B1: FocusEd

Authors: Heidi Batres, Vaheeshta Mehrshahi, Danielle Kakish: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—FocusEd is a daytime driving aid that will alert a driver if it is determined that the driver is engaging in distracted behavior – sleeping or texting. Based on facial detection, facial landmarking, eye classification, and head pose estimation through a video stream, FocusEd will use several software algorithms to detect a sleeping or texting driver. The main components of FocusEd are a Jetson Xavier NX and a Raspberry Pi Camera Module V2, in which the Jetson will be responsible for the computing power of the system and the Raspberry Pi Camera will intake the video of the driver. FocusEd will then output an audio alert in order to alert the driver that they are showing a distracted behavior, allowing the driver to correct before another alert is given. The project focuses on re-educating the driver on the skills that would prevent distracted driving.**

***Index Terms*—distracted driving, Eye Aspect Ratio (EAR), eye classification, facial detection, facial landmarking, focus timer, head pose estimation, Histogram of Oriented Gradients (HOG), Support Vector Machine (SVM)**

1 INTRODUCTION

FocusEd serves as a way for drivers to curb their distracted day driving while simultaneously improving road safety and their own driver education. With the increase in distracted driving-related deaths in recent years, it is rather imperative that a solution be found to help drivers fix these habits. Much of this increase has to do with the increased reliance on smartphones as technology evolves and people feel the need to send a text or email immediately rather than waiting.

Current technologies are mostly focused on the idea of lane detection – alerting the driver if they get too close to another car. However, this solution focuses more on the car than on the driver themself. If a driver is already drifting into another lane without meaning to, they can already be posing a danger to other cars around them. Thus, we want to correct that behavior before the driver even begins to drift and potentially cause an accident. In order to do this, FocusEd must detect a distraction and output an audio alert within a 2 second interval in order for the driver to swiftly correct their behavior.

2 DESIGN REQUIREMENTS

2.1 Face Detection

The first design requirement for this project is face detection. This is important as the rest of the algorithms depends on the smaller image of the driver's face so we would like to have at least 90% detection accuracy with an upper bound of 1% for false negatives and 9% for false positives. To test our Histogram of Oriented Gradients and Support Vector Machine (HoG and SVM) facial detection algorithm, we plan to use the Labeled Faces in the Wild (LFW) face data set in addition to testing on sampled drivers inside a vehicle.

2.2 Facial Landmarking

Secondly, we require accurate facial landmarking to be able to determine head poses and 2D eye landmarks for eye classification. Our device requires landmarking as close to the true points as possible, thus we required a maximum 10 pixel radius difference between our algorithm detected points and the true points of the driver. To test this, we will be calibrating our algorithm with different angles of head poses to determine the landmarks of different individuals and compare detected points with trained points to create a more precise algorithm.

2.3 Head Pose Estimation

Building off of facial landmarking, we also require head pose estimation to determine if the driver is performing a regular driving movement or distracted. We require 85% head pose estimation accuracy. Since we are creating this algorithm on our own, its accuracy relies heavily on that of facial landmarking and the facial detection. Thus, we needed to make the accuracy slightly smaller than that of the detection and landmarking in order to account for this. However, the focus timer dedicated to head pose estimation and the additional case of drowsiness will help to determine more of the distracted driving behaviors. In order to test this, we will compare the actual direction of the driver's head to the head pose estimation output.

2.4 Eye Classifier

Additionally, we require eye classification to determine whether the driver is falling asleep at the wheel. We require our classifier to achieve 90% accuracy with an upper bound of 2% for false negatives and 8% for false positives. To test we will be using a custom training set to determine the threshold for the Eye Aspect Ratio (EAR) algorithm that works with different eye shapes and test our calibration used with head pose estimation as well. We plan to calibrate before the user begins driving.

2.5 Focus Timers

Since there are normal movements that would make it hard to define distracted vs normal driving movements, we require a focus timer for both head pose and eye classification. For example, if the driver's eyes are closed for an extended period of time, they are not blinking and are probably drowsy. Similarly, if a driver is looking down too long, they are likely looking at their phone. The focus timer for eye classification will be 1 second as this can classify between normal blinking and possible drowsiness. The focus timer for head pose will be 2 seconds since this allows for the periodic checks of the driver's mirrors. These focus timers will determine the difference between normal versus distracted behaviors at least 90% of the time. We will be testing these time cutoffs on a sample of average drivers and make any adjustments accordingly. These in combination with the eye classifier and head pose estimator shall take into consideration both positioning and timing.

2.6 Power Supply

Our system will be located in the vehicle during the day, hence the power supply for the device needs to last for the driver's average commute. We will be using a portable power supply and testing that it is powered between 8 to 10 hours to account for a driver's commute to and from work on an average work week. To notify the driver that they are distracted, we require an audio alert triggered 99% of the time when a distracted movement is detected. We also require that the driver responds to the alert by refocusing their head position and eyes to the road, and thus we will test that our system no longer produces an audio alert once the driver refocuses within 3 iterations of the system.

2.7 System Latency

Finally, because we want to correct the driver in real time and prevent accidents, we need an efficient and fast system to alert the driver. We will require our system's full iteration to complete within 2 seconds and we will test this by timing each algorithm separately, at phased integration and at the final integration.

3 ARCHITECTURE OVERVIEW

To meet the system requirements discussed in the previous section, we will be creating an enclosed camera system to be placed directly in front of the driver on the dashboard of the car as depicted in Fig. 1. Placing the system directly in front of the driver will ensure a higher accuracy in our software algorithms. The Jetson Xavier NX will be placed inside a case to ensure that the Jetson is not damaged will driving.

In constructing the system, the Raspberry Pi V2 Camera Module will be plugged into the Jetson Xavier with an opening in the enclosure for the Camera to view the driver. In addition, the USB Speaker that will output an audio alert will also have an opening in order to alert the driver with a loud enough sound, since the sound would be muted otherwise. The TalentCell Battery will also be plugged into the power jack of the Jetson to serve as the power source.

Our user story is depicted in Fig. 8. First, the user will turn on FocusEd to begin the process. Then, the Raspberry Pi V2 Camera module will intake a video stream of the driver's face, and then subsequently send this video stream to the Jetson. The Jetson will then perform the facial detection algorithm in order to detect the driver's face. Following this, the algorithm for facial landmarking will run to landmark the driver's face for further assessment. The system will then simultaneously run both the eye classification and head pose estimation algorithms to determine whether the driver of the vehicle is exhibiting behavior of sleeping or texting, respectively.

During this, the focus timers will be running to actually make this determination. We will be having two separate focus timers, one for drowsiness and one for head pose. The drowsiness focus timer will run in conjunction with our eye classification algorithm, and if the eye classification of the driver's eyes are closed for more than 1 second, then a signal will be communicated that the driver must be alerted. The head pose focus timer will work in conjunction with our head pose algorithm. If it is determined that the driver's head pose is in the negative quadrant of an imaginary axis for longer than 2 seconds, then a signal will be communicated that the driver must be alerted.

The USB speaker that is connected to the Jetson of FocusEd will then output a 1.5 second vocal audio alert that we will record in order to tell the driver to refocus on the road. If the driver does not correct their behavior within the next 2 seconds, the driver will once again be alerted with an audio alert. If the driver is not determined to have been distracted in the beginning, the algorithms will continually run with the video stream of the driver's face so any distraction will be alerted in real-time.

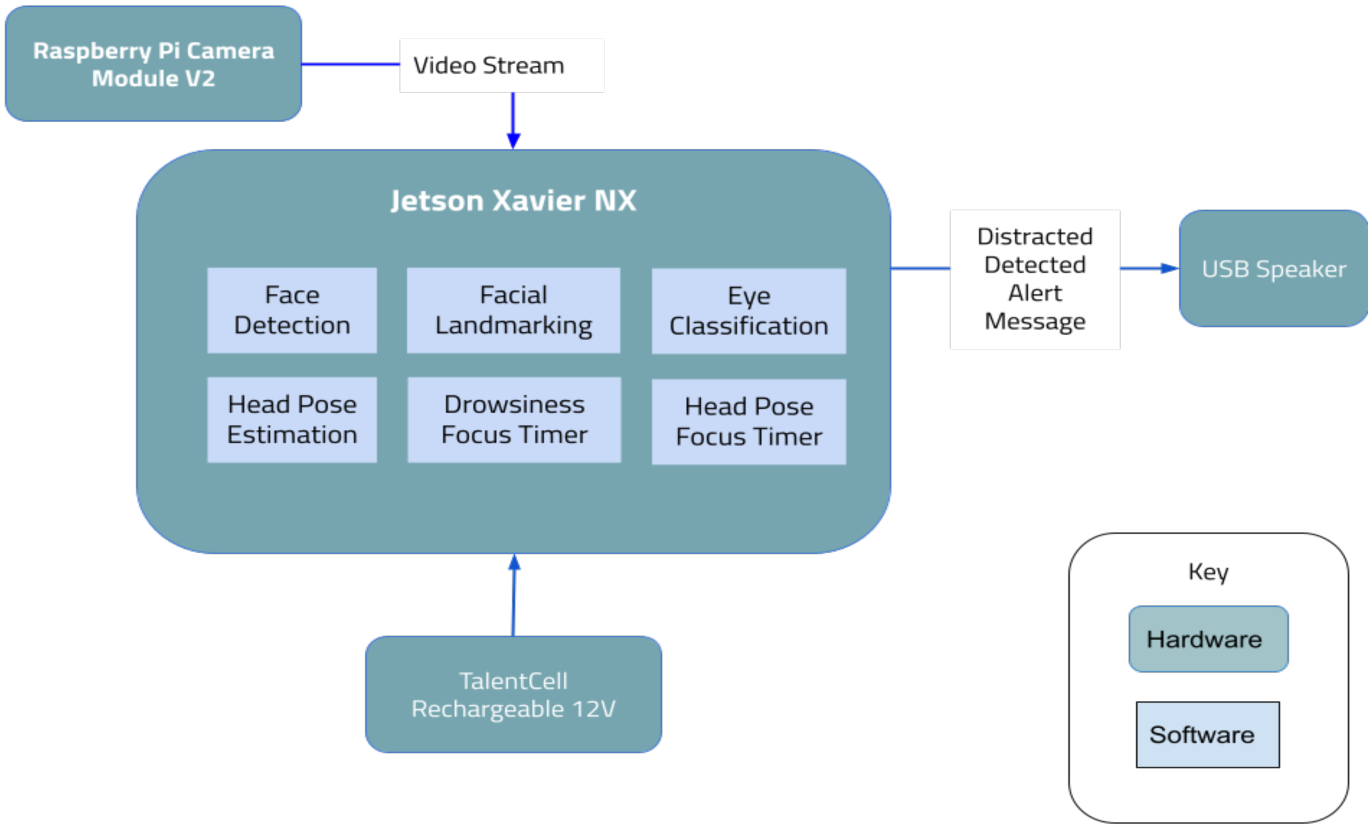


Figure 1: Overall System Block Diagram

4 DESIGN TRADE STUDIES

4.1 Face Detection Algorithm

For this specification of our system we opted to use HoG plus SVM over Haar Cascades. Our reasoning for this choice was that HoG works well for frontal and slightly non-frontal and works under small occlusion. Also, since we are looking at just the driver's face in the image, we don't need to worry about too small of a face for the detector. The HoG and SVM detector is from Dlib's Python libraries.[3]

4.2 Facial Landmarking Algorithm

We will use Dlib's default 68 points landmarks from their shape predictor but will eventually remove unnecessary points so that it can run faster. For example, we will not need eyebrows in our points for eye classification or head pose estimation.[4][2]

4.3 Head Pose Estimation

Initially, we thought following examples that convert the 2D facial landmark points to 3D points would result the best for head pose estimation. However, since we are running multiple algorithms and our landmarks might not be fully precise, we opted to use an imaginary axis over the driver's face, with the origin being the tip of the nose.

4.4 Eye Classifier

For classifying if the driver's eyes are closed or not, we are using the Eye Aspect Ratio (EAR). Other methods include matching an open eye template to the user's eye to determine if the eye is closed or not[6]. Moreover, other eye classifiers rely on thresholding to determine the whites of the eyes and if these white regions disappear or not. However, we wanted an efficient eye classifier that relied on calculations using 2D landmarks that are already outputted by our facial landmarker. Thus, we chose to use the EAR because of the reduced computation time required.

4.5 Focus Timer

Initially, when we were thinking of creating a focus timer, we struggled with determining the proper timing to wait before alerting the driver because we would have to distinguish between distracted driving and normal driving. During our initial design, we determined that we would have a singular focus timer for a set period of time that was yet to be decided upon. However, following our research we determined that having two focus timers – one for texting and another for the sleeping portions of our scope would suit the project best because the timing is different. According to the Rules of Drowsiness for driving, a person is considered drowsy if their eyes are closed for longer than 1 second.[5] As for the texting, we made the determination that 2 seconds would be a sufficient time before alerting a driver because it is generally known that a driver must keep approximately 3-4 seconds behind the car in front of them. So within that time of 2 seconds, alerting the driver would still keep them a couple seconds behind the car in front. Thus, we have two focus timers in our design, one for drowsiness and another for head pose estimation.

4.6 Dlib

Dlib provides various open source machine learning and image processing algorithms. Additionally has good documentation to reference. Dlib also supports GPU acceleration which pairs well with the Jetson Xavier NX.[1]

4.7 OpenCV

OpenCV provides computer vision and machine learning libraries for Python and is used in projects in combination with Dlib for image processing of video and images.

4.8 Jetson Xavier NX

During our initial design, we believed that NVIDIA Jetson Nano would be sufficient for the computing power that we needed for our project and that the size would be smaller and better suited for placement directly in front of the driver. Following our proposal presentation, our instructors expressed concern that the Jetson Nano would simply not be fast or powerful enough for all of the computing that we intended to perform. Following this, we researched further into the NVIDIA Jetson Xavier NX and determined that its size was not much different than the Jetson Nano and was much more powerful, thus increasing the frames per second which would give us a much closer ”real-time” alert.

4.9 Raspberry Pi Camera Module V2

The Raspberry Pi Camera Module is compatible with various NVIDIA products including the Jetson Xavier and its default packages (JetPack SDK) and is recommended in NVIDA forums. Additionally, there are various examples online for reference to consult. Most importantly, it records live camera feed.

5 SYSTEM DESCRIPTION

This project consists of both hardware and software. The project's hardware system consists of a Jetson Xavier NX, Raspberry Pi Camera Module V2, TalentCell Rechargeable 12V Battery Pack, and Mini External USB Stereo Speaker. Fig. 2 presents the hardware block diagram. The software system, which runs on the Jetson Xavier NX, consists of the processing of the Raspberry Pi Camera Module V2's video stream in order to determine of the driver is distracted and signaling subsequent alerts. This includes facial detection, facial landmarking, eye classification, head pose estimation, focus timing, and the audio alert. Fig. 3 presents the software block diagram.

5.1 Jetson Xavier NX

The Jetson Xavier NX will serve as the processor for the video stream it receives from the Raspberry Pi Camera Module V2, which is plugged into port J1 on the Jetson Xavier NX. Upon receiving power, the Jetson will begin processing the video stream input and running the facial detection, facial classification, eye classification, and head pose estimation algorithms on each frame.These algorithms are described in more detail below.

5.2 Raspberry Pi Camera Module V2

Our camera of choice is the Raspberry Pi Camera Module V2, which has a ribbon connector that is connected to the Jetson Xavier NX via port J1, which enables use of CSI cameras. This camera has a Sony IMX219 8-megapixel sensor and GStreamer is used to interface with the camera using the Jetson.

5.3 TalentCell Rechargeable 12V Battery Pack

To power the Jetson Xavier NX, we will be using the TalentCell Rechargeable Battery Pack. The battery pack delivers 12V of power, which satisfies the 9-20V power requirement of the Jetson Xavier NX.

5.4 Mini External USB Stereo Speaker

For our audio alert, we will be using Adafruit's Mini External USB Stereo Speaker. This is a USB-only speaker, and thus is both powered and receives audio via one of the Jetson Xavier NX's USB ports, which are J6 and J7. Because the speaker has a 46 inch cable, we have flexibility in securing the speaker onto our dashboard.

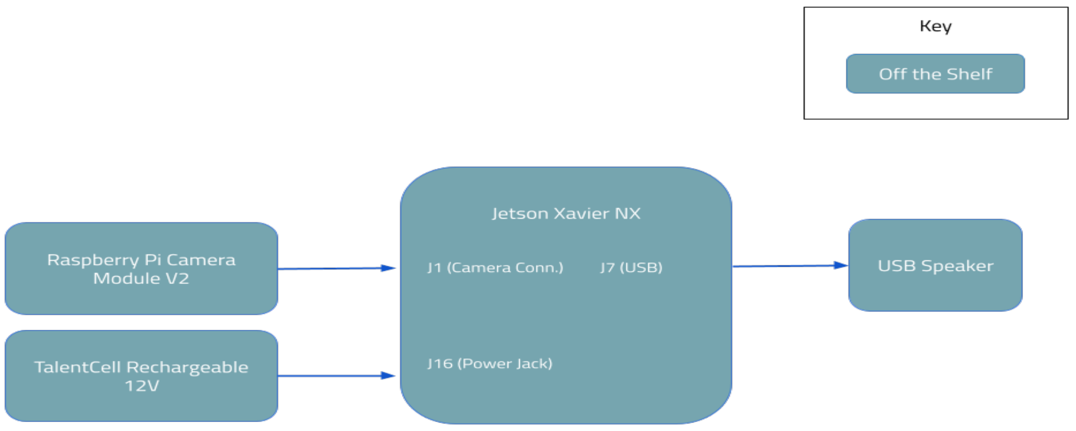


Figure 2: Hardware Block Diagram

5.5 Facial Detection and Landmarking

Facial detection of the driver will be performed using the Histogram of Oriented Gradients and Support Vector Machine (HoG and SVM) algorithm. HoG is a feature representation method. A HOG descriptor is first extracted from our image frame from our video stream. Then, a SVM, which is a supervised machine learning model, is used to train a model to detect a face. Dlib and OpenCV are the libraries that will be used for this.

Facial landmarking will be obtained using Dlib's facial landmark detector. This estimates the location of 68 2D coordinates, which we reduced down to 30 absolutely necessary facial coordinates. These coordinates will include 12 eye landmarks, 3 nose landmarks, 2 mouth landmarks, and the remainder being chin and jaw landmarks.[4][2]

5.6 Eye Classification

Using landmarks retrieved from our facial landmarking, we then classify whether an eye is opened or closed. This is done by finding the proportion between the width and height of the eye based on 6 eye landmarks. This proportion is called the Eye Aspect Ratio, or EAR, which is found using the following formula[7]:

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\,\|p_1 - p_4\|} \tag{1}$$

where $p_1, ..., p_6$ are 2D landmark locations provided by Dlib's 68-point facial landmark detector. The smaller the ratio, the closer the driver's eyelids are together, signalling that their eyes are closing. The threshold for the EAR will be determined through calibration and training.

5.7 Head Pose Estimation

From the landmarks retrieved from the facial landmarking phase, we can calibrate on system turn on, an origin for an imaginary axis. This origin would be the tip of the nose. Thus, with an origin being defined, if the driver's head tilts down or moves to the right, the tip of the nose would be in the negative quadrants of the axis. Hence we could determine that the driver is possibly distracted.

5.8 Focus Timer

The focus timers for head pose and eye classification help determine whether a driver is distracted or not. There are normal driver movements that require the driver to not face forward such as, checking their side and rear view mirrors. The timers account for these brief glances and limit the alerts as to not inconvenience the driver.

5.9 Audio Alert

To ensure that the driver is awoken from their drowsiness, our audio alert is a clear 1.5-second vocal instruction to refocus on the road. The volume of the alert is loud enough to be heard over a car radio. Additionally, we want optimal volume experienced by the driver, so our speaker will be situated facing our driver.

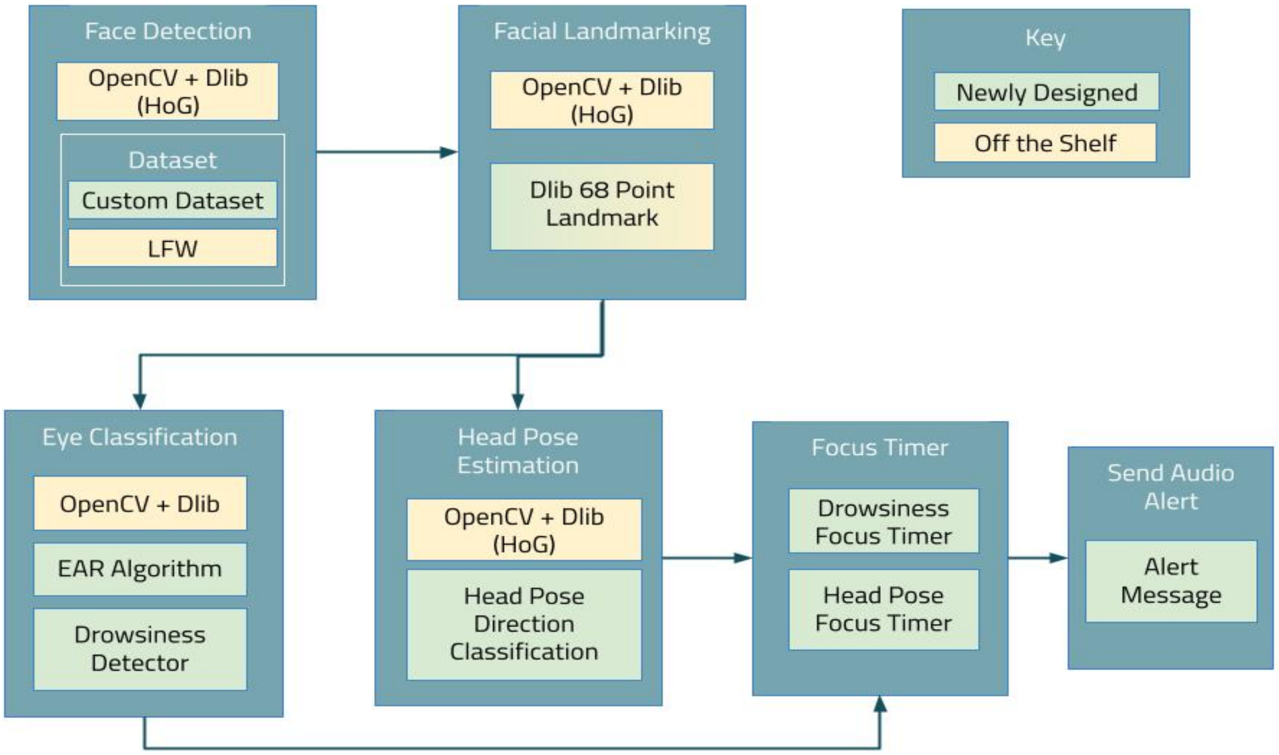


Figure 3: Software Block Diagram

6 PROJECT MANAGEMENT

6.1 Schedule

Fig. 4, 5, and 6 present this project's Gantt Chart. This schedule has faced significant revisions since presenting the project proposal due to the replacement of the vibration alert with an audio alert, as well as adding more time for training and slack. We focused on attention to detail while determining our tasks, and thus our schedule lists a total of 44 tasks.

6.2 Team Member Responsibilities

All team members will be working together throughout the design, development, and testing of this project. We have assigned each team member to be in charge of specific aspects of the design. Heidi is focused on implementing and testing the facial detection, facial landmarking, and head pose estimation algorithms on the Jetson Xavier. Vaheeshta is focused on implementing and testing the eye classification algorithm, training the eye classification algorithm, and creating the system calibration. Danielle is focused on implementing the focus timer, creating the audio alert system, and helping with training.

6.3 Budget

The table represented by Fig. 7 presents all materials that were purchased for the development of this project. Out of our $600 budget, we used $455.57. Most of our budget is used by the NVIDIA Jetson Xavier NX Developer Kit, which totalled $399. However, we were able to still have room in our budget largely due to us having a majority of our other materials either on hand or provided by the 18-500 inventory.

6.4 Risk Management

There are several risks we must consider while planning the project. For one, there is significant concern about our system latency. Specifically, detection of drowsiness and alerting the driver might not happen in the desired time frame. While we have metrics discussed previously to determine how fast we would like our system to be, until we begin our first tests of our system, we are unsure if we can meet these metrics. Therefore, our risk mitigation is in the form of a fallback design. We hope to perform face detection on the Jetson Xavier, then crop out the face and send this to the cloud using AWS to perform the remainder of our computation. Then, the Jetson Xavier would receive the signal from AWS of whether or not to trigger the audio alert. The resources necessary to implement this change are already accounted for in our requested AWS credits. As for our schedule impact, implementing this would be done during our allocated slack time in our schedule. Additionally, another risk we may encounter is that we may face issues with distinguishing between normal movements and distracted movements, and thus do not detect all distracted instances. Our mitigation is to slightly overcorrect to ensure that we at least detect the distracted cases. We would rather have more false positives than false negatives, while still keeping both within reason. Moreover, another risk we anticipate involves our TalentCell power bank not being able to power the Jetson Xavier for 8-10 hours as our metric hopes. Since our Jetson Xavier will be performing continuous computations, this is a very likely possibility. Thus, we will power the Jetson Xavier using our car's power outlet. Lastly, our final risk involves varying lighting conditions affecting our drowsiness detection accuracy. We will avoid this risk by limiting our scope to only direct lighting conditions, with the aid of a ring light.

References

[1] *Compile: Using dlib from Python.* URL: `http://dlib.net/compile.html`.

[2] *Dlib Shape Predictor.* URL: `http://dlib.net/python/index.html#dlib.shape_predictor`.

[3] Vikas Gupta. *Face Detection – OpenCV, Dlib and Deep Learning.* Oct. 2018. URL: `https://learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/`.

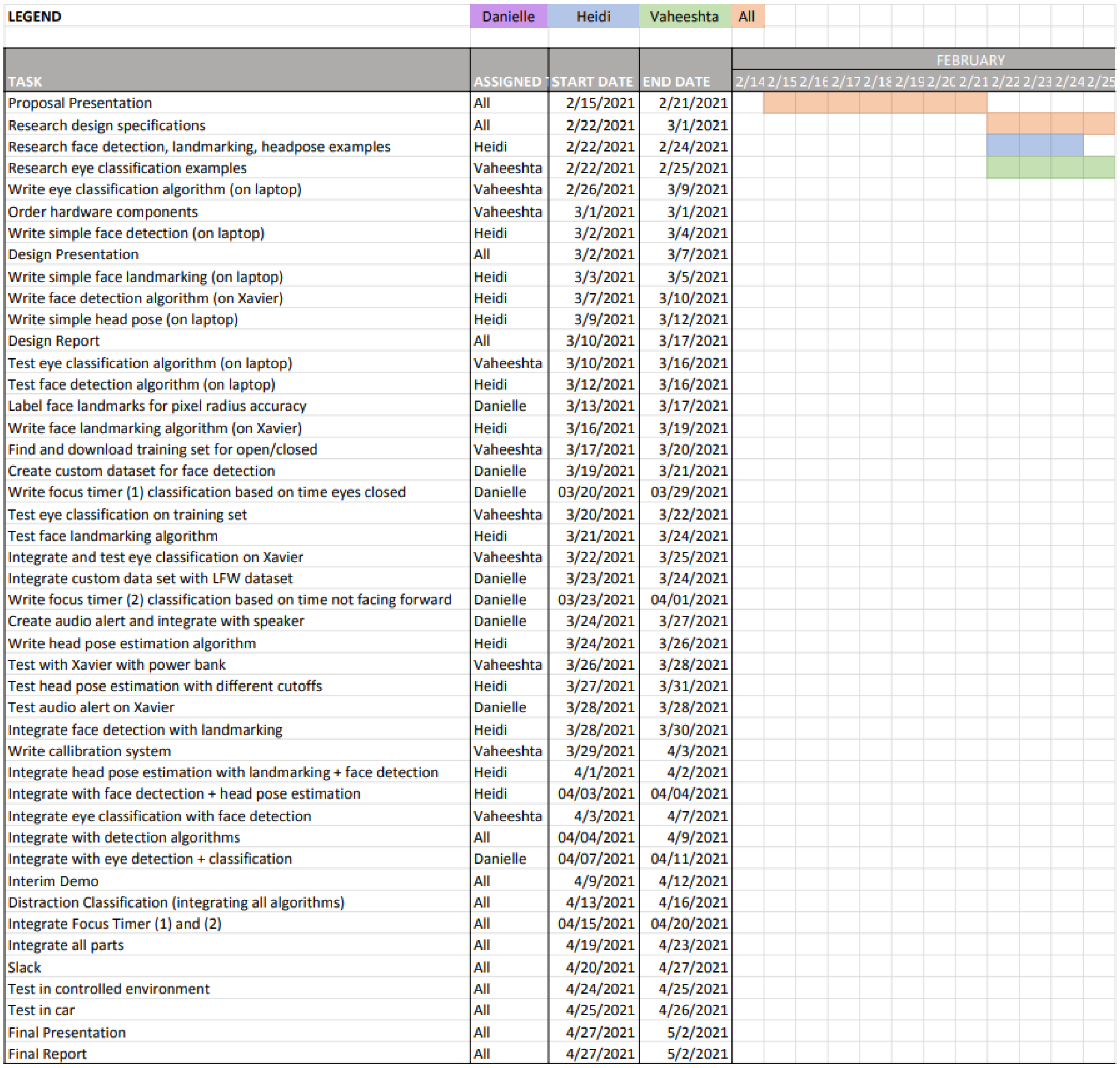
[4] Davis King. *GitHub: Face Landmark Detection.* URL: `https://github.com/davisking/dlib/blob/master/python_examples/face_landmark_detection.py`.

[5] Caio Bezerra Souto Maior et al. "Real-time classification for autonomous drowsiness detection using eye aspect ratio". In: *Expert Systems with Applications* 158 (Nov. 2020), p. 113505.

[6] Talal Bin Noman. "Mobile-Based Eye-Blink Detection Performance Analysis on Android Platform". In: *Front* (Feb. 2018).

[7] Tereza Soukupová. "Real-Time Eye Blink Detection using Facial Landmarks". In: *21st Computer Vision Winter Workshop* (Feb. 2016).

Appendix A



LEGEND	Danielle	Heidi	Vaheeshta	All

				FEBRUARY											
TASK	ASSIGNED	START DATE	END DATE	2/14	2/15	2/16	2/17	2/18	2/19	2/20	2/21	2/22	2/23	2/24	2/25
Proposal Presentation	All	2/15/2021	2/21/2021		🟧	🟧	🟧	🟧	🟧	🟧	🟧				
Research design specifications	All	2/22/2021	3/1/2021									🟧	🟧	🟧	🟧
Research face detection, landmarking, headpose examples	Heidi	2/22/2021	2/24/2021									🟦	🟦	🟦	
Research eye classification examples	Vaheeshta	2/22/2021	2/25/2021									🟩	🟩	🟩	🟩
Write eye classification algorithm (on laptop)	Vaheeshta	2/26/2021	3/9/2021												
Order hardware components	Vaheeshta	3/1/2021	3/1/2021												
Write simple face detection (on laptop)	Heidi	3/2/2021	3/4/2021												
Design Presentation	All	3/2/2021	3/7/2021												
Write simple face landmarking (on laptop)	Heidi	3/3/2021	3/5/2021												
Write face detection algorithm (on Xavier)	Heidi	3/7/2021	3/10/2021												
Write simple head pose (on laptop)	Heidi	3/9/2021	3/12/2021												
Design Report	All	3/10/2021	3/17/2021												
Test eye classification algorithm (on laptop)	Vaheeshta	3/10/2021	3/16/2021												
Test face detection algorithm (on laptop)	Heidi	3/12/2021	3/16/2021												
Label face landmarks for pixel radius accuracy	Danielle	3/13/2021	3/17/2021												
Write face landmarking algorithm (on Xavier)	Heidi	3/16/2021	3/19/2021												
Find and download training set for open/closed	Vaheeshta	3/17/2021	3/20/2021												
Create custom dataset for face detection	Danielle	3/19/2021	3/21/2021												
Write focus timer (1) classification based on time eyes closed	Danielle	03/20/2021	03/29/2021												
Test eye classification on training set	Vaheeshta	3/20/2021	3/22/2021												
Test face landmarking algorithm	Heidi	3/21/2021	3/24/2021												
Integrate and test eye classification on Xavier	Vaheeshta	3/22/2021	3/25/2021												
Integrate custom data set with LFW dataset	Danielle	3/23/2021	3/24/2021												
Write focus timer (2) classification based on time not facing forward	Danielle	03/23/2021	04/01/2021												
Create audio alert and integrate with speaker	Danielle	3/24/2021	3/27/2021												
Write head pose estimation algorithm	Heidi	3/24/2021	3/26/2021												
Test with Xavier with power bank	Vaheeshta	3/26/2021	3/28/2021												
Test head pose estimation with different cutoffs	Heidi	3/27/2021	3/31/2021												
Test audio alert on Xavier	Danielle	3/28/2021	3/28/2021												
Integrate face detection with landmarking	Heidi	3/28/2021	3/30/2021												
Write callibration system	Vaheeshta	3/29/2021	4/3/2021												
Integrate head pose estimation with landmarking + face detection	Heidi	4/1/2021	4/2/2021												
Integrate with face dectection + head pose estimation	Heidi	04/03/2021	04/04/2021												
Integrate eye classification with face detection	Vaheeshta	4/3/2021	4/7/2021												
Integrate with detection algorithms	All	04/04/2021	4/9/2021												
Integrate with eye detection + classification	Danielle	04/07/2021	04/11/2021												
Interim Demo	All	4/9/2021	4/12/2021												
Distraction Classification (integrating all algorithms)	All	4/13/2021	4/16/2021												
Integrate Focus Timer (1) and (2)	All	04/15/2021	04/20/2021												
Integrate all parts	All	4/19/2021	4/23/2021												
Slack	All	4/20/2021	4/27/2021												
Test in controlled environment	All	4/24/2021	4/25/2021												
Test in car	All	4/25/2021	4/26/2021												
Final Presentation	All	4/27/2021	5/2/2021												
Final Report	All	4/27/2021	5/2/2021												

Figure 4: Gantt Chart Section 1

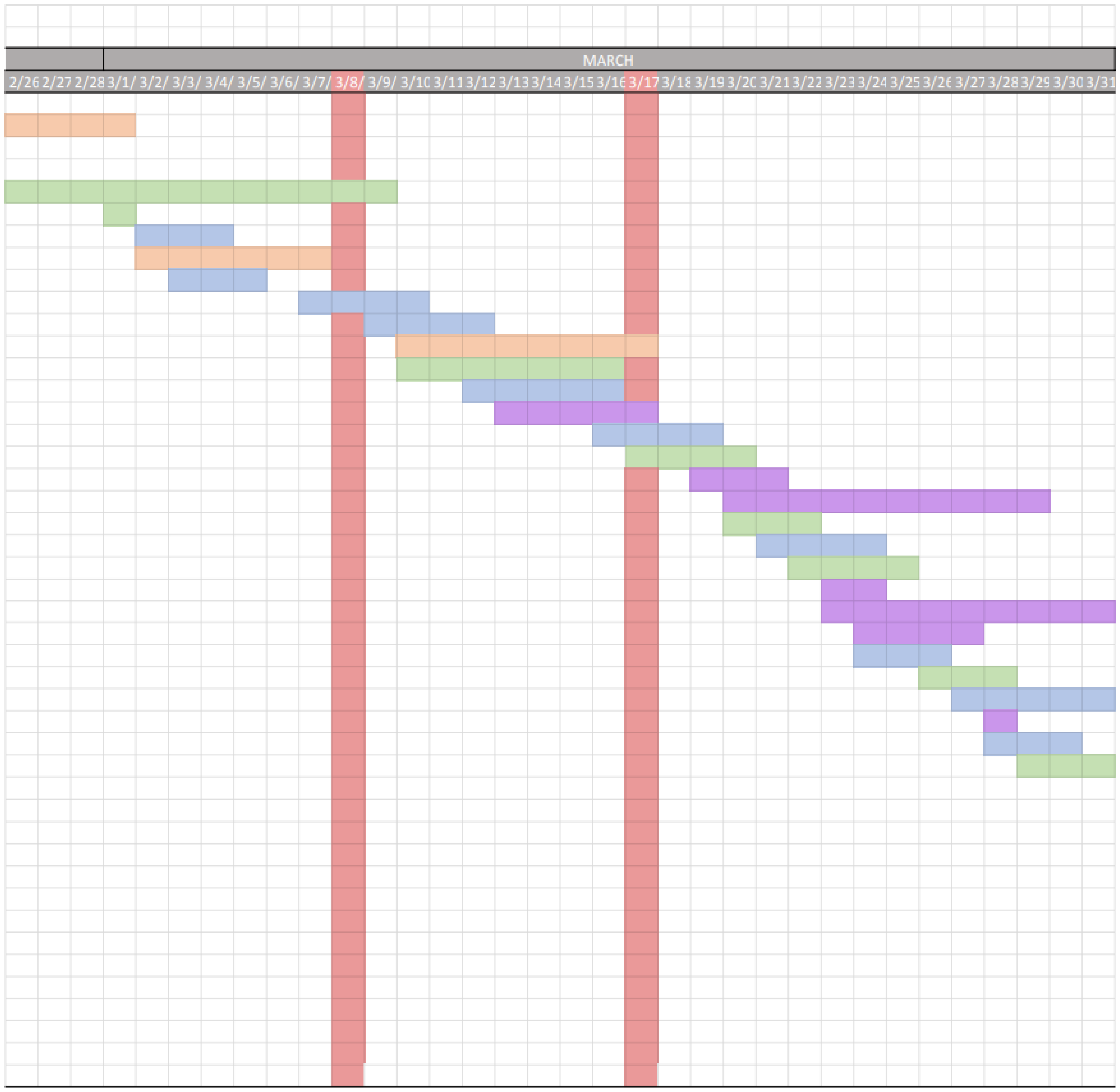


Figure 5: Gantt Chart Section 2

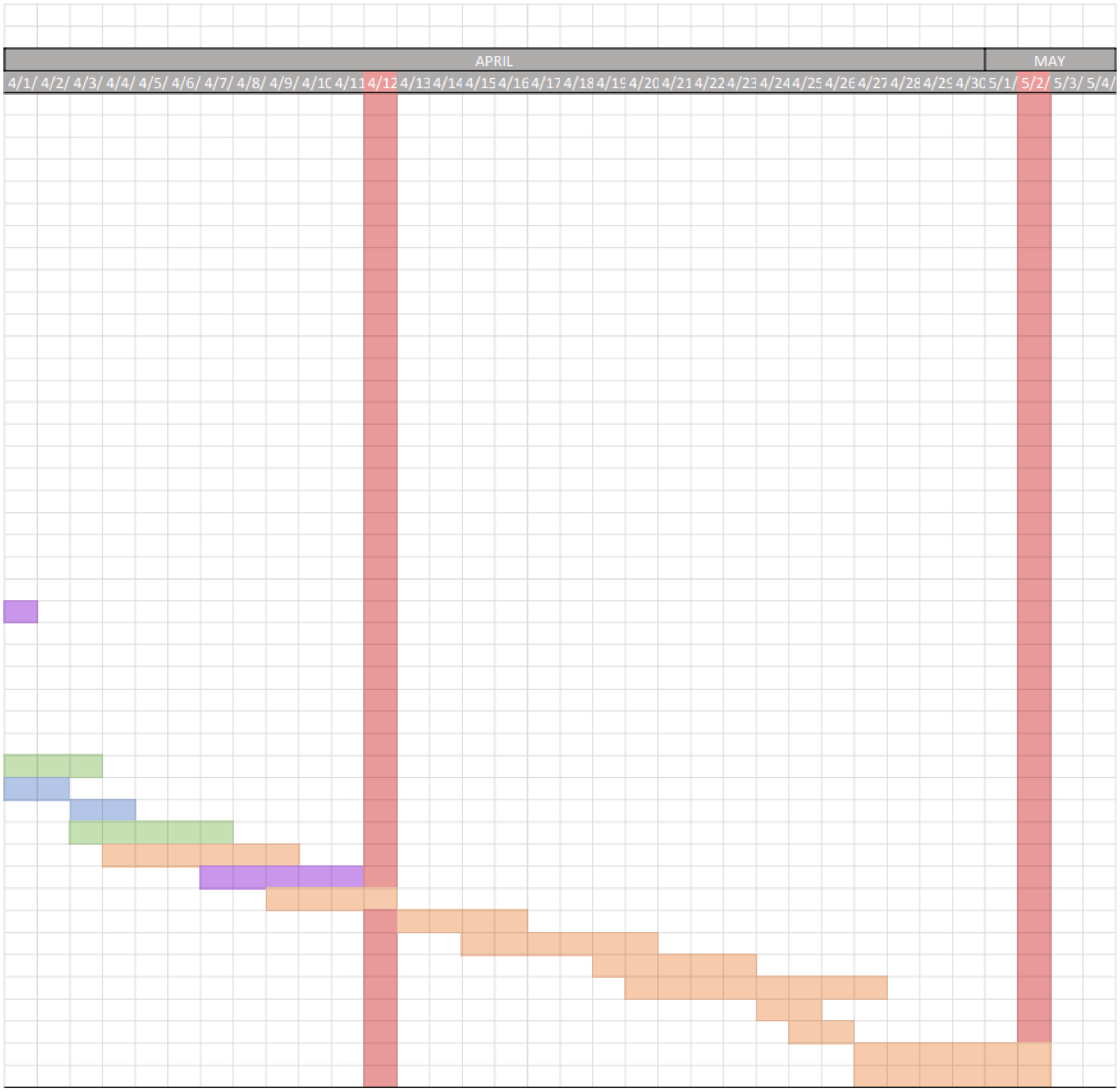


Figure 6: Gantt Chart Section 3

Item	**Supplier**	**Quantity**	**Price per Unit**	**Cost**
NVIDIA Jetson Xavier NX Developer Kit	Amazon	1	$399.00	$399.00
Raspberry Pi 3 Model B v1.2	18-500 Inventory	1	$0.00	$0.00
MicroSD Card 64 GB	Amazon	1	$13.15	$13.15
MicroSD Card 8 GB	Amazon	1	$5.93	$5.93
Raspberry Pi Camera Module V2	Amazon	1	$24.99	$24.99
Ring Light	Personal Inventory	1	$0.00	$0.00
TalentCell Rechargeable 12V Battery	18-500 Inventory	1	$0.00	$0.00
Mini External USB Stereo Speaker	Amazon	1	$12.50	$12.50
Monitor	Personal Inventory	1	$0.00	$0.00
Keyboard + Mouse	Personal Inventory	1	$0.00	$0.00
HDMI	Personal Inventory	1	$0.00	$0.00
Micro USB Charging Cable	Personal Inventory	1	$0.00	$0.00
Total				**$455.57**

Figure 7: Bill of Materials

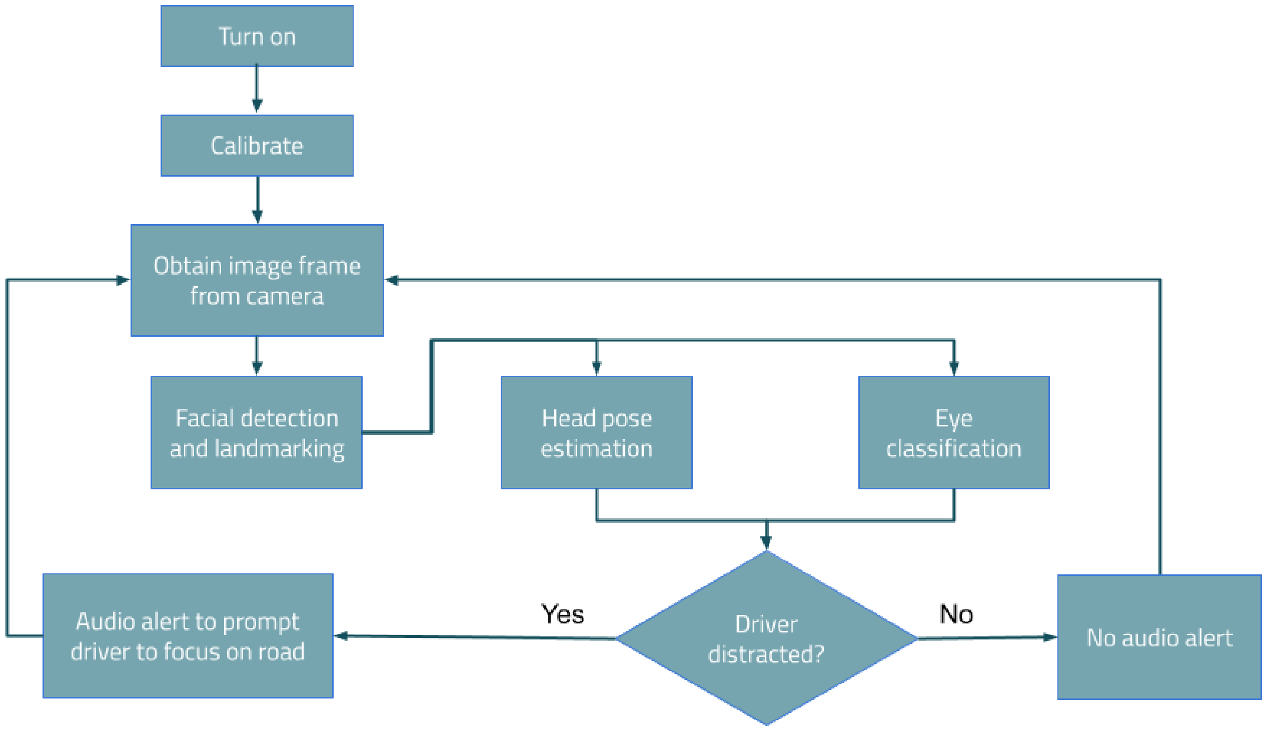

Turn on
Calibrate
Obtain image frame from camera
Facial detection and landmarking
Head pose estimation
Eye classification
Driver distracted?
Yes
No
Audio alert to prompt driver to focus on road
No audio alert

Figure 8: User Story