the Yawning Part
71     # build the VGG16 network
72     model = applications.VGG16(include_top=False,
73                               weights='vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5') # VGG-16 is a pre-trained model used for image classification.
74     datagen = ImageDataGenerator(rescale=1. / 255)
75     generator = datagen.flow_from_directory( # Taking and resizing the Dataset as Needed
76         train_data_dir,
77         target_size=(img_width, img_height),
78         batch_size=batch_size,
79         class_mode=None,
80         shuffle=False)
81     print(len(generator filenames))
82     print(generator.class_indices)
83     print(len(generator.class_indices))
84
85     nb_train_samples = len(generator.filenames)
86     num_classes = len(generator.class_indices)
87
88     predict_size_train = int(math.ceil(nb_train_samples / batch_size))
89
90     bottleneck_features_train = model.predict(
91         generator, predict_size_train)
92
93     np.save('bottleneck_features_train.npy', bottleneck_features_train)
94
95     generator = datagen.flow_from_directory(
96         validation_data_dir,
97         target_size=(img_width, img_height),
98         batch_size=batch_size,
99         class_mode=None,
100         shuffle=False)
101
102     nb_validation_samples = len(generator.filenames)
103
104     predict_size_validation = int(
105         math.ceil(nb_validation_samples / batch_size))
106
107     bottleneck_features_validation = model.predict_generator(
108         generator, predict_size_validation)
109
110     np.save('bottleneck_features_validation.npy',
111            bottleneck_features_validation)
```

Figure 1: saving dataset features into bottleneck file



```

258
259 def train_top_model():
260     datagen_top = ImageDataGenerator(
261         rescale=1. / 255) # ImageDataGenerator class is used to expand the training dataset in order to improve the
262     # performance and ability of the model to generalize and original images consist in RGB coefficients in the
263     # 0-255, but such values would be too high for our model to process therefore we target values between 0 and 1
264     # instead by scaling with a 1/255.
265     generator_top = datagen_top.flow_from_directory(
266         train_data_dir,
267         target_size=(img_width, img_height),
268         batch_size=batch_size,
269         class_mode='categorical',
270         shuffle=False)
271
272     nb_train_samples = len(generator_top.file_names)
273     num_classes = len(generator_top.class_indices)
274
275     # save the class indices to use later in predictions
276     np.save('class_indices.npy', generator_top.class_indices)
277
278     # load the bottleneck features saved earlier
279     train_data = np.load('bottleneck_features_train.npy')
280
281     # get the class labels for the training data, in the original order
282     train_labels = generator_top.classes
283
284     # convert the training labels to categorical vectors
285     train_labels = to_categorical(train_labels, num_classes=num_classes)
286
287     generator_top = datagen_top.flow_from_directory(
288         validation_data_dir,
289         target_size=(img_width, img_height),
290         batch_size=batch_size,
291         class_mode=None,
292         shuffle=False)
293
294     nb_validation_samples = len(generator_top.file_names)
295
296     validation_data = np.load('bottleneck_features_validation.npy')
297
298     validation_labels = generator_top.classes
299     validation_labels = to_categorical(

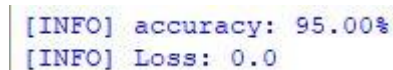
```

Figure 2: training dataset bottleneck file based on the sequential model method

After Creating and saving our dataset features into the bottleneck features file, the file is being taken and trained based on the sequential model method and creating our fully connected layer. Finally, the trained model is being saved and the Accuracy is being evaluated.

1.4.2 Final Stage

The data science component of the project uses a convolutional neural network model called VGG16. The VGG16 model has trained on 14 million images and used in large scale image recognition. The model has a 92.7% accuracy on the original 14million data set. In this project the model was retained using the data set we gathered, and it produce a 95% accuracy to identify yawning.



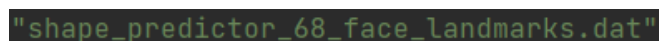
```

[INFO] accuracy: 95.00%
[INFO] Loss: 0.0

```

Figure 3: accuracy of the data science model

The video feed is captured using the cv2 library and the built-in camera. High-order random projection algorithm is used to detect the face from the frame. Then the 68-point facial landmarks were detected by using the following file.



```

"shape_predictor_68_face_landmarks.dat"

```

Figure 4: 68 facial landmarks file

Face detection and processing of the frames is run in an infinite while loop, so that the face detection is running until the user stop it. First the captured frame is resized, and gray scaled. Then the detector will detect the face from the frame and after that the eyes are detected. After detection the eye aspect ratio algorithm is used to determine the oneness of the eye. The eye aspect ratio is calculated the using 6 points placed in the eye lids and the two ends of the eye. If the eye aspect ratio is larger than 0.2 the user is detected as drowsy.

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Figure 5: ear formula

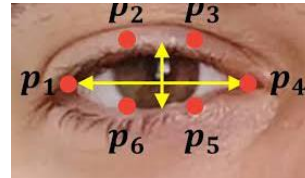


Figure 6: points on the eye

For the prediction of yawning the mouth is also extracted from the frames.

```
mouth = shape[48:68]
mouthHull = cv2.convexHull(mouth)
x = (mouthHull[0][0][0])
y = (mouthHull[1][0][1])
mouth = frame[(y - 20):y + 60, (x - 20):x + 60]
# mouth=frame[(y-20):y+20, (x-40):x+10]
im_gray = cv2.cvtColor(mouth, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(im_gray, cv2.HOUGH_GRADIENT, 1, 1, param1=50, param2=30, minRadius=0,
                           maxRadius=10)
```

Figure 7: capturing the mouth from the frames

Both the yawning and the head position us identified by the model and prediction is presented. Therefore, two pictures 1.jpg and 2.jpg is taken from the live camera feed and saved in the folder. First the model read these image files and process the image by resizing and normalizing. Then the processed image is inputted into the model.

```
image1 = load_img(image_path, target_size=(hSize, wSize))
image1 = img_to_array(image1)
image1 = image1 / 255
image1 = np.expand_dims(image1, axis=0)
# print("BBBBBB")
model = applications.VGG16(include_top=False, weights='vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5')
# print("BB")
bottleneck_prediction = model.predict(image1)
```

Figure 8: capturing the eyes from the frame

Then the prediction is calculated and the stored as 1 or 0 in the inId variable. Then the below if condition is used to choose between drowsy or not. (figure show the mouth features prediction)

```
if inID == 0:
    print(" Mouth close")

if inID == 1:
    print("Mouth open")
    voiceFlag = True
    yawnflag = 1
    lock.release()
```

Figure 9: if condition to determine the drowsiness

The above prediction will change the voiceflag to true if drowsiness is detected. This will cause the following if condition to be satisfied. In the voice function google text to speech is used to produce “Alert” sound at the case of drowsiness detection.

```
def voice():
    global voiceFlag
    global eyeFlag
    global yawnflag
    global headflag
    while True:
        if (voiceFlag == True):
            global count

            tts = gTTS(text="Alert", lang='en')
            tts.save(f'speech{count % 2}.mp3')
            mixer.init()
            mixer.music.load(f'speech{count % 2}.mp3')
            mixer.music.play()
            count += 1

            voiceFlag = False
```

Figure 10: voice function to sound the alert voice

1.5 Implementation of the backend component

In this project backend component consists of all APIs and logics and the database that frontend needs to call over the GUI(Graphical User Interface). Backend component has implemented using java language Springboot framework, mysql has used to create the database and manipulate data, that receive through datascience component and frontend

User_API class

```

@RestController
public class User_API {

    @Autowired
    private UserFactory userFactory;

    @GetMapping(value = "/")
    public ResponseEntity<?> test01(){
        return ResponseEntity.ok(new ResponseModel( message: "Backend Works!!!", HttpStatus.ACCEPTED));
    }

    @GetMapping(value = "/api/test") //to testing
    public ResponseEntity<?> test(){
        return ResponseEntity.ok(new ResponseModel( message: "Backend Works!!!", HttpStatus.ACCEPTED));
    }

    @GetMapping(value="/api/get-statistics") //to get statistics from database
    public ResponseEntity<?> userStatistics(@RequestParam("id") long id){
        return ResponseEntity.ok(userFactory.getStat(id));
    }
}

```

Figure 11: user api class image1

```

@GetMapping(value="/api/get-statistics") //to get statistics from database
public ResponseEntity<?> userStatistics(@RequestParam("id") long id){
    return ResponseEntity.ok(userFactory.getStat(id));
}

@GetMapping(value="/api/device-id") //to check user entered device id mapping with database
public ResponseEntity<?> deviceId(@RequestParam("id") String deviceid){
    return userFactory.getDeviceId(deviceid);
}

@PostMapping("/api/contact")//to get user feedback and save in database
public void contact(@RequestBody ContactForm contactForm) { userFactory.addMessage(contactForm); }

@RequestMapping(value = "/me", method = RequestMethod.GET)
@ResponseBody
public ResponseEntity<?> currentUserSimple(HttpServletRequest request) {
    Principal principal = request.getUserPrincipal();
    return ResponseEntity.ok(userFactory.getUser(principal.getName()));
}

@PostMapping("/api/assign-device") //to assign device id to the relevant user
public ResponseEntity<?> assignDevice(@RequestParam("userId") long userId,
                                     @RequestParam("deviceId") String deviceId){
    return userFactory.assignDevice(userId, deviceId);
}
}

```

Figure 12: use api class image2

Backend contains all API's for that needs to make the connection between frontend, backend and database.

Each API will call over the relevant GUI function in frontend.

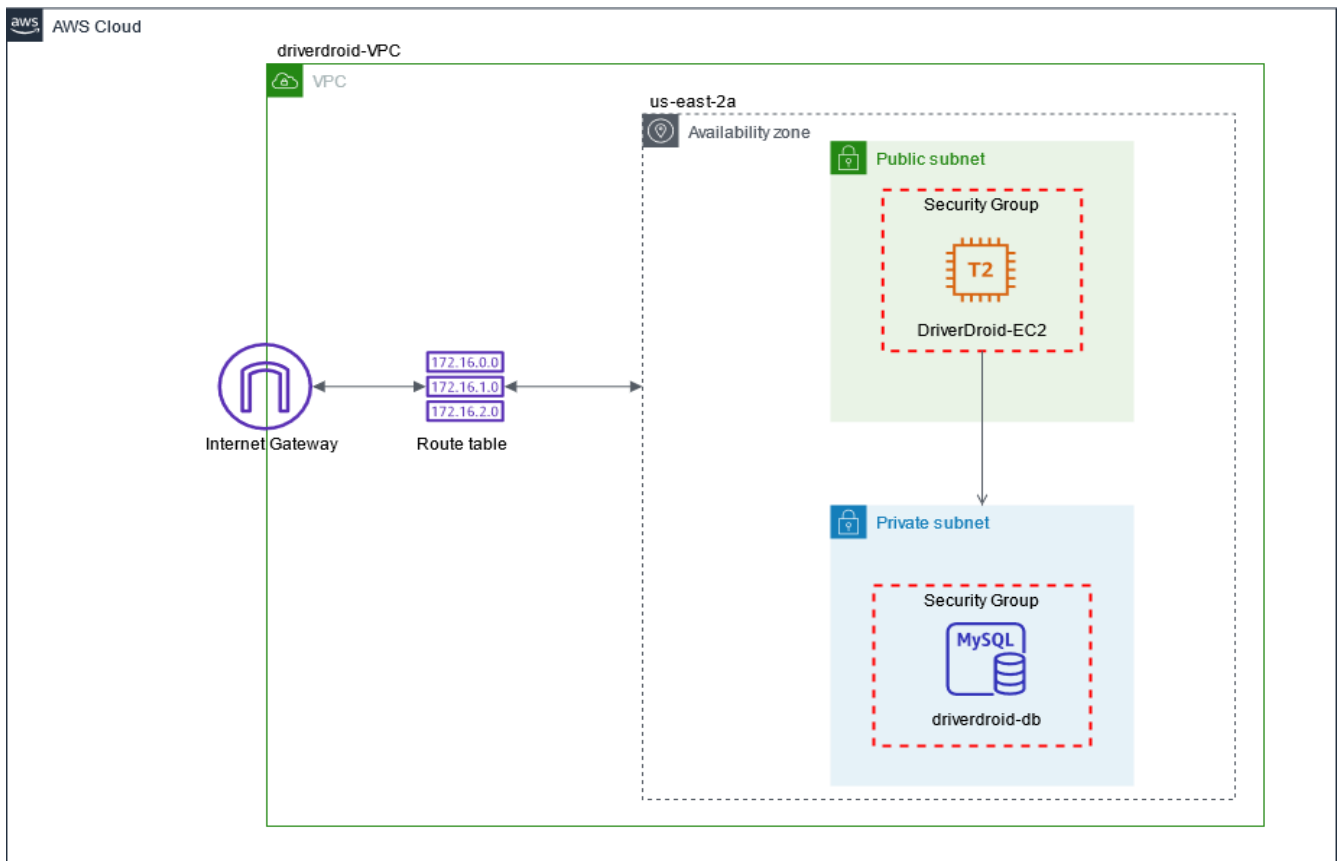


Figure 13: backend network

As in the above diagram have deployed the backend in AWS.

The diagram clearly shows the structure within AWS which shows whole application is within a VPC (Virtual Private Cloud)

A VPC is a logically isolated virtual network that we define in order to make our application more secure and at one place.

VPC contains subnets which is within a AZ (Availability Zone).

An AZ is a collection of 1 or more data centers within an AWS region.

I have deployed my Backend in a t2.micro type EC2 (Elastic Compute Cloud) instance, named as DriverDroid-EC2, within a public subnet, within a security group which acts as a firewall which filters traffic in and out of that EC2 instance.

Resources in public subnets can be accessed publicly.

Database is hosted in a MySQL database with the help of AWS RDS (Relational Database Service) and is also within a security group but on a private subnet which can be only accessed by the above DriverDroid-EC2 instance.

Since the Resources on the VPC are private and cannot be accessed over the internet we attach an Internet Gateway which allows traffic in and out from internet into the resources within the VPC.

A route table is used to route traffic in from the internet gateway to the resources and also vice-versa.

This model above is much efficient and secure since it uses private/public key access to access the resources and the database cannot be accessed elsewhere, other than only by the DriverDroid-EC2 instance.

1.6 Implementation of the mobile application component

The proposed 'Driverdroid' driver drowsiness detection system includes a mobile application which users can use along with the IoT device. Driverdroid mobile application consists of two major features which are, suggestion of the nearby rest places and the visualization of the drowsiness warnings history of the user.

React Native open-source mobile application framework was selected as the framework to use in the development of the 'Driverdroid' mobile application. Development of the mobile application was done according to the wireframe designs sketched at the design phase of the system and further improvements of the UIs were made to assure the high usability of the application.

When implementing the mobile application, developers took necessary actions to assure the maintainability and the scalability of the code. Therefore, the structure of the mobile application project was organized in the following manner.

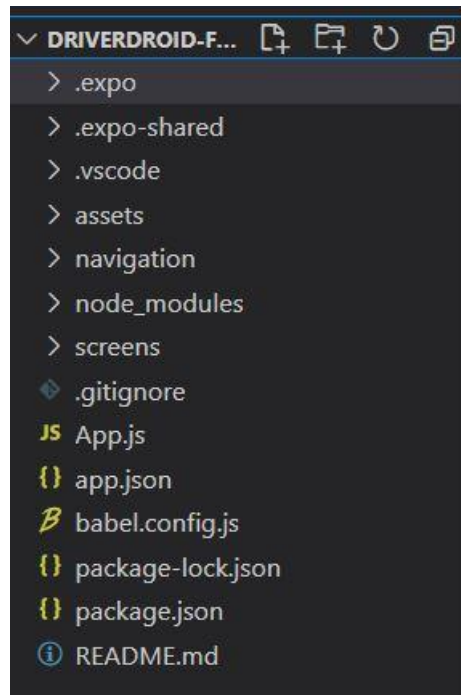


Figure 14: project structure of the mobile application

1. assets folder – contains all the asset files (images, fonts etc.)
2. navigation folder – contains all the JavaScript files related to app navigation
3. screens folder – contains all the JavaScript files related to screens that are present in the app

Screens that are implemented to suggest nearby rest places and to visualize the drowsiness warnings history of the user have been explained below.

1.6.1 Implemented Rest Places screen

In rest places screen, the app will get the current location of the user and it will show nearby rest places according to the user's location. When user selected a specific restaurant and click on 'get directions' button, the directions to the selected location will be provided with the help of 'Google Maps' app.

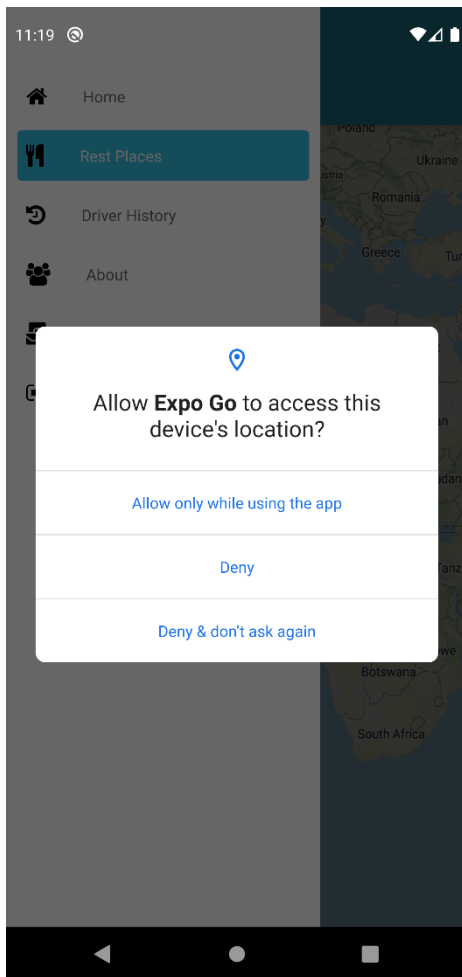


Figure 15: request location access image

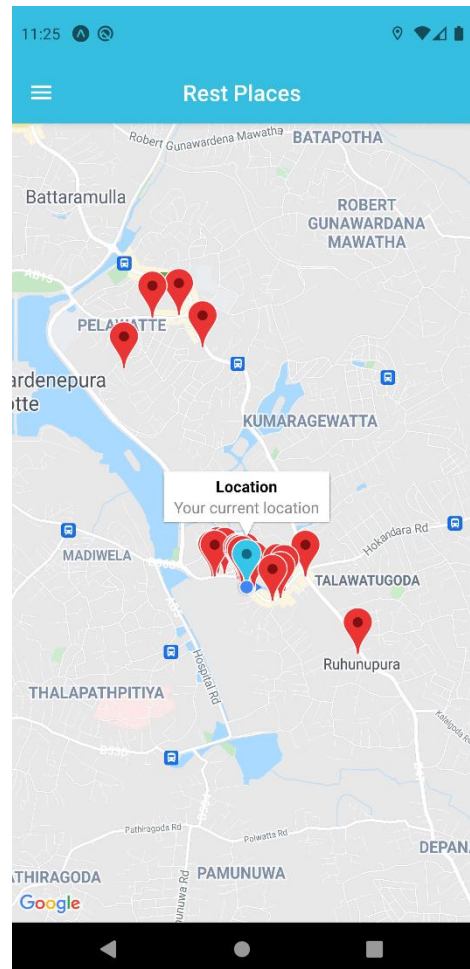


Figure 16: nearby rest places image



Figure 18: get directions to rest place image

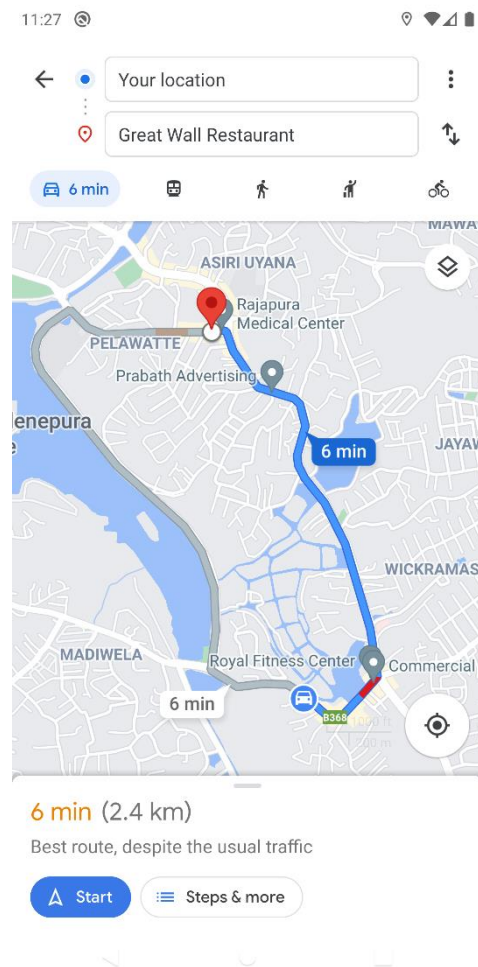


Figure 17: start getting directions image

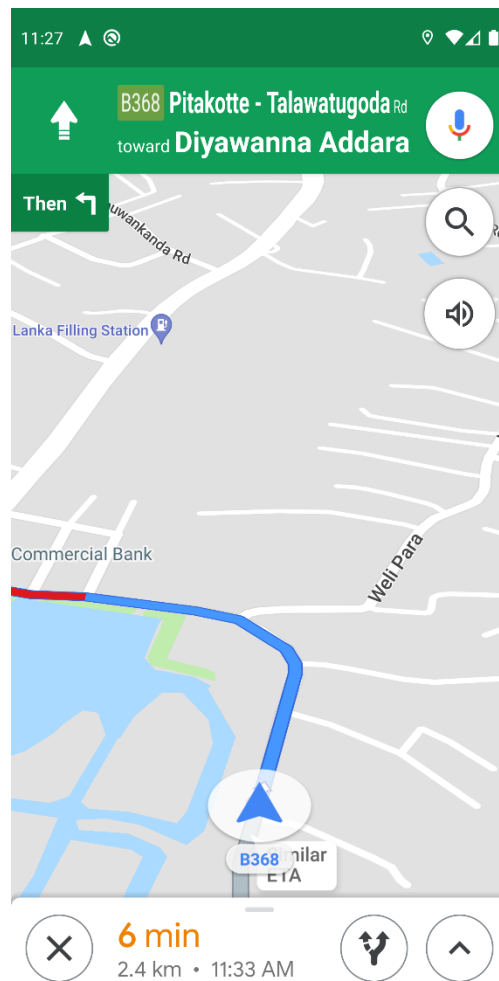


Figure 19: getting directions image

```
useFocusEffect(
  React.useCallback(() => {
    (async () => {
      let { status } = await Location.requestPermissionsAsync(); //request location permission from the user
      if (status !== 'granted') {
        Alert.alert(
          'Permission to access location was denied',
          'We want your current location to suggest you rest places. Would you like to give us permission to access your location?',
          [
            {
              text: 'Yes',
              onPress: () => { navigation.navigate('Rest Places') }
            },
            {
              text: 'No',
              onPress: () => { navigation.navigate('Home') }
            }
          ],
          { cancelable: false },
        );
      }
    })();
    return;
  })
);
```

Figure 20: code to request location permission from user

```

let location = await Location.getCurrentPositionAsync({}); //get current location of the user
setLatitude(location.coords.latitude);
setLongitude(location.coords.longitude);
setRegion({ latitude: location.coords.latitude, longitude: location.coords.longitude, latitudeDelta: 0.0922, longitudeDelta: 0.0421 });

//get nearby restaurants using google places API
function getNearbyPlaces() {
  if (latitude != null && longitude != null) {
    const url = 'https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=' + latitude + ',' + longitude
      + '&radius=2000&type=restaurant&key=AizaSyByamnP1uIhr_56Qn94KuInBvctHRgMA';
    fetch(url)
      .then((response) => {
        return response.json()
      })
      .then((Jsonresponse) => {
        setPlaces(Jsonresponse.results);
      })
      .catch((error) => {
        console.log(error);
        alert('An error has occurred while trying to find nearby rest places!')
      });
  }
}
getNearbyPlaces();
})();
}, [latitude, longitude])
);

```

Figure 22: code to get nearby rest places

```

65
66 //get directions to the selected rest place with the help of 'google maps' app
67 function getDirections(destinationLoc) {
68   const url = `https://www.google.com/maps/dir/?api=1&travelmode=driving&dir_action=navigate&destination=${destinationLoc}`
69   Linking.canOpenURL(url).then(supported => {
70     if (!supported) {
71       console.log('An error occurred while trying to get directions');
72     } else {
73       return Linking.openURL(url);
74     }
75   }).catch((error) => {
76     console.error(error)
77   });
78 }
79

```

Figure 21: code to get directions

1.6.2 Implemented Driver History screen

In driver history screen, user will be presented with the details of his latest drowsiness warning, available within a month. The screen provides a ‘find out more’ button, which when clicked navigate the user to driver history statistics screen.

If user has not received any drowsiness warnings during past month the app will show that he/she has no history of drowsiness warnings for past month.

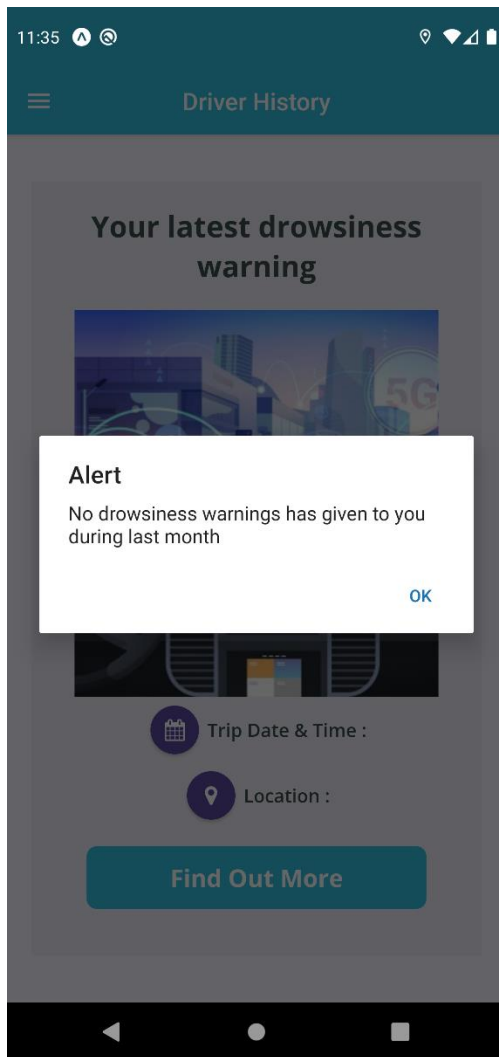


Figure 24: no warnings history image

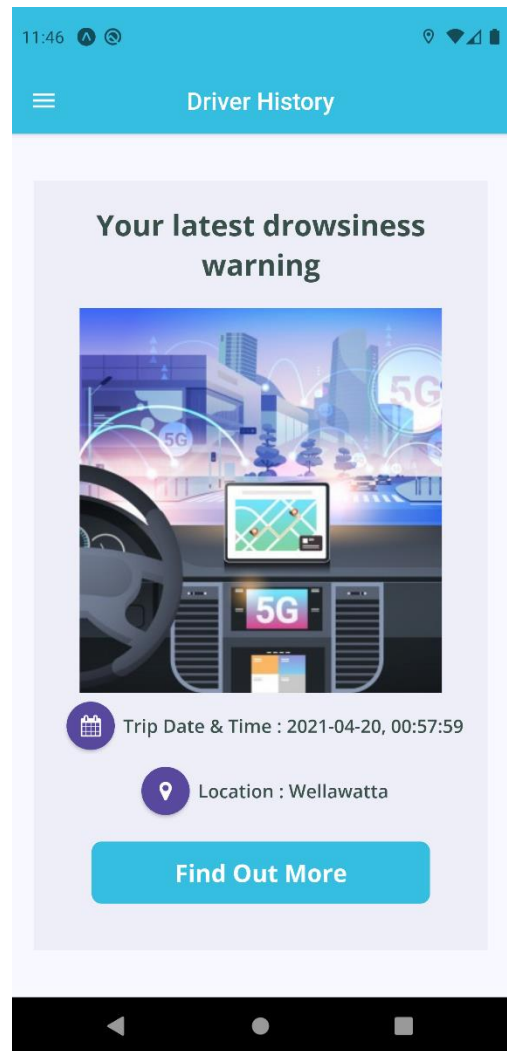


Figure 23: latest drowsiness warning image

If user tries to choose 'find out more' option without having any drowsiness warnings within past month, the app will not let user to navigate to driver history statistics screen.

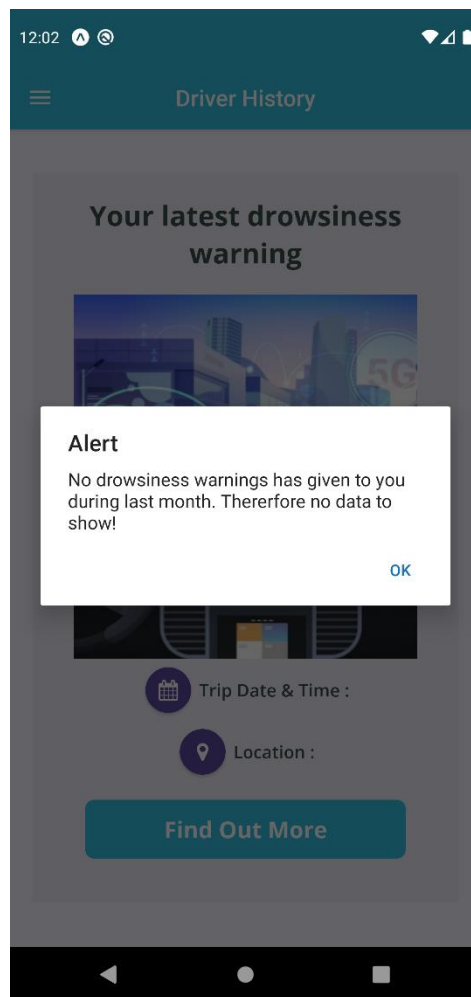


Figure 25: no drowsiness warnings list image

```
//get drowsiness warnings history details of the user from the server
function getHistoryData() {
  const url = `http://18.221.60.193/api/get-statistics?id=${userData.id}`;
  fetch(url, {
    method: 'GET',
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json',
      'Authorization': 'Bearer ' + userToken.token,
    }
  })
  .then((response) => {
    const statusCode = response.status;
    const data = response.json();
    return Promise.all([statusCode, data]);
  })
  .then(([status, Jsonresponse]) => {
    if (status !== 200) {
      alert('Error occurred while trying to get history data. Please try again later');
    } else if (Jsonresponse.length !== 0) {
      const sortedResponse = Jsonresponse.sort(function (a, b) { //sort history data according to the warned date
        return new Date(b.dateTime) - new Date(a.dateTime);
      });
      setHistoryData(sortedResponse);
      let lastTrip = sortedResponse[0]; //get latest drowsiness warning details of the user
      let lastTripDate = lastTrip.dateTime.substring(0, 10); //extract last warning date
      let lastTripTime = lastTrip.dateTime.substring(11, 19); //extract last warning time
      let lastTripLocation = lastTrip.location; //extract last warning location
      setLastTripDate(lastTripDate);
      setLastTripTime(lastTripTime);
      setLastTripLocation(lastTripLocation);
    } else if (Jsonresponse.length === 0) {
      alert('No drowsiness warnings has given to you during last month')
    }
  })
  .catch((error) => {
```

Figure 26: code to get warnings history data

1.6.3 Implemented Driver History Statistics screen

In this screen, user will be presented with his/ her drowsiness warnings history within last month.

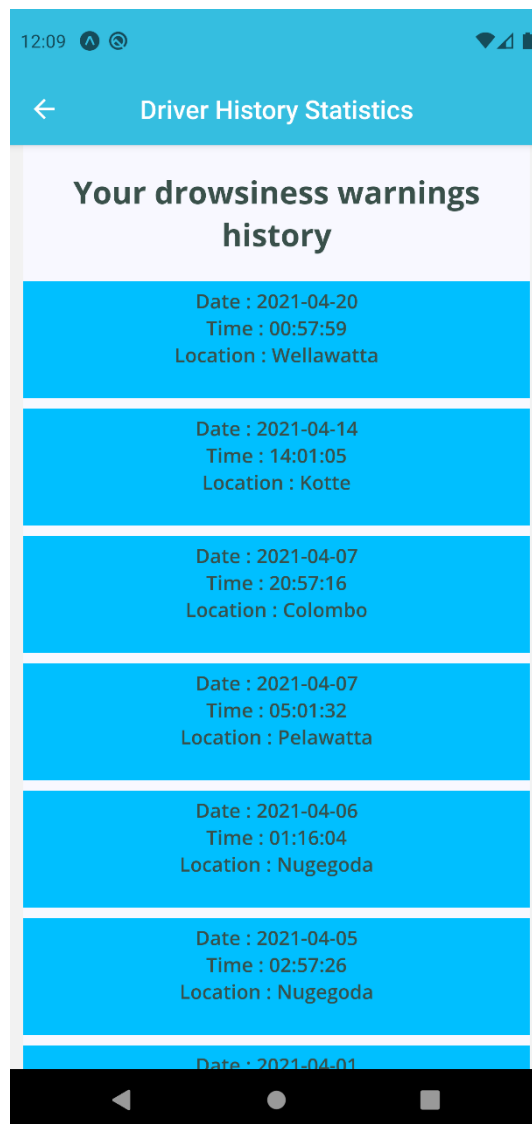


Figure 28: driver warnings history image

```
useFocusEffect(
  React.useCallback(() => {
    (async () => {
      const data = route.params.historyData; //get driver warnings history details passed with navigation routes
      let dataList = [];
      for (var i = 0; i < data.length; i++) { //extract relevant details from the retrieved history data
        let dataItem = {};
        dataItem['date'] = data[i].dateTime.substring(0, 10);
        dataItem['time'] = data[i].dateTime.substring(11, 19);
        dataItem['location'] = data[i].location;
        dataList.push(dataItem);
      }
      setHistoryDataList(dataList);
    })();
  }, [route])
);
```

Figure 27: code that extract drowsiness warnings history details from the navigation routes

```

<View>
  {(historyDataList.length != 0) &&
    <FlatList
      data={historyDataList} //populate flat list with user warnings history data
      ListHeaderComponent={FlatListHeader}
      keyExtractor={({item, index}) => index.toString()}
      renderItem={({ item }) => (
        <Text style={styles.listItem}>Date : {item.date}{"\n"}Time : {item.time}{"\n"}Location : {item.location}{"\n"}</Text>
      )}
    )}
  />
}
</View>
</View>

```

Figure 29: code to populate list with warnings data

All the details of the rest of the implemented screens in the mobile application can be found under *Appendix section A.1*

1.7 Deployments/ CI-CD Pipeline

A CI/CD pipeline is a series of steps that capable of take source code all the way into production. Depending on the requirements in differnet organisations, there could be multiple pipelines required to achieve the final goals.

CI (Continous Intergration) signifies all of the steps needed to automate the build and packaging of software.

CD (Continuous Deployment) reflects all of the assuarance, infrastructure, and deployment steps to deploy artifacts into production successfully.



Figure 30: CICD pipeline

There are few steps a CI/CD pipeline consist of,

Build - The stage where the application is compiled.

Test - The stage where code is tested. Automation here can save both time and effort.

Release - The stage where the application is delivered to the repository.

Deploy - In this stage code is deployed to production.

A CI/CD pipeline can be triggered by some significant event, as a pull request from a source code repository (a code change).

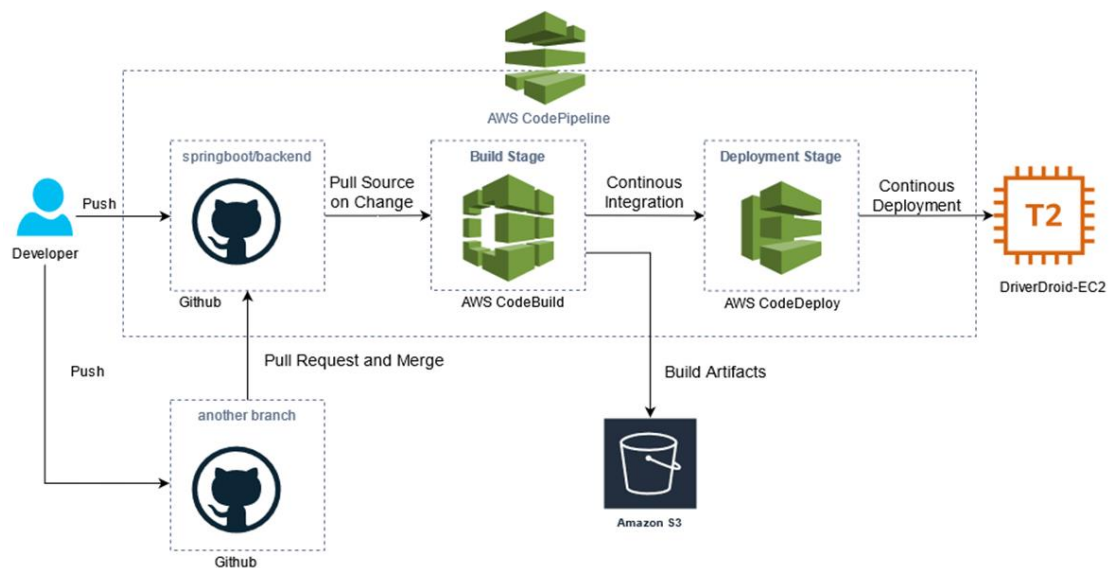


Figure 31: AWS CI/CD pipeline

According to this project, as seen in the diagram, in backend component AWS code pipeline consist of three stages which are git hub, AWS build stage and deployment stage.

When developer push the code into github, and to a particular branch and when Pull request or merge the code, it will trigger the AWS CodePipeline and from that AWS build stage start to automate the code and if there are changes of the code, check for the changes of the code and build artifacts (jar files) in the build stage. This process is called as Continuous Integration (CI) and also created artifacts (jar files) will store in Amazon S3(simple storage service), then it will move to deployment stage which is called as CD (Continuous Deployment) which contents deployment process to deploy artifacts into the relevant server. In this project it will deploy to the AWS EC2(Elastic Compute Cloud) server. CodeDeploy and codebuild within AWS code pipeline is used to maintain CI/CD and deployed in EC2(Elastic Compute cloud).

1.8 Chapter Summary

This chapter discussed the implementation of the prototype and the technology selection for the project. The core development language was python, and the front end was developed using java. The chapter discuss about the used libraries and frameworks. The chapter also explains the implementation code in detail. This includes the core data science front end GUI and the backend. These explanations are accompanied by the code snippets. Also, the continuous integration and continuous delivery was implemented in the above chapter.

Chapter 02 – Testing

2.1 Chapter Overview

The previous chapter discussed the Implementation phase of Driverdroid. This chapter will focus on the testing phase of Driverdroid. The Testing of the functional and non-functional requirements of Driverdroid will be discussed to ensure that it meets all the required standards. This chapter will provide a detailed description about the limitations faced during the testing process with an evaluation of the testing results.

2.2 Testing Criteria

The Testing for Driverdroid was done according to the IEEE 829 test plan Template, where the Functional and Non-Functional testing are being tested. To make sure that Driverdroid performs all the required core Functional and Non-Functional Requirements several testcases were identified and carried out in order to make sure that Driverdroid works as expected. These test cases will define the conditions whether the Driverdroid system functions as required.

2.3 Testing functional requirements

The testing for the functional requirements was carried out using the black box testing approach to verify whether the requirements of our prototype worked as expected. Several test cases were done to test these functional requirements.

Test case no.	Feature tested	Test case description	Test case condition	Expected result	Actual result	Status
1.0	FR 1	Monitor any Facial Drowsiness State	Test Driver drowsy state by lowering head posture	Alarm rings alerting the driver that he is drowsy	Alarm rings alerting the driver that he is drowsy	Pass (100%)
2.0	FR 2	Monitor the Eye Posture of the Driver	Test Driver Drowsy state when his eyes are closing	Alarm rings alerting the driver that he is drowsy	Alarm rings alerting the driver that he is drowsy	Pass (100%)
3.0	FR 3	Sound an alarm whenever a drowsiness state is detected.	Test whether all the drowsy states are identified from the	Alarm rings alerting the driver that he is drowsy	Alarm rings alerting the driver that he is drowsy	Pass (100%)

			Driver			
4.0	FR 4	Suggest Nearby resting places to rest	Test whether the Mobile App show nearby Resting places for the driver to rest after alerting the driver's drowsiness state.	Mobile App show nearby Resting places for the driver	Mobile App show nearby Resting places for the driver	Pass (100%)
5.0	FR 5	Show the Shortest Route for the driver to his Final destination	Test whether the Mobile App show the shortest route for the driver to reach his destination safely and in a short time.	Mobile App show the shortest route for the driver to reach his destination safely and in a short time.	Mobile App show the shortest route for the driver to reach his destination safely and in a short time.	Pass (100%)
6.0	FR 6	Monitor Whether the driver is drowsy Or tired through Yawning.	Test Driver Drowsy state through Yawning	Alarm rings alerting the driver that he is drowsy	Alarm rings alerting the driver that he is drowsy	Pass (100%)

Table 1: functional requirements testing

2.4 Testing non-functional requirements

The Testing for the Non-functional requirements of the system was carried out, which includes Accuracy, performance, reliability, usability, and scalability.

Several test cases were done to test these Non-functional requirements.

Test case no.	Feature tested	Test case description	Test case condition	Expected result	Actual result	Status
---------------	----------------	-----------------------	---------------------	-----------------	---------------	--------

1.0	NFR 1	Performance	The IoT system should run at a reasonable speed without slowing down in a way that could harm the user experience	The IoT system runs at a reasonable speed without slowing down.	The IoT system runs at a reasonable speed without slowing down.	Pass (100%)
2.0	NFR 2	Security	The IoT system must always maintain the confidentiality of data obtained from the user	The users' details saved securely through JWT user authentication and SSH(Secure shell) to access server securely.	The users' details saved securely through JWT user authentication and SSH(Secure shell) to access server securely.	Pass (100%)
3.0	NFR 3	Reliability	The IoT system should perform reliably without crashing and should be reliable to the drivers to use	The IoT system detects drowsiness without crashing once turned On until the user off the device	The IoT system detects drowsiness without crashing once turned On until the user off the device	Pass (100%)
4.0	NFR 4	Accuracy	The IoT system should alert the driver if and only if the driver is drowsy. It should not alarm the driver if he is not showing any symptoms of drowsiness	The Prototype should alarm and alert the Driver when he is at any of the Drowsiness State.	The Prototype should alarm and alert the Driver when he is at any of the Drowsiness State.	Pass (100%)
5.0	NFR 5	Extensibility	The IoT system should maintain the scalability and it should be	The IoT system can be upgraded with any new	The IoT system can be upgraded with any	Pass (100%)

			easy to maintain in future	features in the future.	new features in the future.	
6.0	NFR 6	Usability	The system should provide user friendly and easy to understand interfaces when interacting with the user	The User should be given a Screen to start and stop detecting the drowsiness. The User should be able to View the Drowsiness times from the Mobile App.	The User is given a Screen to start and stop detecting the drowsiness. The User is able to View the Drowsiness times from the Mobile App.	Pass (70%)

Table 2: non-functional requirements testing

2.5 Unit testing

Unit tests were done during development and the training stage and the bugs encountered were identified during the testing stage and were fixed. Unit testing was performed to enhance the quality of the prototype and to fix all the bugs and run the test cases. Below are two examples where unit testing was performed to check if the Accuracy of the Prototype was up to the standard.

```
[INFO] accuracy: 90.00%
[INFO] Loss: 0.0
```

Figure 32: test result 1

```
[INFO] accuracy: 95.00%
[INFO] Loss: 0.0
```

Figure 33: test result 2

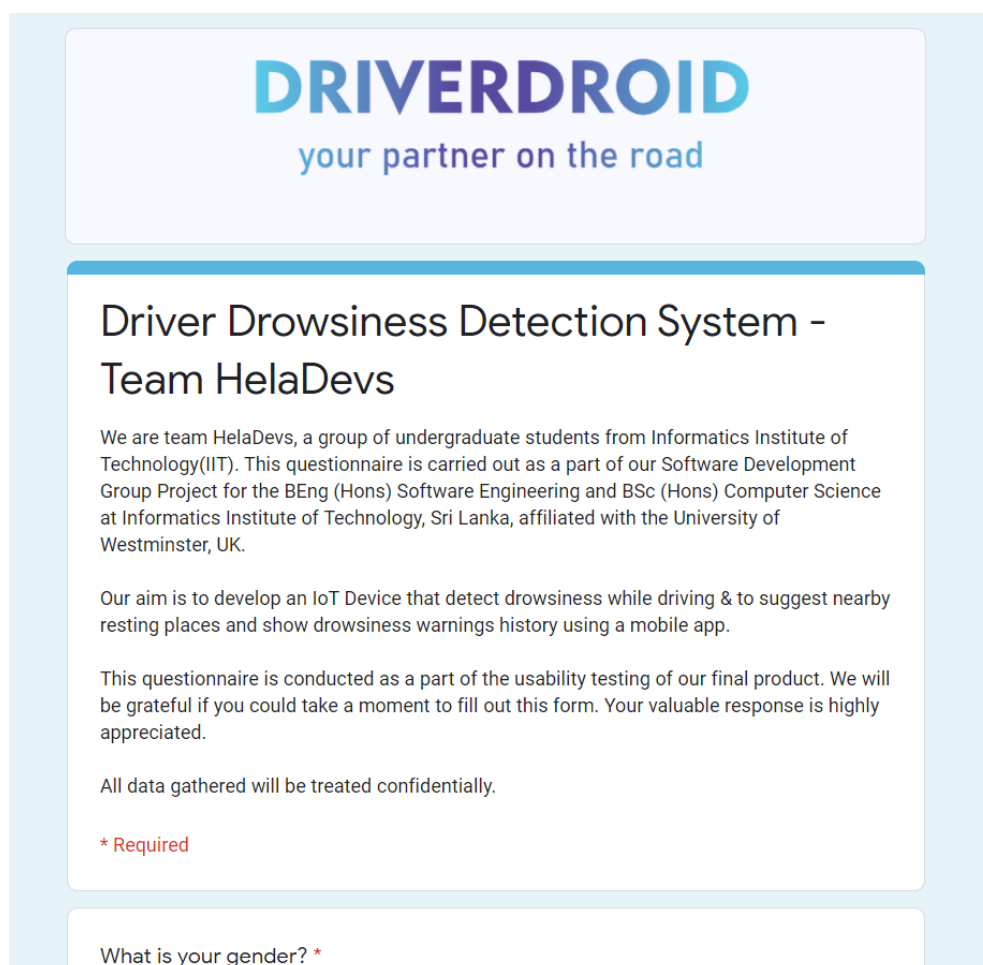
2.6 Performance testing

The machine learning model to detect drowsiness was run in windows-based machine and also the raspberry pi module. The model ran without any crashes and the drowsiness was detected with acceptable responsiveness.

2.7 Usability testing

The aim of the usability testing is to test and to get feedback about the product from the end users. This is a very important phase of testing as it let developers to directly get feedback from their actual users.

In Driverdroid, developers have ensured that both IoT device and the mobile application maintain a good user experience. To get further confirmation about the usability of the Driverdroid, the developers distributed a questionnaire and received responses has analyzed below.



The image shows a digital form titled "DRIVERDROID your partner on the road". Below the title, the form is for the "Driver Drowsiness Detection System - Team HelaDevs". It contains several paragraphs of text explaining the project and the purpose of the questionnaire. At the bottom, there is a question "What is your gender? *" with a red asterisk indicating it is required.

DRIVERDROID
your partner on the road

Driver Drowsiness Detection System - Team HelaDevs

We are team HelaDevs, a group of undergraduate students from Informatics Institute of Technology(IIT). This questionnaire is carried out as a part of our Software Development Group Project for the BEng (Hons) Software Engineering and BSc (Hons) Computer Science at Informatics Institute of Technology, Sri Lanka, affiliated with the University of Westminster, UK.

Our aim is to develop an IoT Device that detect drowsiness while driving & to suggest nearby resting places and show drowsiness warnings history using a mobile app.

This questionnaire is conducted as a part of the usability testing of our final product. We will be grateful if you could take a moment to fill out this form. Your valuable response is highly appreciated.

All data gathered will be treated confidentially.

* Required

What is your gender? *

Figure 34: usability testing form image1

What is your gender? *

☐ Male

☐ Female

☐ Prefer not to say

Choose your age group? *

☐ 18 - 30

☐ 31 - 40

☐ 41 - 50

☐ 51 - 60

☐ Above 60

[Next](#)

Never submit passwords through Google Forms.

This form was created inside of Informatics Institute of Technology. [Report Abuse](#)

Google Forms

Figure 35: usability testing form image2

Driverdroid - IoT Device

Objective of this IoT device is to detect driver drowsiness by monitoring & analyzing the facial features, yawning and eye posture of the driver and to alert the driver, if he is detected as a drowsy driver.

Effectiveness of Driverdroid IoT Device Features *

	Needs Improvement	Good	Efficient	Very Efficient
Drowsiness Detection	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Alerting Driver	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Any Comments?

Your answer

BackNext

Never submit passwords through Google Forms.

This form was created inside of Informatics Institute of Technology. [Report Abuse](#)

Google Forms

Figure 36: usability testing form image3

Driverdroid - Mobile Application

Objective of this mobile application is to show nearby rest places when driver detected as a drowsy driver & to show a history of drowsiness warnings that has given to driver by IoT device.

Link to access the mobile application - <https://expo.io/@gee99/projects/Driverdroid>

Effectiveness of Driverdroid Mobile Application Features *

	Needs Improvement	Good	Efficient	Very Efficient
Sign in, Sign up and Sign out process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Nearby rest place suggestion	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Providing directions to rest places	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Showing drowsiness warnings history	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sending feedback messages	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Sending feedback messages

☐

☐

☐

☐

Any Comments?

Your answer

Back

Submit

Never submit passwords through Google Forms.

This form was created inside of Informatics Institute of Technology. [Report Abuse](#)

Google Forms

Figure 37: usability testing form image4

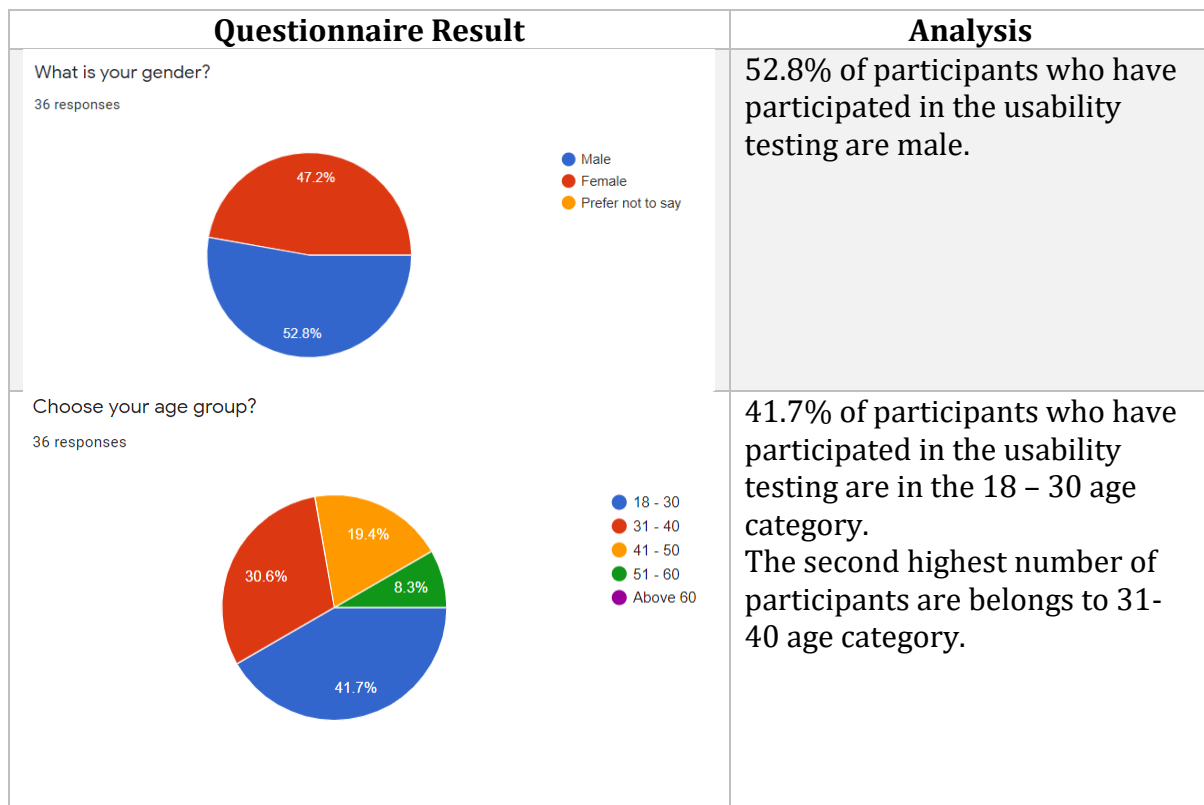


Figure 38: questionnaire result analysis image1

Driverdroid - IoT Device

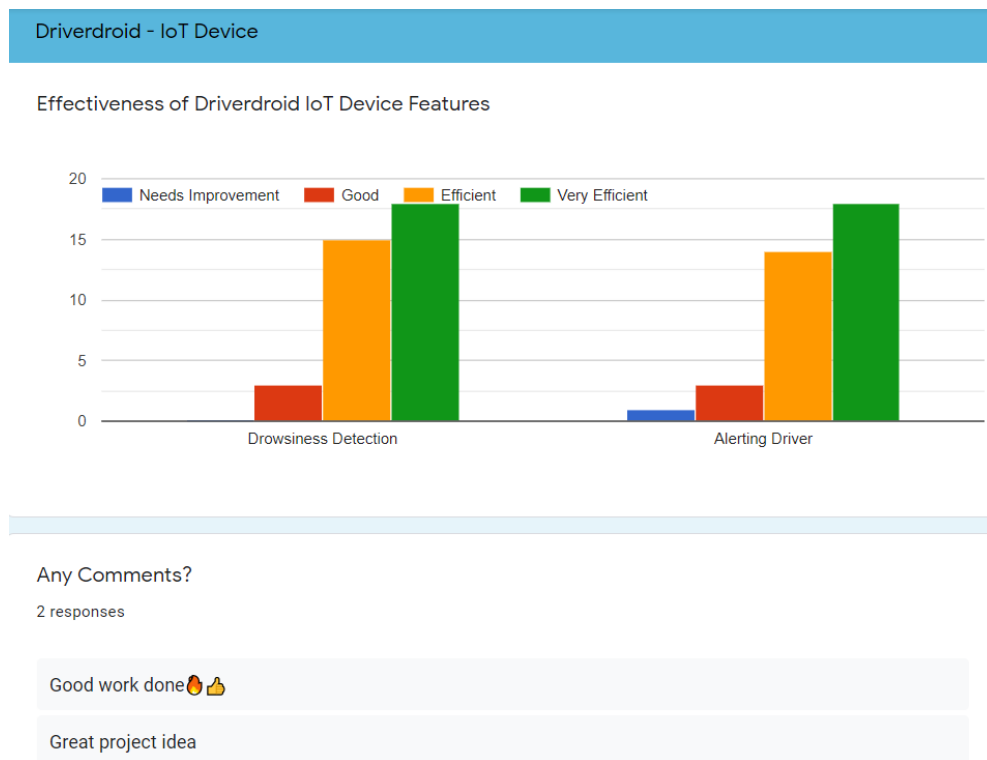


Figure 39: questionnaire result analysis image2

Majority of the responses received regarding the IoT device were positive. For both major features of the IoT device which are drowsiness detection and alerting the driver have marked as very efficient by majority of the participants.

The retrieved feedback about the IoT device were also consisted positive feedbacks.

Driverdroid - Mobile Application

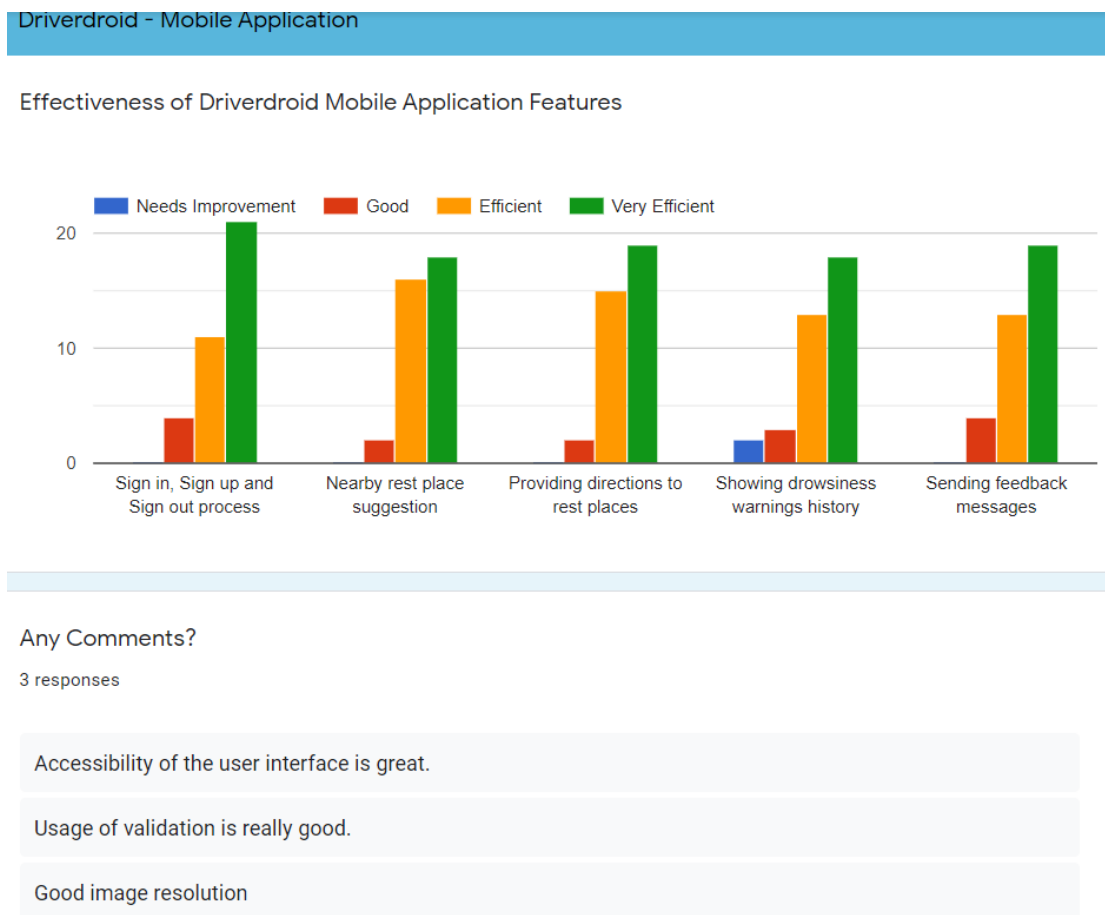


Figure 40: questionnaire result analysis image3

Majority of the responses received regarding the mobile application were also positive. For every major feature of the mobile application has marked as very efficient by majority of the participants.

The retrieved feedback about the mobile application were also consisted positive feedbacks.

2.8 Compatibility testing

Compatibility make sure that system application runs well across different type of mobile phones. The 'Driverdroid' mobile application was tested using mobile phones with minimum requirments such as Os- (Android 9.0 (Pie), EMUI 9.1), Chipset- (Mediatek MT6761 Helio

A22 (12 nm)), CPU- (Quad-core 2.0 GHz Cortex-A53), Internal-(16GB 2GB RAM, 32GB 2GB RAM eMMC 5.1).

All the functionalities worked well across all type of android phones even with minimum requirments.

2.9 Chapter Summary

This chapter discussed the testing phase details of our project Driverdroid along with goals and objectives of the system, testing the functional and non-functional requirements of Driverdroid and also the limitations faced during the testing process. The next chapter discusses the critical evaluations done for our system Driverdroid.

Chapter 03 – Evaluation

3.1 Chapter Overview

In previous chapter focused on the testing of the actual prototype. This chapter will be the evaluation of the architectural proposal made by 'HelaDevs'. Here will go through the Evaluation methods, quantitative evaluation such as feedback from users, domain experts and industry experts and also discuss on self evaluation.

3.2 Evaluation methods

During the Implementation process the prototype was evaluated to check if all the core functionalities were working as Expected. The Following methods were used to evaluate our Prototype.

- End user – The end users of the system (Driver's) was given a prototype and checked whether all the expected features are available in the system.
- Benchmarks and statistics – The Accuracy of the Machine learning Model was evaluated to check if the Accuracy of the model is up to a satisfiable Level.

3.3 Quantitative evaluation

Quantitative evaluation was performed to get results on the Accuracy of Trained model. We used a dataset of size 60GB that contain RGB videos of users driving and other times of drowsiness.

Using this dataset, we were able to train our Model to detect both yawning and any drowsy state of the user while Driving. Our Team was able to achieve an Accuracy 90% for the detection of Yawning and 95% Accuracy on the detection of any other state of drowsiness.

We also Collected the Accuracy of Our Prototype by getting feedbacks from drivers after using our System. The Accuracy and Efficiency of our Prototype from the Feedbacks received was up to the expectations and standards.

3.4 Qualitative evaluation

A questionnaire for qualitative evaluation was sent to the end users and the industry experts to collect feedback about the final prototype of the project. A video with a detailed explanation of all the functionalities and features of the application and the project specification documentation were included in a google drive folder and that folder was shared with the end users and the industry experts along with questionnaire. 10 people were selected to gather feedback to the questionnaire based on their expertise domain. Out of those 10 people, 3 people added their response about the project.

Screenshots of the shared questionnaire and feedback responses are listed in below.

DRIVERDROID
your partner on the road

Qualitative Evaluation for the Driverdroid - Mobile Application

We are team HelaDevs, a group of undergraduate students from Informatics Institute of Technology(IIT). This questionnaire is carried out as a part of our Software Development Group Project for the BEng (Hons) Software Engineering and BSc (Hons) Computer Science at Informatics Institute of Technology, Sri Lanka, affiliated with the University of Westminster, UK.

Our aim is to develop an IoT Device that detect drowsiness while driving & to suggest nearby resting places and show drowsiness warnings history using a mobile app.

This questionnaire is conducted as a part of the qualitative evaluation of our mobile application. We will be grateful if you could take a moment to fill out this form. Your valuable response is highly appreciated.

* Required

Email address *

Your email

Figure 41: qualitative evaluation form image1

The form is a vertical stack of five white input boxes with rounded corners, set against a light blue background. Each box contains a label followed by a red asterisk, a text input field, and a horizontal line for the answer. The labels are: 'Email address', 'Name', 'Designation', 'Organization', and 'Effectiveness of Driverdroid Mobile Application Features'. The last box has a rating scale below the input field with four options: 'Needs Improvement', 'Good', 'Effective', and 'Very Effective'.

Email address *

Your email

Name *

Your answer

Designation *

Your answer

Organization *

Your answer

Effectiveness of Driverdroid Mobile Application Features *

Needs Improvement Good Effective Very Effective

Figure 42: qualitative evaluation form image2

Effectiveness of Driverdroid Mobile Application Features *

	Needs Improvement	Good	Effective	Very Effective
Sign in, Sign up and Sign out process	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Nearby rest place suggestion	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Providing directions to rest places	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Showing drowsiness warnings history	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sending feedback messages	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Feedback about the mobile application/project *

Your answer _____

Figure 43: qualitative evaluation form image3

The form contains the following sections:

- Two rows of radio button options for "Showing drowsiness warnings history" and "Sending feedback messages", each with four empty circles.
- A text input field labeled "Feedback about the mobile application/project *".
- A text input field labeled "Your answer".
- A question: "Would you like to give permission to include your feedback in our implementation report? *".
- Two radio button options: "Yes" and "No".
- A blue "Submit" button.
- Footer text: "Never submit passwords through Google Forms." and "This form was created inside of Informatics Institute of Technology. [Report Abuse](#)".
- The Google Forms logo.

Figure 44: qualitative evaluation form image4

Analysis

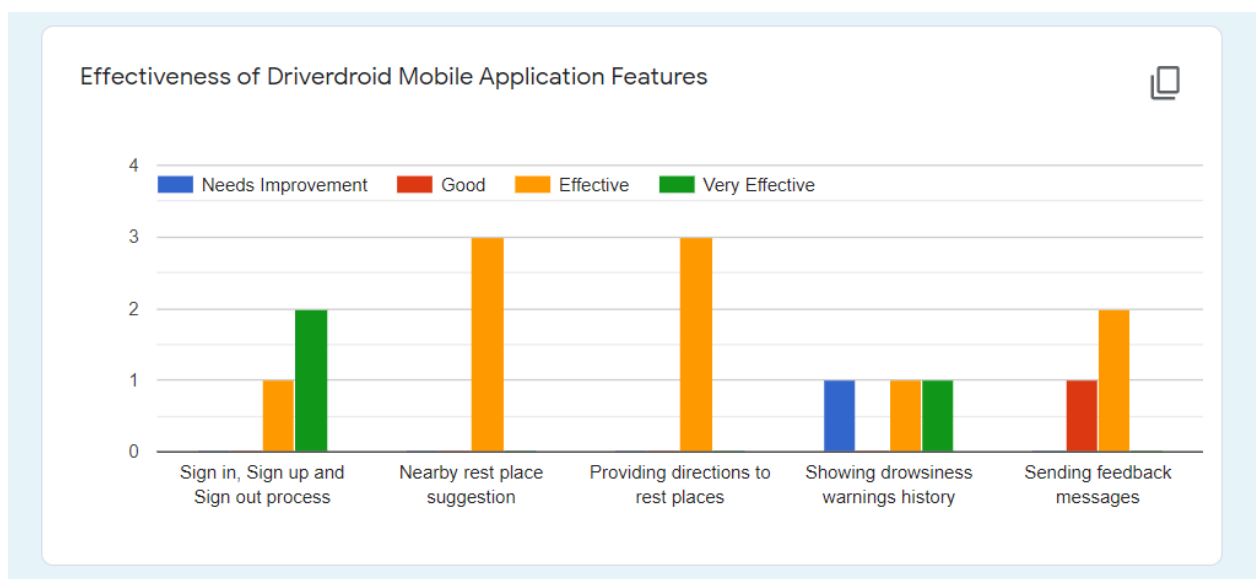


Figure 45: questionnaire result analysis image

- From the retrieved responses, majority has voted that Sign in, Sign Up and Sign Out process is very effective.

- From the retrieved responses, majority has voted that Nearby rest place suggestion is effective.
- From the retrieved responses, majority has voted that Providing directions to rest places is effective.
- From the retrieved responses, majority has voted that Showing drowsiness warnings history is very effective.
- From the retrieved responses, majority has voted that Sending feedback messages is very effective.

Feedback received from the evaluators are listed in the below table

Name and Designation	Yasas Sri Wickramasinghe Former Associate Technical Lead at 99x, Lecturer at University of Moratuwa
Feedback	“Very good project idea, looking forward to see this app getting released to general public.”
Name and Designation	Nimesh Ekanayaka Software Engineer at IFS Sri Lanka
Feedback	“This is very useful initiative. Try to improve this app continuously, hopefully many will use this.”
Name and Designation	Suresh Michael Peiris Chief Executive Officer at Inforwaves (Private) Limited
Feedback	“Team HelaDevs has done a great job while developing this system. The majority of road accidents happen due to driver drowsiness, sometimes even we fell asleep while travelling long distances. Therefore, this application motivates the driver to stay awake and get some rest from the nearby rest house.”

Figure 46: qualitative evaluation feedback

3.5 Self evaluation

‘Driverdroid’ was designed and developed with the aim of detecting drowsy drivers and alerting them about their drowsiness. Throughout the project, all the team members of ‘HelaDevs’ did their best to make sure that ‘Driverdroid’ is a success. The project domain was an entirely new aspect to each team member. With proper background research on the project domain, all developers gained the necessary knowledge that was needed to make ‘Driverdroid’ a reality.

Developers believe that ‘Driverdroid’ consisted of a satisfactory scope for a second-year group project. All the functional and non-functional requirements identified in the project were also successfully achieved. The implemented prototype works reliably, and the received testing results also prove that as well. Developers have assured the code of the ‘Driverdroid’ to be maintainable and scalable. Thus, any further enhancements for the project also can be done without any hassle.

Therefore, overall, developers of the ‘Driverdroid’ believe that they were able to successfully complete a project which will be very useful to address a very important real-world problem.

3.6 Chapter Summary

The details gained from both qualitative and quantitative evaluations were analysed in this chapter. A special demo video and an evaluation feedback surveys were created to gain evaluations. Drivers, domain experts and industry experts were used for evaluation and received positive feedback from evaluators used. Evaluations indicates that the scope of the evaluators were also satisfied with the technological stack used for the implementation.

Chapter 04 – Conclusion

4.1 Chapter Overview

In the previous chapter testing results were evaluated and studied. This will be the concluding chapter of the project. It will be included the summary of this IoT device, accomplishment and highlights of the project during the period of research and implementation. In this chapter we will discuss the achievements of aims and objectives, legal, social, ethical and professional issues team faced and resolved, and the range of the research and limitations of this project. Also, it discusses the future enhancement that can be done and extra works, research papers done on this project.

4.2 Achievements of aims and objectives

4.3 Legal, social, ethical and professional issues

From the beginning to the end of this entire Software Development Group Project, the team members of ‘HelaDevs’ worked hard to ensure that any legal, social, ethical or professional issues that might arise in the course of the progress of this project were successfully handled. Such legal, social, ethical and professional issues that might arise, were handled in the following manner.

4.3.1 Legal

When developing ‘Driverdroid’ IoT device along with the mobile application, high regard was given to data ethics and protection regulations. Publicly available ‘UTA Real-Life Drowsiness Dataset’ by The University of Texas at Arlington has been used as the dataset for this project. The dataset was not used for any other purpose outside of the project use, and obtaining the dataset was not subjected to any legal constraints.

Users' personal information was not collected through the questionnaires sent out during the requirement gathering stage of the project. The information collected was handled confidentially and the users' privacy was secured. Collected information was used solely for the purpose of enhancing performance and the productivity. The identities of those who completed the questionnaire were kept anonymous.

Throughout this project, the Microsoft Office package, CorelDraw, StarUML software's along with Canva online tool were used for the documentation purposes, while PyCharm, IntelliJ, Anaconda (V4.8.3), DataGrip, Android Studio – Android Virtual Device, Visual Studio Code, Postman software's, Expo CLI online tool and Expo Go mobile app have been used for the implementation. All those software's were properly licensed software's.

To accommodate this project, it was ensured that no copyright laws or intellectual property rights were violated. The sources of the information and materials used have been correctly credited, and appropriate acknowledgment has been provided.

4.3.2 Social

The implemented 'Driverdroid' driver drowsiness detection system is not intended to target any specific religion, race or culture, and the developers of the 'Driverdroid' have ensured that it does not conflict with any specific race, religion, or culture's values.

Furthermore, no idea or technique has used during the design and implementation phases of this project, which may result in a socially controversial situation.

The implemented system currently communicates using the English language, and it could negatively influence the people who do not understand English. As a possible future enhancement of the 'Driverdroid', multi-language support can be suggested which will result in more users being able to communicate with the system in his/ her native or desired language without any interruption.

4.3.3 Ethical

The team members of 'HelaDevs' made sure that this project was always carried out in accordance with ethical standards. As a step of following ethical standards, before collecting relevant data via the questionnaire, the researchers provide the community with an overview of the project and the intent of collecting the relevant data.

During the information gathering phase, the project team made sure not to collect any information that might expose people's identities. Moreover, the researchers worked tirelessly, to maintain the confidentiality of the information gathered, leaving no space for data to be used outside of the project.

Only 36 out of the 60 people who participated in the data collection phase of the dataset which has used in this project, have provided permission to publish their images in public, according to the guidelines given on the 'UTA Real-Life Drowsiness Dataset' web page. It is strictly prohibited to reveal the identities of any of the 60 people. Therefore, the researchers always took necessary actions to fully comply with the given guidelines.

4.3.4 Professional

The researchers always tried their best to maintain the professionalism throughout the entire project. The dataset and requirement gathering was gathered professionally. The people who completed the questionnaire and the assessment were notified in advance, so that they had

enough time to thoroughly plan their responses. They were also told, how the data would be used in this project, and permission was first obtained.

All the project's activities were completed in a professional manner by the team members. Professional project management technologies have been used such as 'Trello', and 'Microsoft Teams' has been used as the meeting platform to arrange team meetings.

4.4 Limitations of the research

4.4.1 Limitation of hardware

The used hardware is not the latest and not the best performing. Better performing hardware can be obtained with proper funding to improve the performance of the device.

4.4.2 Limitation of language

The model only represents the out but in English this can be improved to support multilanguage graphical user interface.

4.4.3 Cost of the prototype

The cost of the prototype limits the performance improvement of the prototype. If the product were to be produced in mass scale, we could exploit economies of scale to reduce the cost and improve the performance.

4.4.3 Limitation of the dataset

Due to the limited time allocation, the training of the machine learning model is limited, and the dataset is limited to relatively small diversity. The diversity of the data set can be improved by gathering more images of different people in different ages and different genders to improve the accuracy of the machine learning model.

4.5 Future enhancements

4.5.1 Improve the accuracy

The current machine learning model used in the project re-trained using a limited data set. But the accuracy of the model can be increased if the model were to be trained in a wide variety of data. This can be done over time and a more accurate model can be obtained.

4.5.2 Decrease the cost

The cost of the hardware can be reduced by considering other options. Also, the performance can be improved by considering other hardware. If we were to improve the project to larger scale production, we could exploit economies of scale to further decrease the product.

4.5.3 Increase the performance

The current hardware can be improved (eg. Camera) to have a higher resolution input and faster processing time. The model can be improved to capture and process other data (eg. Heart rate) to improve the drowsiness prediction.

4.5.4 Improve user experience

The graphical user interface can be improved to better suit the user to improve the experience and more data can be presented to the user.

4.6 Extra work

- HaXmas2021 inter university hackathon

Team 'HelaDevs' participated in the HaXmas2021 inter university hackathon.

4.7 Concluding remarks

The aim was this project to design, develop, test, and evaluate a system to detect the driver drowsiness and alert driver about his drowsiness or fatigue accurately. The team 'HelaDevs' confidently believe that proposed idea was achieved through the created IoT device successfully. The team successfully met the predefined proposed functional requirements of the project. Also, the testing chapter and the evaluation chapter show that performance of the IOT device is up to standard.

References

Driverdroid - Qualitative Evaluation – Google Drive (2021). Drive.google.com. Available from

<https://drive.google.com/drive/folders/1ABAJ6EoU0m0XEhefvAFO2b3WAYIvI1yL?usp=sharing> [Accessed 21 April 2021].

Driverdroid - Usability Testing – Google Drive (2021). Drive.google.com. Available from <https://forms.gle/fvZZybHGyA2Xn8N57>

Driverdroid - Implementation Video link -

https://drive.google.com/file/d/1JOqNr0QPSZNxAOqqNgWFYdoi_3xpTmad/view?usp=sharing

Sacolick, I. (2021). What is CI/CD? Continuous integration and continuous delivery explained. InfoWorld. Available from <https://www.infoworld.com/article/3271126/what-is-cicd-continuous-integration-and-continuous-delivery-explained.html> [Accessed 23 March 2021].

UTA Real-Life Drowsiness Dataset. 2019. UTA-RLDD. [online] Available at: <https://sites.google.com/view/utarldd/home> [Accessed 21 April 2021].

Bibliography

Amazon EC2 (2021). Amazon Web Services, Inc.. Available from <https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc> [Accessed 11 February 2021].

Building a Convolutional Neural Network (CNN) in Keras (2018). Medium. Available from <https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5> [Accessed 14 January 2021].

Cloud Architect Course | Cloud Architect Certification Training (2021). Simplilearn.com. Available from https://www.simplilearn.com/cloud-solutions-architect-masters-program-training?&utm_source=google&utm_medium=cpc&utm_term=aws&utm_content=1670407902-64731278836-324103636735&utm_device=c&utm_campaign=Search-TechCluster-Cloud-AWSNew-ROW-Main-AllDevice-adgroup-AWS-Exact&gclid=Cj0KCQjwvYSEBhDjARIsAJMn0lhghKGkj-JWDla3hznmcUjX9Hd4PJrB1OD10MmqVQeQ6CDvmFdCnQaAgIGEALw_wcB [Accessed 14 March 2021].

Google Maps Platform | Google Developers (2021). Google Developers. Available from <https://developers.google.com/maps> [Accessed 4 January 2020].

JPA Repositories (2021). Docs.spring.io. Available from <https://docs.spring.io/spring-data/jpa/docs/1.5.0.RELEASE/reference/html/jpa.repositories.html> [Accessed 30 February 2021].

JWT.IO | JSON Web Tokens (2021). Jwt.io. Available from <https://jwt.io/> [Accessed 24 March 2021].

Phillip Webb, S. (2021). Spring Boot Reference Documentation. Docs.spring.io. Available from <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/> [Accessed 16 December 2020].

Places API | Google Developers (2021). Google Developers. Available from <https://developers.google.com/maps/documentation/places/web-service/overview> [Accessed 5 January 2021].

React Native Documentation · React Native (2021). Reactnative.dev. Available from <https://reactnative.dev/docs/getting-started> [Accessed 5 November 2020].

React Native Tutorial for Beginners - Build a React Native App [2020] (2020). Youtube.com. Available from <https://www.youtube.com/watch?v=0-S5a0eXPoc> [Accessed 12 November 2020].

Spring Boot CRUD Operations - javatpoint (2021). www.javatpoint.com. Available from <https://www.javatpoint.com/spring-boot-crud-operations> [Accessed 23 February 2021].

TensorFlow Documentation | TensorFlow Core | Machine Learning for Beginners and Experts (2021). TensorFlow. Available from <https://www.tensorflow.org/overview> [Accessed 23 December 2020].

VPC with public and private subnets (NAT) - Amazon Virtual Private Cloud (2021). [Docs.aws.amazon.com](http://docs.aws.amazon.com). Available from https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Scenario2.html [Accessed 15 March 2021].

What is Secure Shell (SSH) and How Does it Work? (2020). SearchSecurity. Available from <https://searchsecurity.techtarget.com/definition/Secure-Shell> [Accessed 16 March 2021].

Appendix Section A - Implementation

Appendix Section A.1 – Implementation of the mobile application component

A.1.1 Implemented Sign up screen

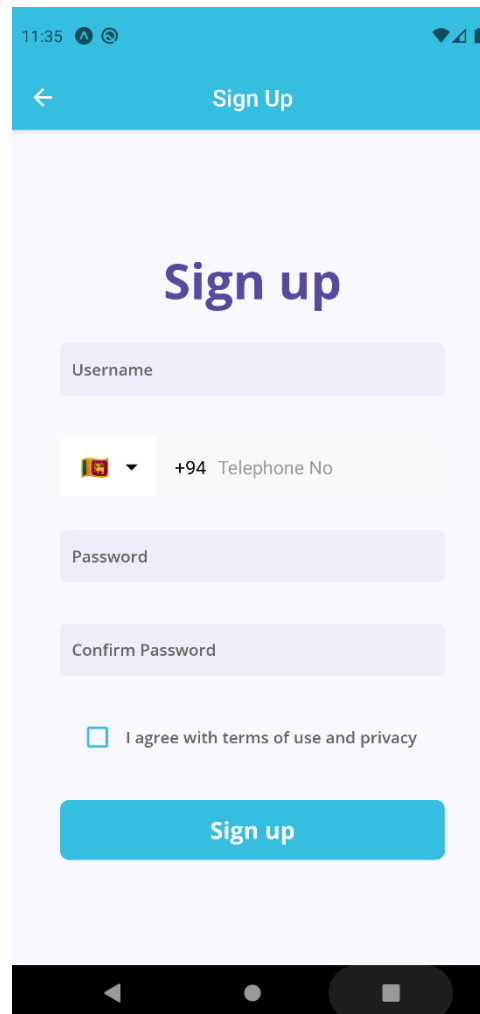
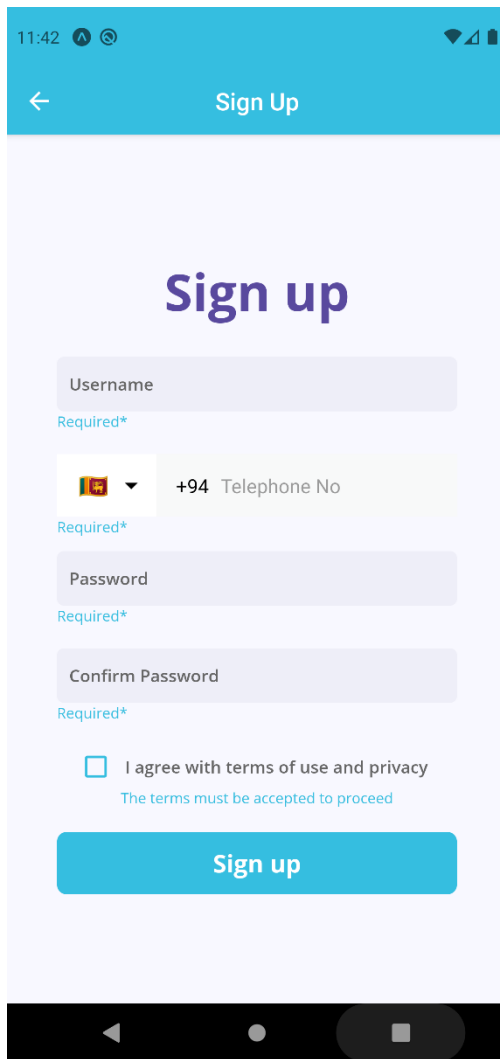
The screenshot shows a mobile application interface for signing up. At the top, there is a blue header bar with a back arrow on the left and the text "Sign Up" on the right. Below the header, the title "Sign up" is displayed in a large, bold, purple font. The form consists of several input fields: a light purple box for "Username", a light yellow box for a phone number (with a dropdown menu showing a flag and "+94 Telephone No"), a light purple box for "Password", and another light purple box for "Confirm Password". Below these fields is a checkbox labeled "I agree with terms of use and privacy". At the bottom of the form is a large blue button with the text "Sign up". The entire form is set against a light purple background. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

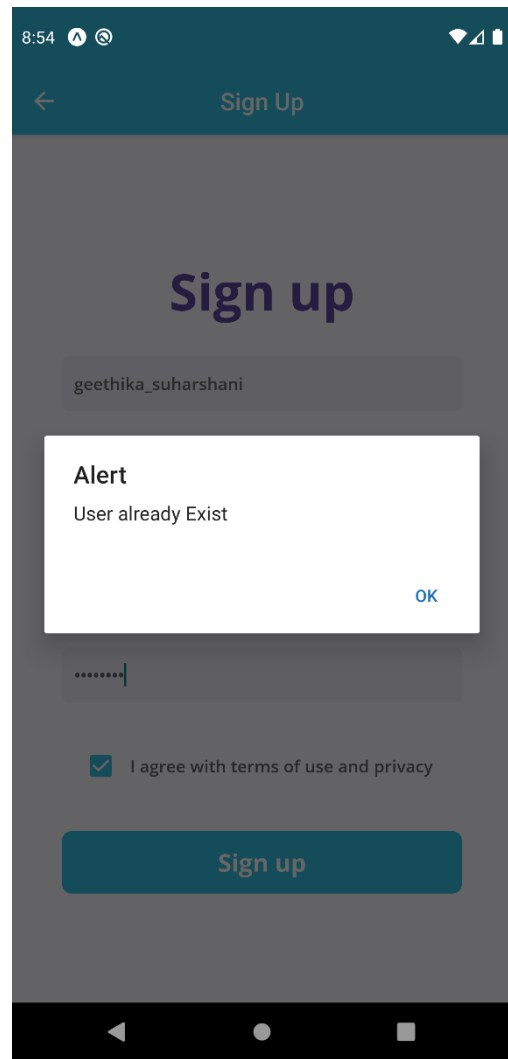
Figure 47: sign up screen

When using the app for the first-time, user will need to sign up using the sign up screen where he/ she will be asked to provide username, telephone number and password. User will also have to agree to the terms of use in order to proceed with the sign up process. Necessary validations for the user inputs were added to ensure only valid data will be send to the server.



The screenshot shows a mobile application's 'Sign Up' screen. At the top, there is a blue header bar with a back arrow and the text 'Sign Up'. The main content area has a light purple background. The title 'Sign up' is displayed in a large, bold, dark purple font. Below the title, there are four input fields, each with a 'Required*' label in blue text below it: 'Username' (light purple), '+94 Telephone No' (light yellow, with a dropdown menu showing a flag icon), 'Password' (light purple), and 'Confirm Password' (light purple). Below the password fields, there is a checkbox labeled 'I agree with terms of use and privacy' with the text 'The terms must be accepted to proceed' in blue below it. At the bottom, there is a large blue button with the text 'Sign up' in white. The status bar at the top shows the time 11:42 and various icons.

Figure 48: sign up screen input validation



The screenshot shows the same 'Sign Up' screen as Figure 48, but with a dark grey background. A white alert dialog box is overlaid in the center. The dialog has the title 'Alert' and the message 'User already Exist'. There is an 'OK' button in the bottom right corner of the dialog. The background form is dimmed, showing the 'Username' field with the text 'geethika_suharshani' and the 'I agree with terms of use and privacy' checkbox checked. The status bar at the top shows the time 8:54 and various icons.

Figure 49: user already exist alert

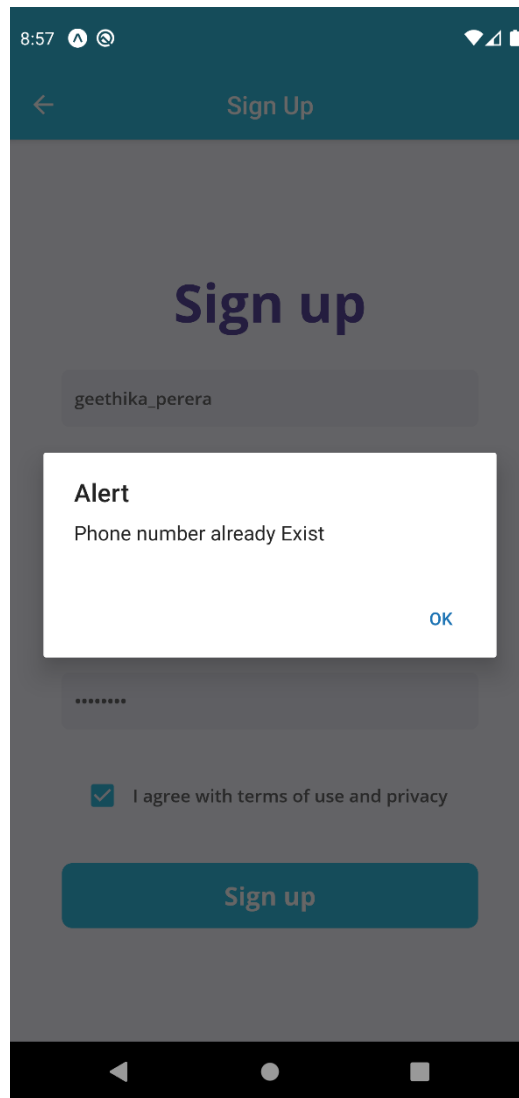


Figure 50: phone no. already exist alert

A.1.2 Implemented Verify Device screen

On successful provision of valid data in sign up screen, user will be directed to the verify device screen where he/ she will be asked to provide the device id mentioned in his/ her IoT device.

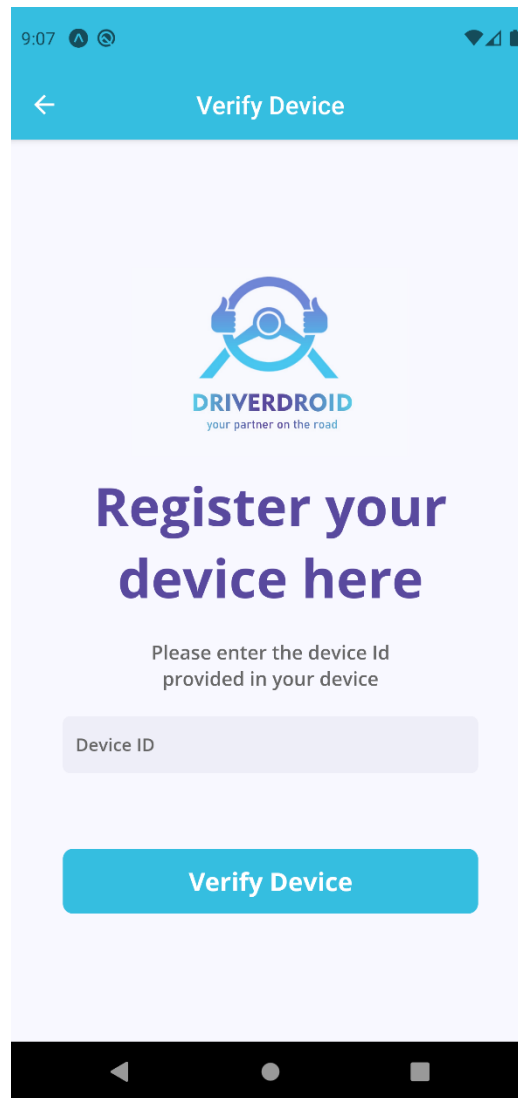


Figure 51: verify device screen

User inputs will be validated to ensure the provided device id is a valid device id and it is not registered with another user.

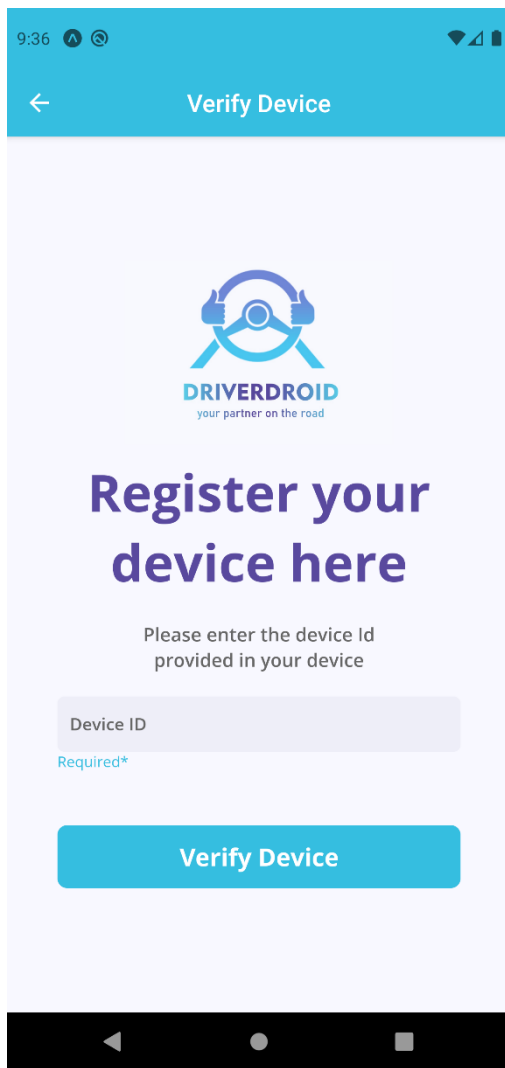


Figure 53: verify device validations

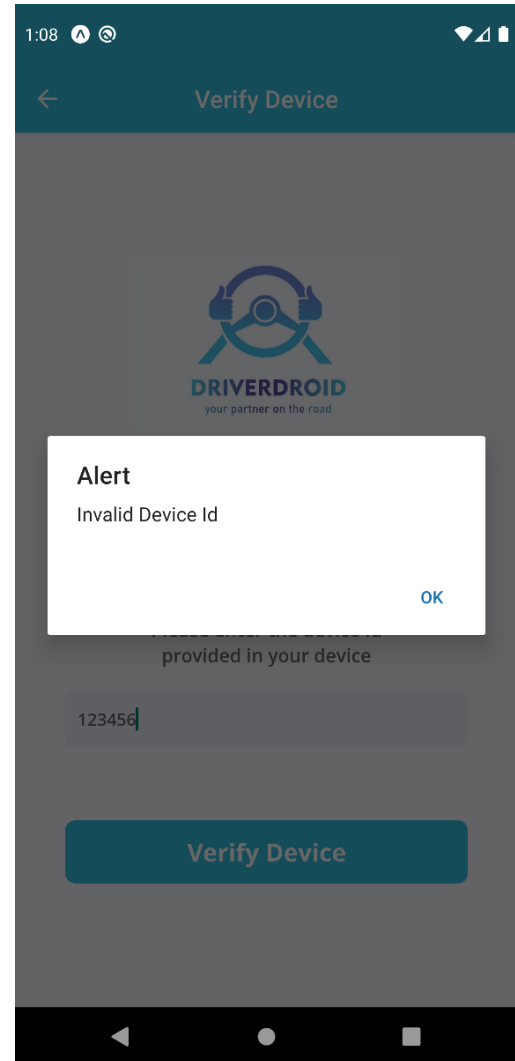


Figure 52: invalid device id alert

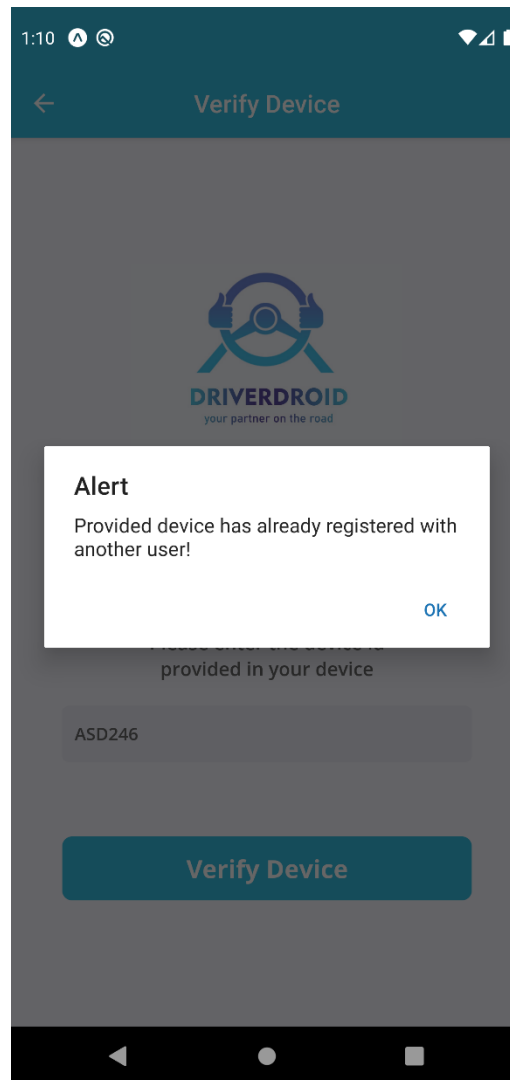


Figure 54: already registered device alert

A.1.3 Implemented Register Device screen

If user provided a valid device id in the verify device screen, he/ she will be directed to the register device screen where he/ she will be asked to provide the verification code that has sent to his/ her registered phone number.

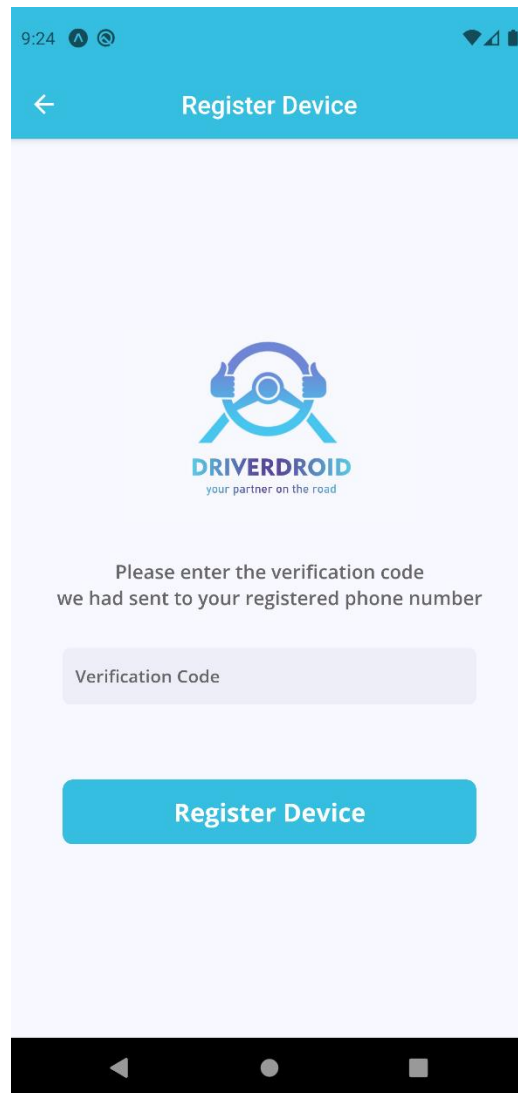


Figure 55: register device screen

The user input will be validated in this process also to ensure the provided verification code is valid and if it is valid the device will be registered with the user account.

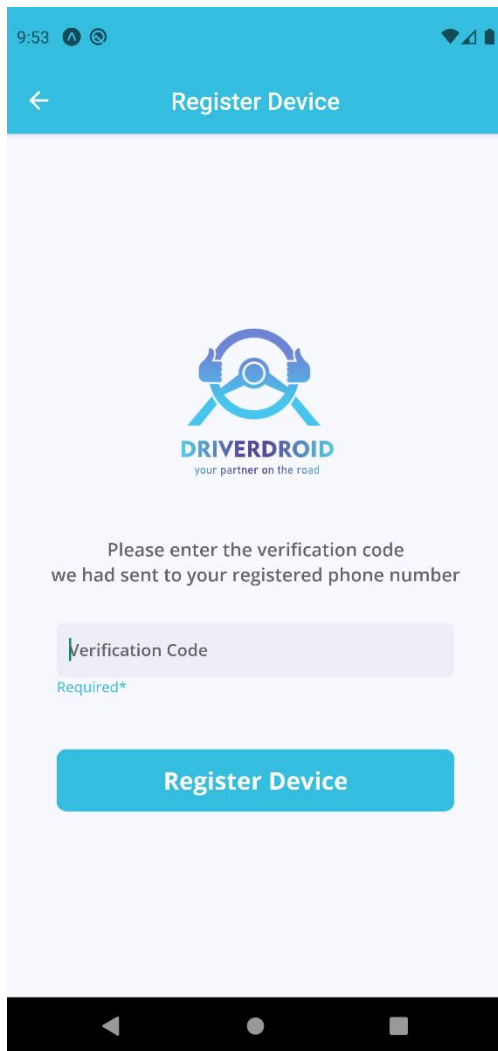


Figure 57: register device validations

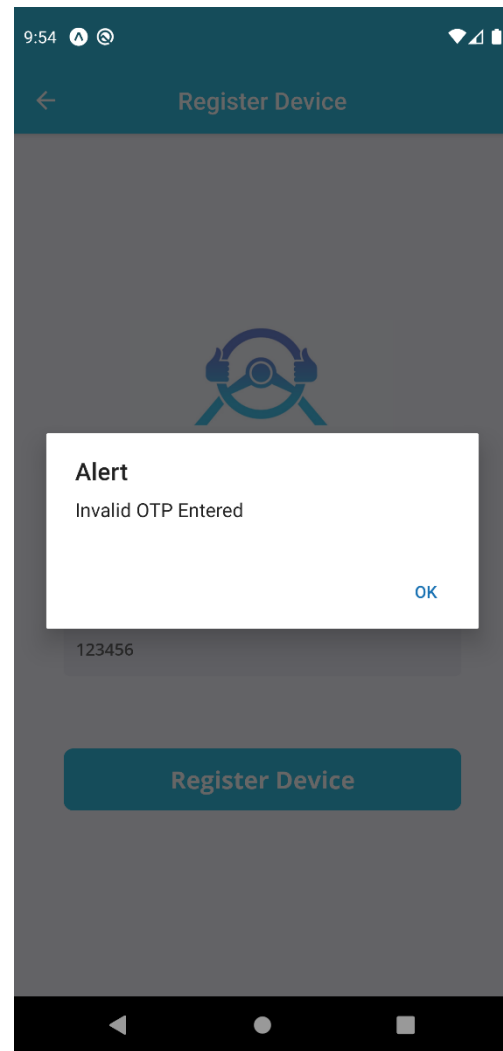


Figure 56: invalid otp alert

A.1.4 Implemented Sign In screen

On successful completion of sign up process, user will be directed to the sign in screen where he/ she will be asked to provide his/ her username and the password.

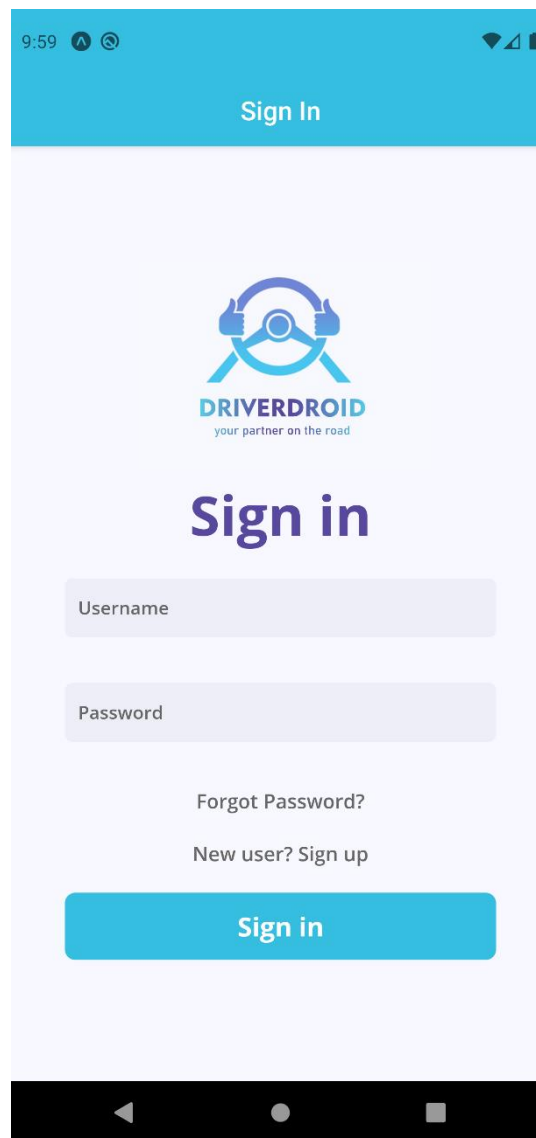


Figure 58: sign in screen

User input validation will be carried out in this process also.

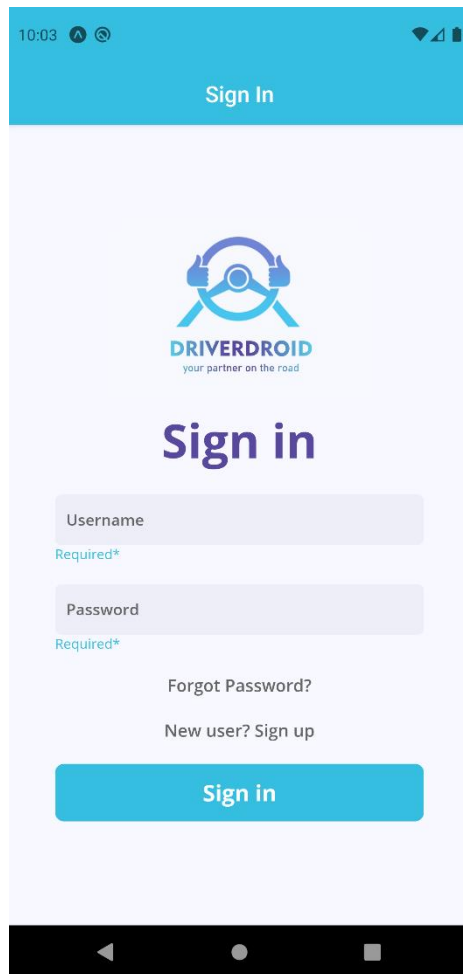


Figure 59: sign in screen validations

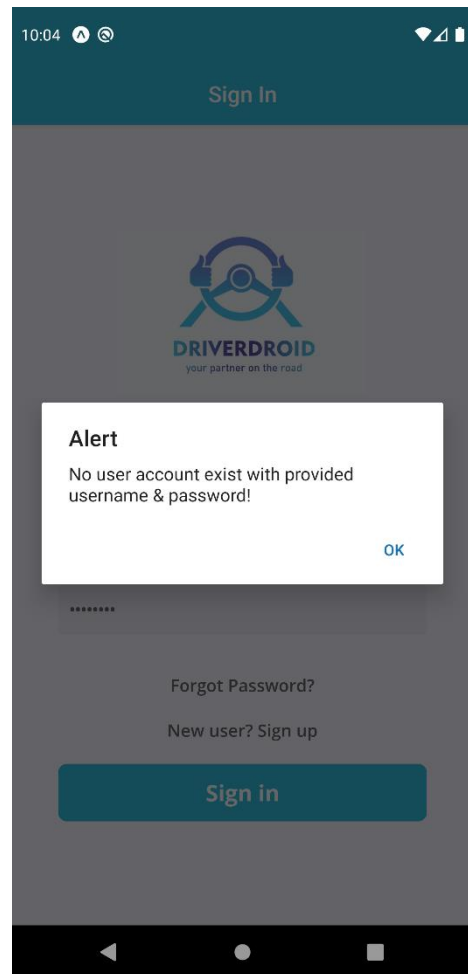


Figure 60: no such user account alert

A.1.5 Implemented Forgot Password screen

When signing in, if user want to reset his/ her password he/ she will be needed to choose forgot password option in the sign in screen which will direct him/ her to forgot password screen.

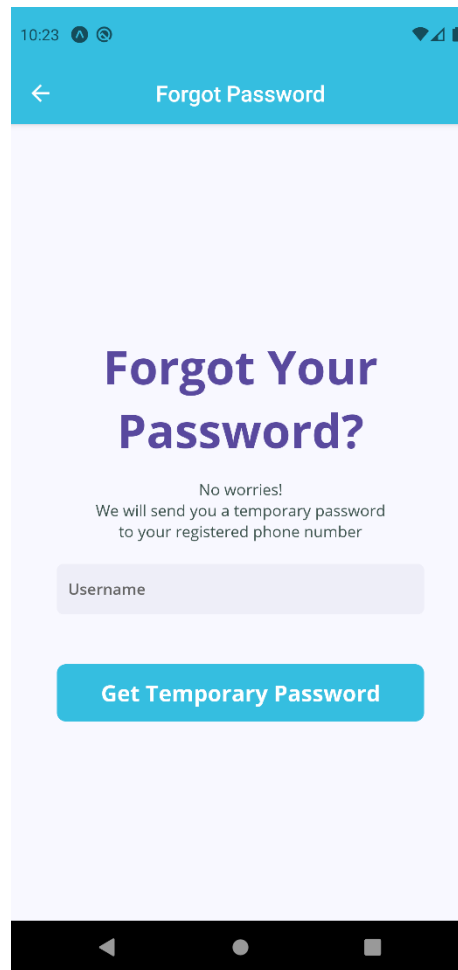


Figure 61: forgot password screen

When directed, user will be needed to provide his/ her username, which will be used to send a temporary password to the user's registered phone number.

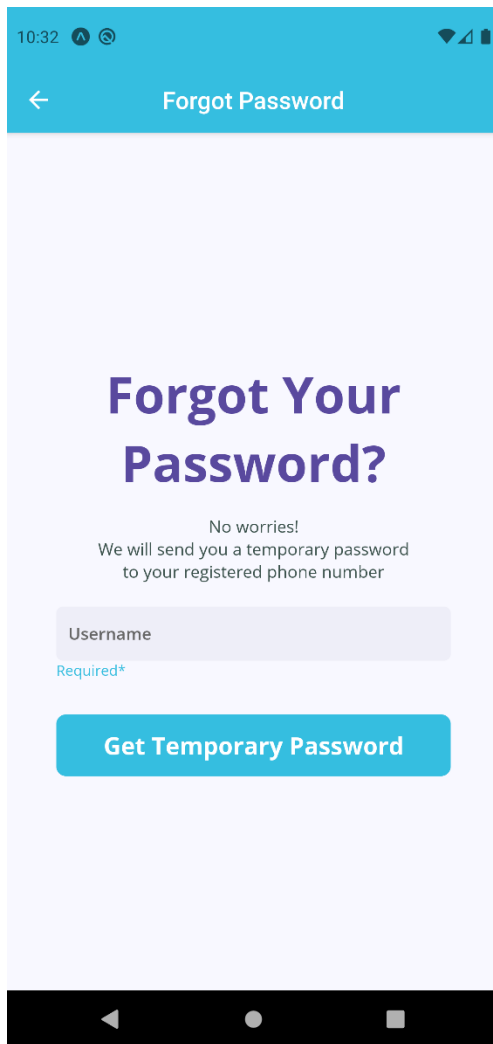


Figure 62: forgot password validations

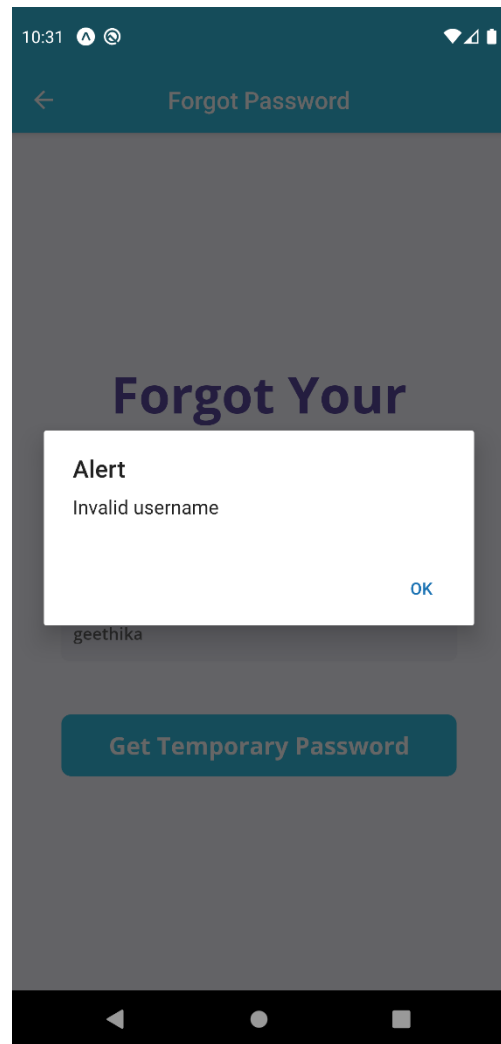


Figure 63: invalid username alert

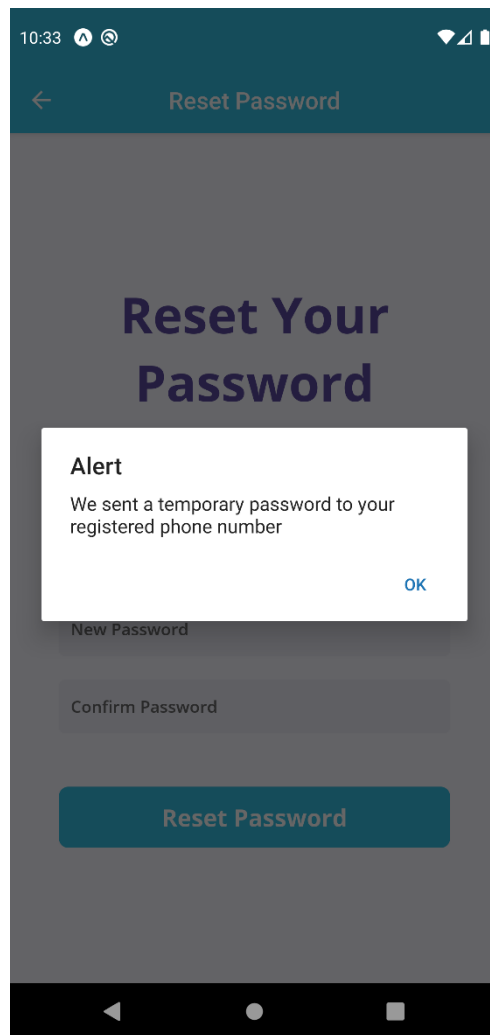
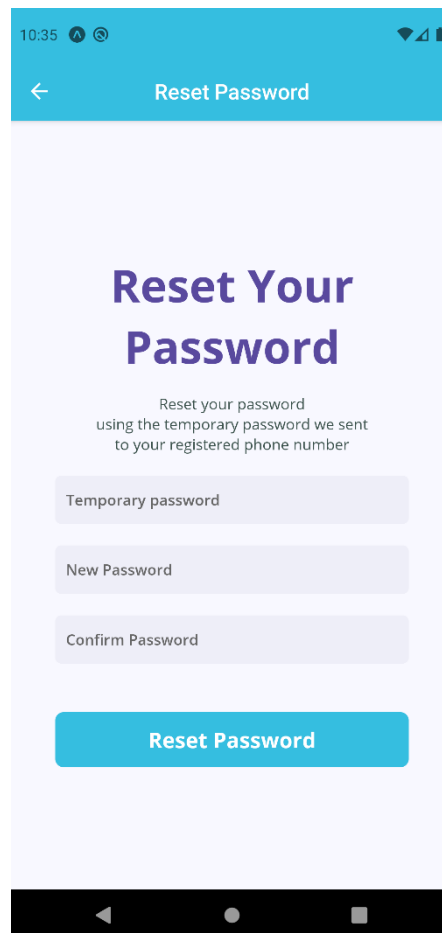


Figure 64: temp password sent alert

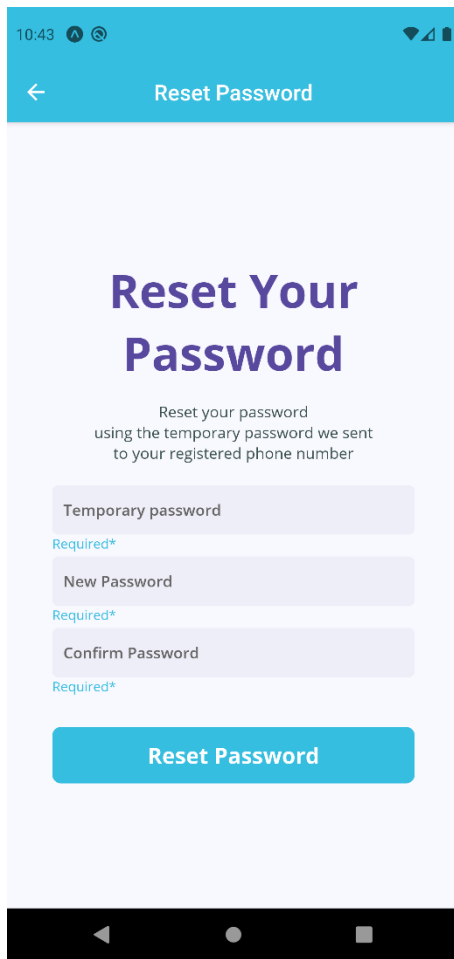
A.1.6 Implemented Reset Password screen

On successful provision of a valid username, a temporary password will be sent to user's registered phone number and the user will be directed to the reset password screen where he will be prompted to enter the provided temporary password along with the new password, he/she want to have.



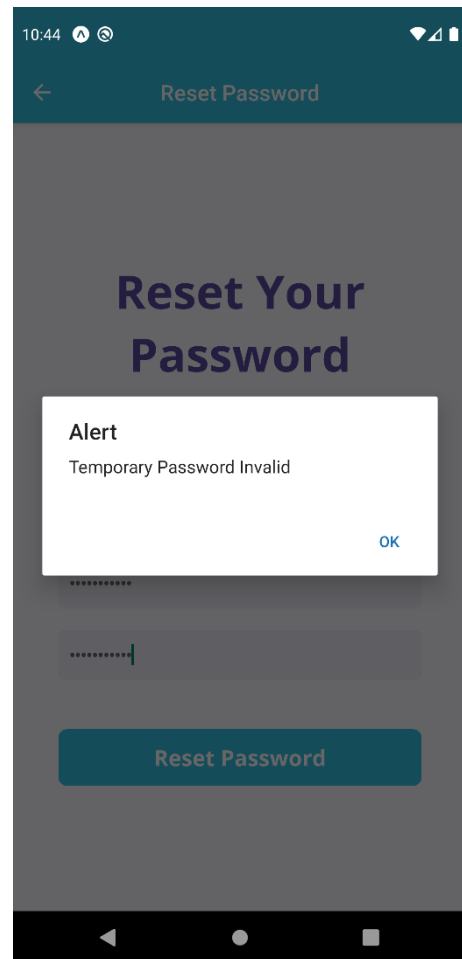
The screenshot shows a mobile application interface for resetting a password. At the top, there is a blue header bar with a back arrow on the left and the text "Reset Password" in the center. Below the header, the main content area has a light purple background. It features the title "Reset Your Password" in a large, bold, dark blue font. Underneath the title, a smaller line of text reads: "Reset your password using the temporary password we sent to your registered phone number". There are three input fields with light purple backgrounds and gray placeholder text: "Temporary password", "New Password", and "Confirm Password". Below these fields is a prominent blue button with the text "Reset Password" in white. At the very bottom of the screen, there is a black navigation bar with three white icons: a back arrow, a circle, and a square.

Figure 65: reset password screen



The screenshot shows a mobile app interface for resetting a password. At the top, there's a blue header with a back arrow and the text 'Reset Password'. Below the header, the title 'Reset Your Password' is displayed in a large, bold, purple font. Underneath the title, a subtitle reads: 'Reset your password using the temporary password we sent to your registered phone number'. There are three input fields: 'Temporary password', 'New Password', and 'Confirm Password'. Each field has a light blue border and a 'Required*' label below it. At the bottom, there is a large blue button labeled 'Reset Password'.

Figure 66: reset password validations



The screenshot shows the same 'Reset Your Password' screen as Figure 66, but with an alert dialog box overlaid. The alert dialog has a white background and a dark blue border. It contains the text 'Alert' and 'Temporary Password Invalid'. There is an 'OK' button in the bottom right corner of the dialog. The background of the app is dimmed.

Figure 67: invalid temp password alert

On successful provision of the valid temporary password and the new password, the password of the user will be reset.

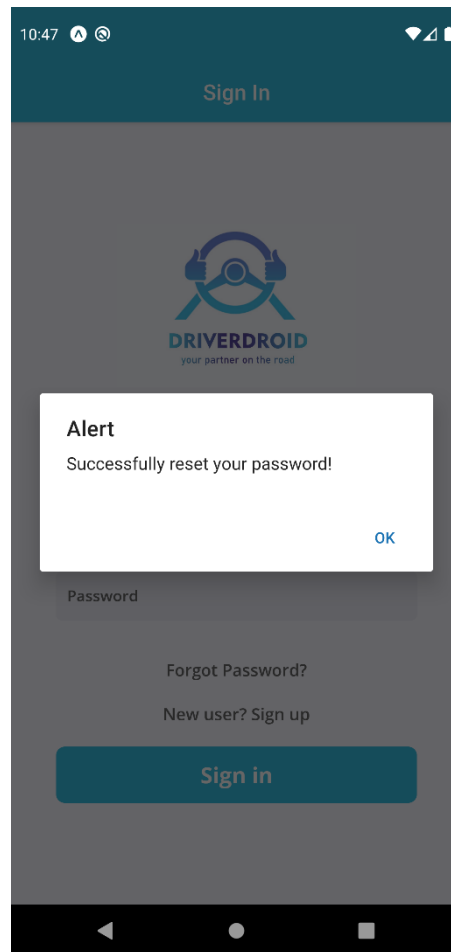


Figure 68: reset password alert

A.1.7 Implemented Home screen

When user successfully login to the app he/ she will be directed to the home screen where he/ she will be presented with a drawer to navigate between the screens available in the app.

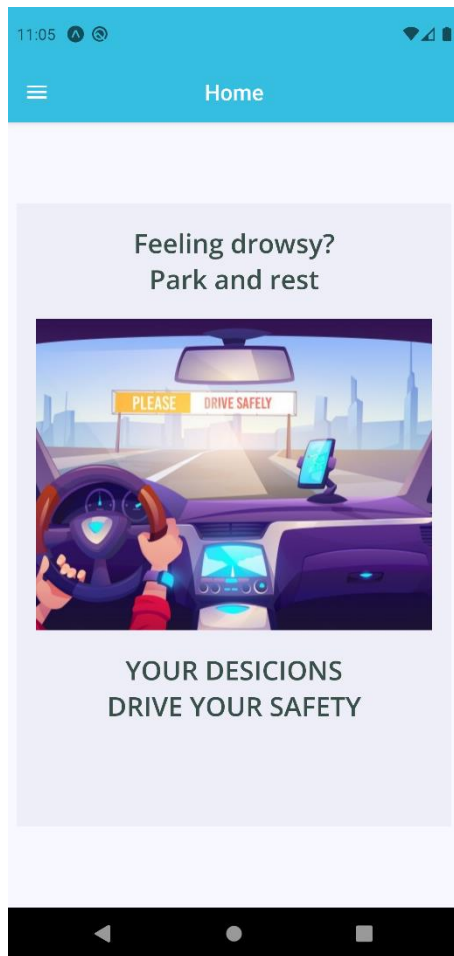


Figure 69: home screen

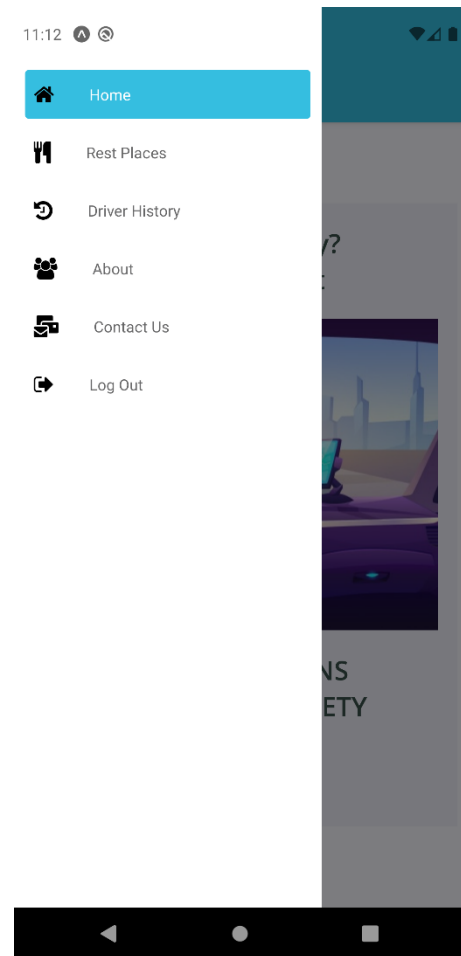


Figure 70: drawer navigator

A.1.8 Implemented About screen

This screen holds information about the ‘Driverdroid’ mobile application.

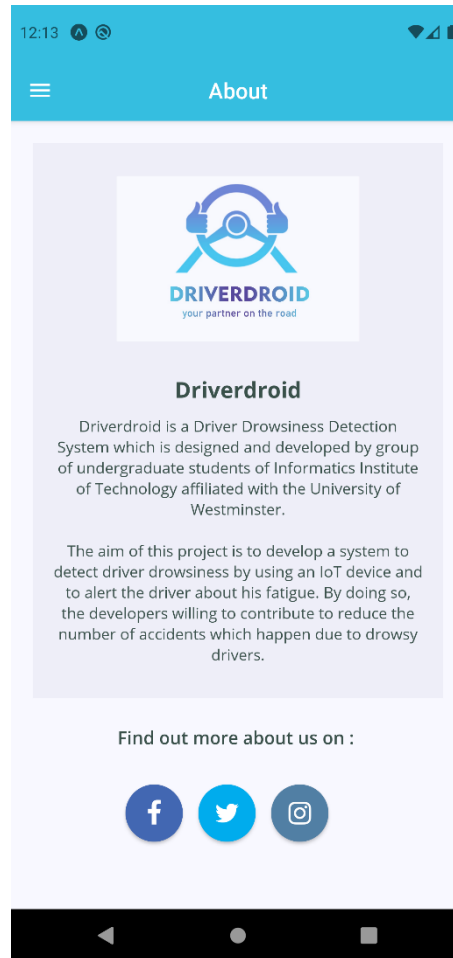
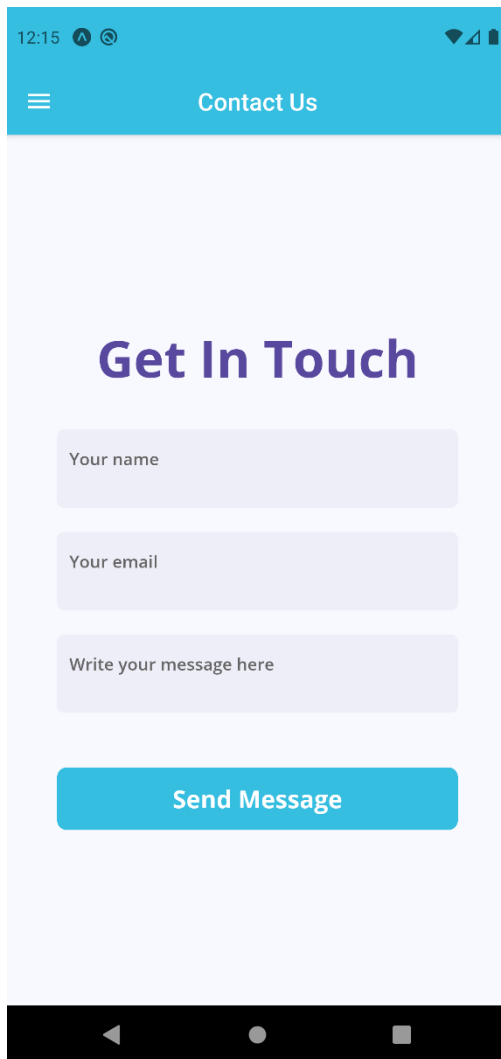


Figure 71: about screen

A.1.9 Implemented Contact Us screen

This screen let user to send his/ her feedback about the app, to the developers of the mobile application.



12:15

Contact Us

Get In Touch

Your name

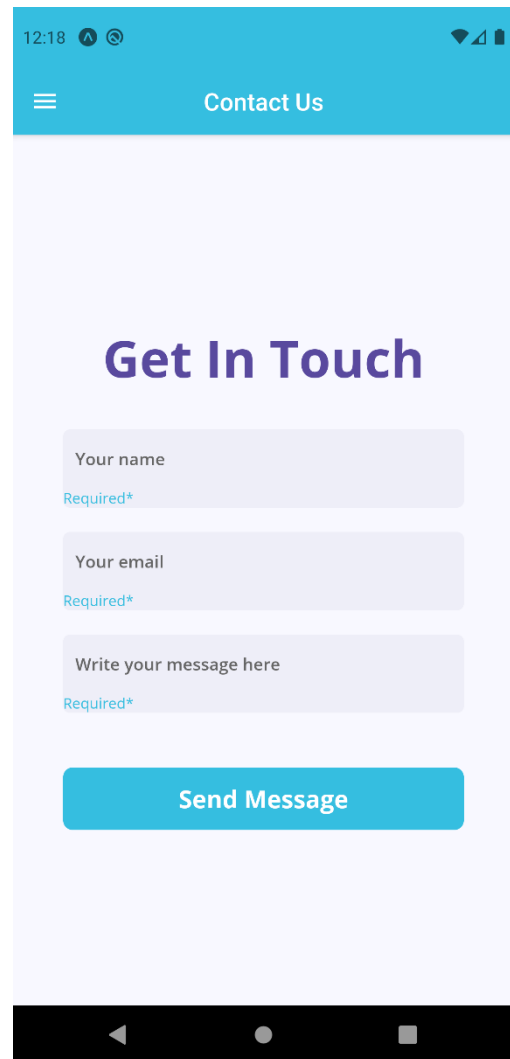
Your email

Write your message here

Send Message

This screenshot shows the 'Contact Us' screen at 12:15. It features a blue header with a hamburger menu icon and the text 'Contact Us'. Below the header is a large light purple area containing the heading 'Get In Touch' in bold purple text. Underneath the heading are three light purple input fields: 'Your name', 'Your email', and 'Write your message here'. At the bottom of this section is a blue button labeled 'Send Message'. The Android navigation bar is visible at the very bottom.

Figure 72: contact us screen



12:18

Contact Us

Get In Touch

Your name
Required*

Your email
Required*

Write your message here
Required*

Send Message

This screenshot shows the 'Contact Us' screen at 12:18, displaying validation messages. The layout is identical to Figure 72, but each input field now has a red 'Required*' error message below it. The 'Send Message' button remains blue and is still visible at the bottom of the form area.

Figure 73: contact us screen validations

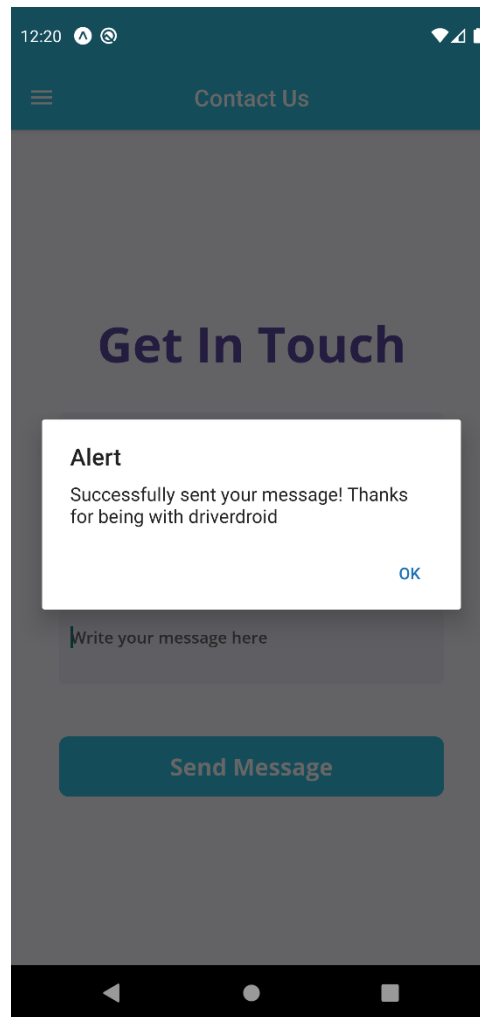


Figure 74: sent feedback alert

A.1.10 Implemented Log Out screen

This screen let user to choose whether he/ she want to log out from the application.

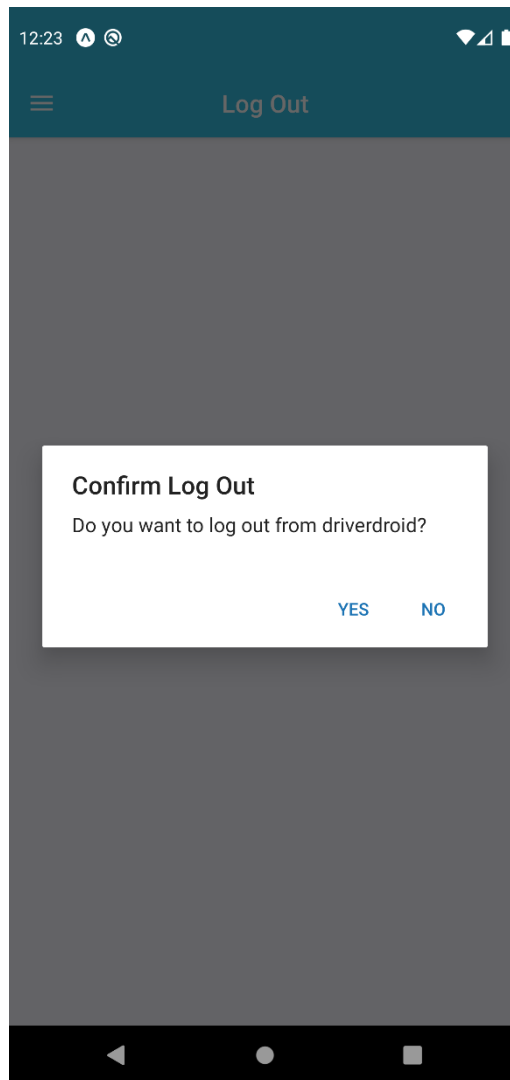


Figure 75: log out screen

If user select to log out from the app, he/ she will be logged out from the driverdroid, and then will be directed to the sign in screen, while if he/ she selected not to log out, he/ she will be directed to the home screen.

Team contribution on the report

Chapter 01 - Implementation

- 1.1 Chapter Overview - Dilanka
- 1.2 Overview of the prototype - Ravindu
- 1.3 Technology selections - Sanduni, Geethika
- 1.4 Implementation of the data science component - Ravindu, Odhil
- 1.5 Implementation of the backend component - Rusiru, Dilanka
- 1.6 Implementation of the mobile application - Sanduni, Geethika
- 1.7 Deployments/CI-CD Pipeline - Dilanka
- 1.8 Chapter Summary - Rusiru

Chapter 02 - Testing

- 2.1 Chapter Overview - Rusiru
- 2.2 Testing Criteria - Odhil
- 2.3 Testing functional requirements - Odhil
- 2.4 Testing non-functional requirements - Odhil
- 2.5 Unit testing - Rusiru
- 2.6 Performance testing - Ravindu
- 2.7 Usability testing - Sanduni, Geethika
- 2.8 Compatibility testing - Dilanka
- 2.9 Chapter Summary - Rusiru

Chapter 03 - Evaluation

- 3.1 Chapter Overview - Dilanka
- 3.2 Evaluation methods - Odhil, Ravindu
- 3.3 Quantitative evaluation - Rusiru
- 3.4 Qualitative evaluation - Sanduni
- 3.5 Self evaluation - Geethika
- 3.6 Chapter Summary – Dilanka

Chapter 04 - Conclusion

- 4.1 Chapter Overview - Dilanka
- 4.2 Achievements of aims and objectives - Odhil
- 4.3 Legal, social, ethical and professional issues - Sanduni, Geethika
- 4.4 Limitations of the research - Rusiru
- 4.5 Future enhancements - Ravindu
- 4.6 Extra work – Rusiru

4.7 Concluding Remarks - Dilanka