Project report on-

Digital Assistant for Vehicle Drivers

Submitted By-
Md. Tasnimul Khair Tousif
Roll: 1607015
Rahat Alamgir Porosh
Roll: 1607047

Under the Guidance Of:
Dola Das

Department of Computer Science and Engineering,
Khulna University of Engineering and Technology.

Date of Submission: 26/12/19

Signature:

Acknowledgement

By the grace of Almighty, we have finally completed this project.

It was a tough job to select a project to complete within time. After observing multiple real life-related ideas, this one was selected since we saw potential in improving this concept to a greater extent.

A special gratitude we give to our project supervisor **Dola Das Madam** (Lecturer, Department of Computer Science and Engineering, KUET), whose contribution in stimulating suggestions and encouragement, helped us to coordinate our project especially in writing this report. It seemed like a dream to complete this huge project in right time. But in that difficult moment, our respected teacher **Dola Das Madam** stood beside us with helpful ideas and guidelines to perfectly complete this task. This motivated us that we can do. Finally, we would like to express our deepest appreciation to all those who provided us the possibility to complete this report.

From-

Md. Tasnimul Khair Tousif, Roll: 1607015

and

And Rahat Alamgir Porosh, Roll: 1607047

Department of Computer Science and Engineering,
Khulna University of Engineering and Technology, Khulna.

Table of Contents

List of Figures:

Chapter 1: Introduction

Digital assistant alternatively referred to as virtual assistant is a computer program or software agent that is designed to help a user to automate his/her daily tasks and performs question-answer based approach to help him/her. This concept can be applied to vehicle drivers. Such software can have safety-related functionalities, customized infotainment, personal health and well-Being, gateway to IoT (e.g. home control) etc. Digital assistant can play significant role in terms of road security, improved interaction between vehicles and drivers and security of the drivers.

This project primarily aims to automate some simple tasks to help vehicle drivers. It can also help the vehicle owners to keep track of their vehicles by means of mobile and send various status of vehicles.

Chapter 2: Software requirements and specifications

2.1 PROJECT OBJECTIVES

The finished project will perform as digital assistant for vehicle drivers. Most of the existing projects like 'Chris', 'Alexa' are closed sourced and high priced. This project aims to provide good functionalities free of cost and will also be open source for other developers to work in for improvement. Such crowd-souring approach is bound to make the project successful. The common features that the project will contain are-

1. Voice based input command and output.
2. Searching while driving to find a location about things.
3. Read news by crawling a website.
4. Simple Q/A based conversation.
5. Face detection of vehicle drivers to identify who is driving.
6. Drowsiness detection of driver to monitor whether they are sleeping or not. If he is sleeping then alert him.
7. Location tracking and sending data to firebase server.
8. Automating tasks like web browsing.

2.2 USER CHARACTERISTICS

The users are mainly of 2 types-

1. Vehicle drivers: Can be of all ages and educational background. The project is designed in such a way that even people with minimum computer skill will be able to access its full functionality. The project uses the advantage of voice base I/O as much as possible because while driving, pressing a key or two is dangerous.
2. Vehicle owners: They will be able to monitor the vehicle's location based on the application that the project includes.

2.3 PERFORMANCE REQUIREMENTS

As the vehicle driving is a critical task, input and output must be as fast as possible. So efficient and well-studied algorithms have to be used. Methods like sleep detection and location tracking has to be kept activated as long as the vehicle is on the move, so use of multithreading techniques is a must. As the finished project is supposed to be run on a low powered, critically resourced computing device, efficiency and memory usage of hardware will be as minimum as possible.

Chapter 3: Project plan

3.1 TECHNOLOGY USAGE

Python is used as the main programming language of the project because-
1. It is a preferred language many AI and robotics enthusiasts use, this may come in handy while others are contributing on the open source repository,
2. Many algorithms on Computer Aision, Voice Based Computing are implemented on python which may be used to the implementation if needed.
3. Good for rapid prototyping of the projects as less runtime errors are faced compared to other language.
4. Has hardware support for IOT based devices.

Other libraries and modules that will be used are-
1. OpenCV: To implement image processing features.
2. Firebase: To implement data sending to server.
3. Google Speech Recognition: For recognizing user's speech to turn it into appropriate command.
4. Pyttsx3: Python text to speech module.
5. threading, signal: For concurrent execution of methods.
6. Dlib: Facial landmarks.
7. Imutils: Facial landmarks detection.
8. Scipy, Numpy: Computer vision related functions.
9. Webbrowser: Automation of web browsing.
10. Face_recognition: Recognition of driver's face, count number of passengers based on faces.
11. Android studio: To build a simple app to display vehicle location to the owner.

3.2 DEVELOPMENT ENVIRONMENT

Developed in windows 10 operating system, will be shifted to Raspberry-pi based hardware for real-time use, as a laptop is not feasible to fit inside a car. Associated application will be developed for Android platform.

3.3 PROJECT STRUCTURE

The project is developed into several files to keep the modularity and discreet nature of various programs. They are all linked via python module. A main.py file is used to call all the modules whenever needed.

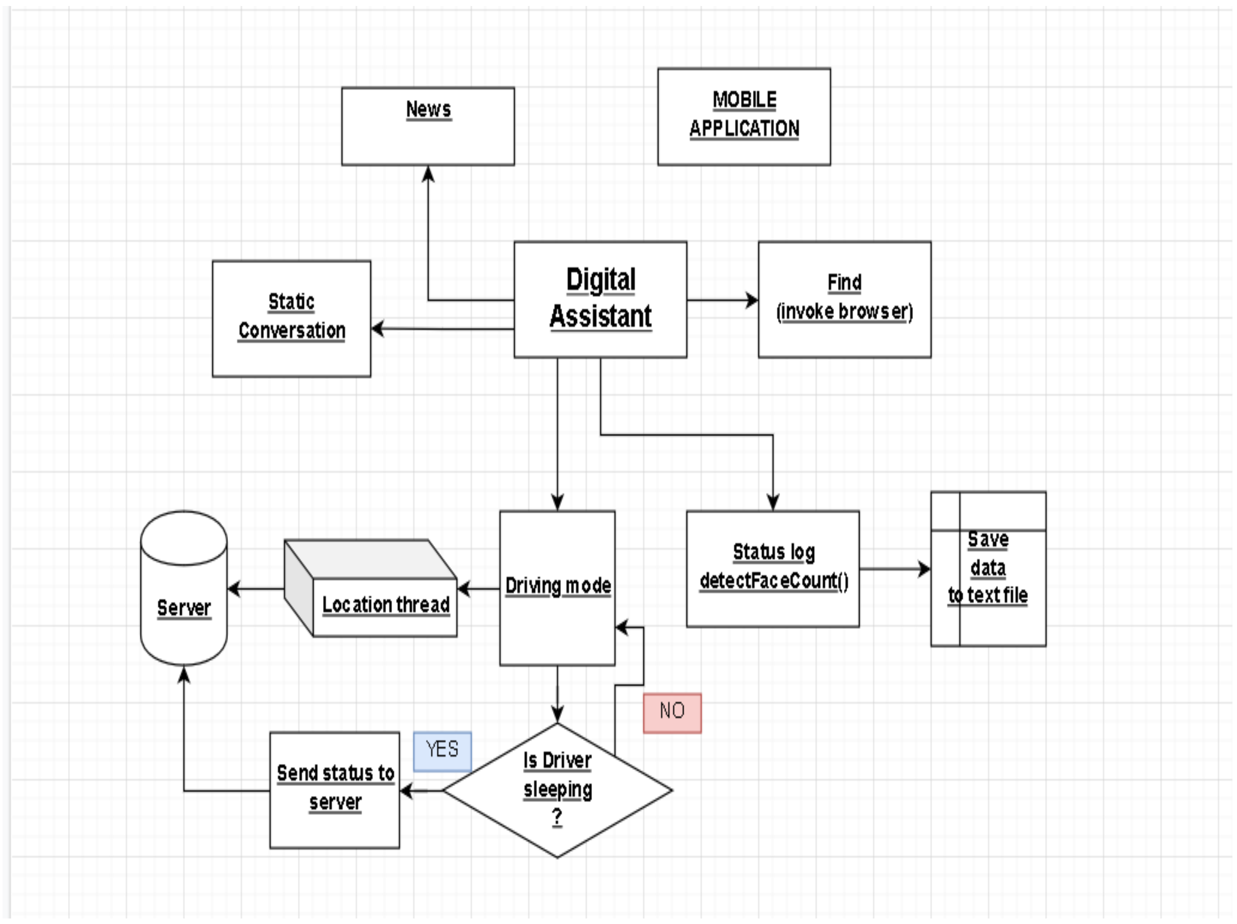
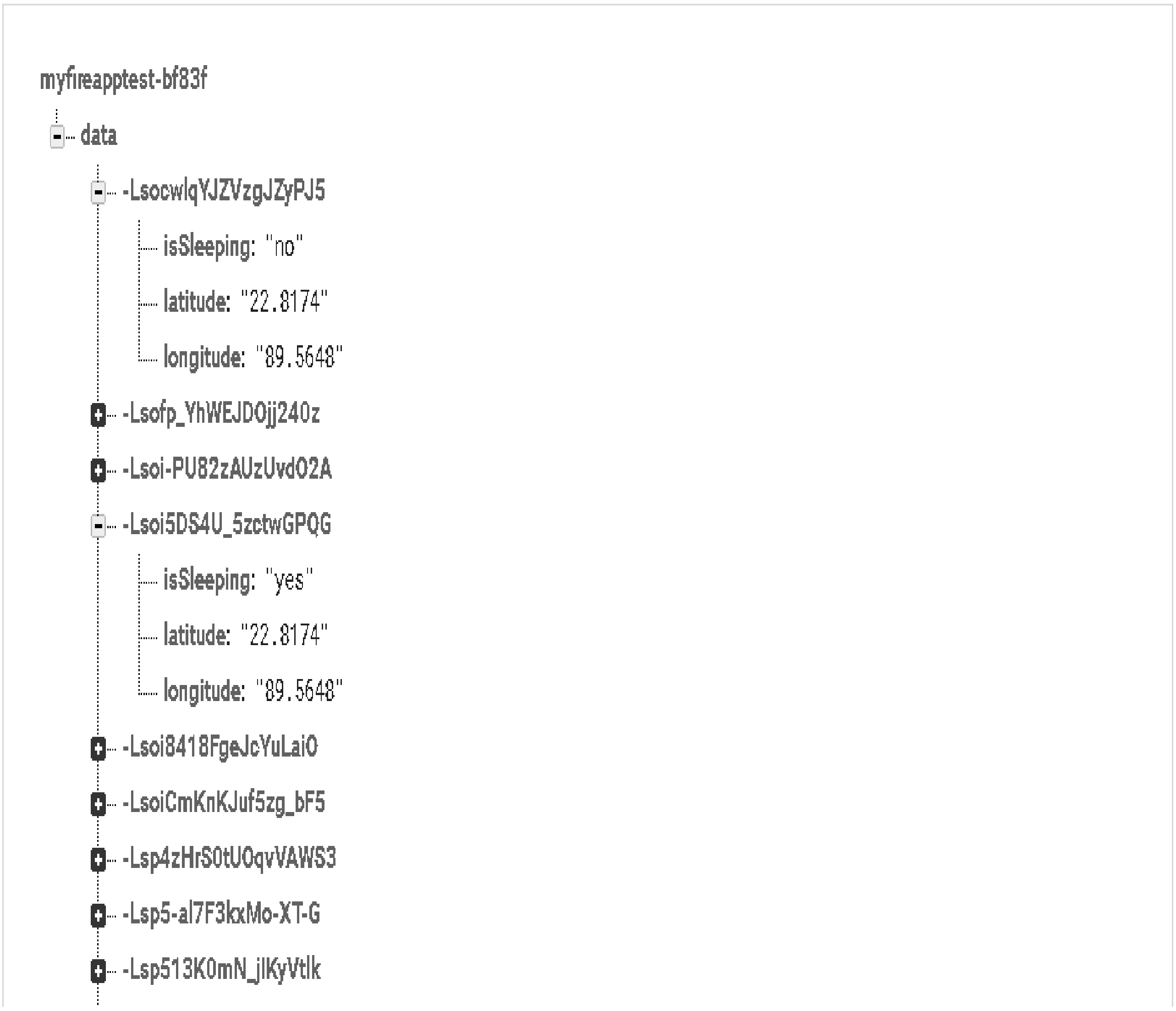
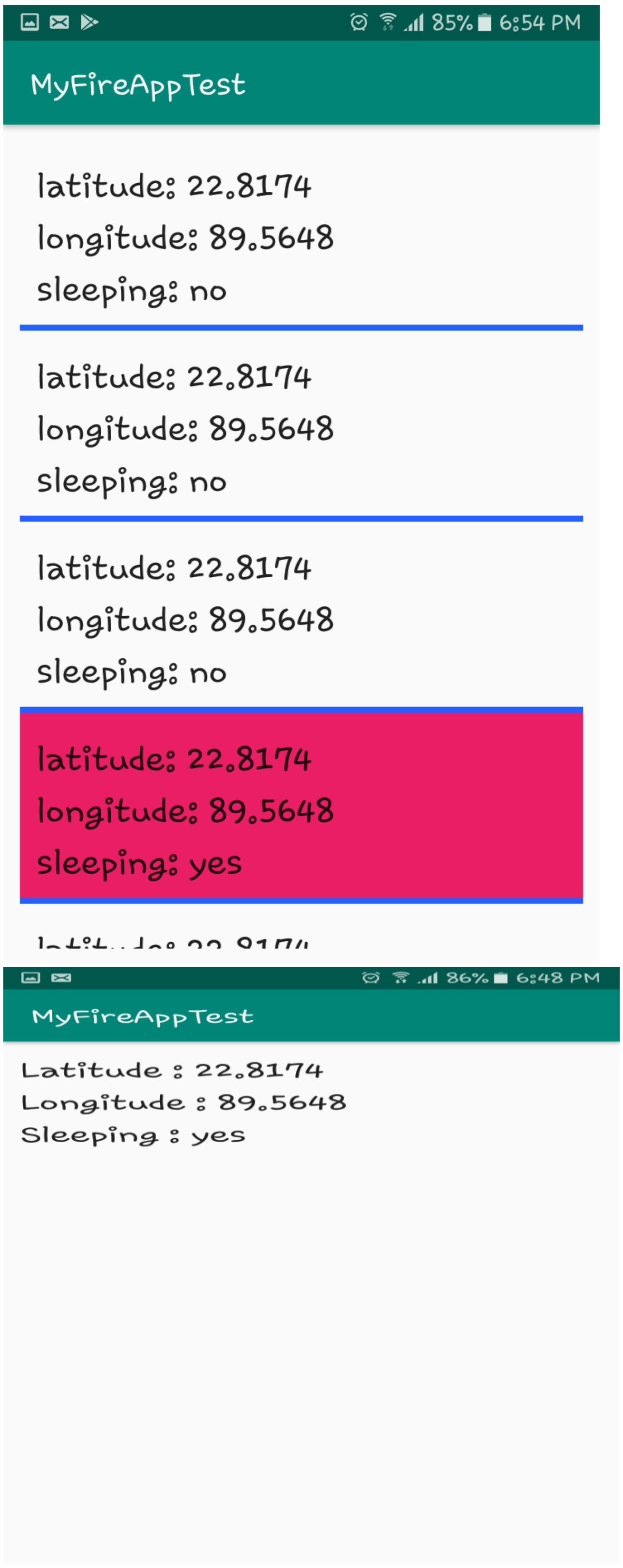


Fig. 3.3.1: Process Flow diagram



```
myfireapptest-bf83f
  data
    -LsocwlqYJZVzgJZyPJ5
      isSleeping: "no"
      latitude: "22.8174"
      longitude: "89.5648"
    -Lsofp_YhWEJDOjj240z
    -Lsoi-PU82zAUzUvdO2A
    -Lsoi5DS4U_5zctwGPQG
      isSleeping: "yes"
      latitude: "22.8174"
      longitude: "89.5648"
    -Lsoi8418FgeJcYuLaiO
    -LsoiCmKnKJuf5zg_bF5
    -Lsp4zHrS0tUOqvVAWS3
    -Lsp5-al7F3kxMo-XT-G
    -Lsp513K0mN_jlKyVtlk
```

Fig. 3.3.2: Firebase real-time database



latitude: 22.8174
longitude: 89.5648
sleeping: no

latitude: 22.8174
longitude: 89.5648
sleeping: no

latitude: 22.8174
longitude: 89.5648
sleeping: no

latitude: 22.8174
longitude: 89.5648
sleeping: yes

Latitude : 22.8174
Longitude : 89.5648
Sleeping : yes

Fig. 3.3.3: Android application presenting data

Chapter 4: Implementation

4.1 VOICE BASED INPUT OUTPUT:

Google speech recognition is used for voice based input command. Google Cloud Speech-to-Text enables developers to convert audio to text by applying powerful neural network models in an easy-to-use API. The API recognizes 120 languages and variants to support your global user base. You can enable voice command-and-control, transcribe audio from call centers, and more. It can process real-time streaming or prerecorded audio, using Google's machine learning technology. It captures audio and recognizes speech and turns it to text-based input which are feed into software. It is instantiated when the project starts and runs till the end. A get_audio() method utilizes Google speech recognition to take input.

Python text to speech module is used for voice output. Python text to speech *or* pyttsx is a Python package supporting common text-to-speech engines on Mac OS X, Windows, and Linux. A speak() method uses this module.

4.2 DRIVING MODE

It is the most crucial part of the project. Several things are invoked when initiating this method-

1. Check if the driver is the intended one or not. If driver is not matched don't allow to drive.
2. If driver matches the software invokes driving mode. It has 2 threads running concurrently-
    a. Monitor if the driver is drowsy or not,
    b. Send location and sleeping log to the server.

Now the theories behind the face recognition, drowsiness detection scheme and location tracking will be discussed-

4.2.1 FACE RECOGNITION:

This is a python module that can recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. It is built using dlib's state-of-the-art face recognition with deep learning. The model has an accuracy of 99.38% on the " Labeled faces in the wild" benchmark."
In order to recognize driver's face, first the software takes a picture from the web camera. Then it invokes the **recognizeDriverFace()** method is invoked which iterates through the list of registered drivers and invokes another method **faceRecognitionMethod(name)** for each name. If a face is recognized the method **faceRecognitionMethod(name)** returns true. And such the loop is terminated and driver is recognized. The face recognition is done comparing the saved image with the captured image.
The face recognition module also does the checking of face count in the passenger seat. To do that the **detectFaceCount()** method is invoked. It uses **face_locations(image)** method of the library to detect faces and results are stored in a list. The number of faces is the length of the list.

4.2.2 DROWSINESS DETECTION:

Facial landmarks are used to localize and represent salient regions of the face, such as: Eyes, Eyebrows, Nose, Mouth, Jawline. Facial landmarks have been successfully applied to face alignment, head pose estimation, face swapping, blink detection and much more. Detecting facial landmarks is a *subset* of the *shape prediction* problem. Given an input image (and normally an ROI that specifies the object of interest), a shape predictor attempts to localize key points of interest along the shape. But this project's region of interest is Eyes, which will be used to continue the processing further.

Detecting facial landmarks is therefore a two step process:

- **Step #1:** Localize the face in the image.
- **Step #2:** Detect the key facial structures on the face ROI.

The facial landmark detector included in the dlib library is an implementation of the *One Millisecond Face Alignment with an Ensemble of Regression Trees* paper by Kazemi and Sullivan (2014). This method starts by using:

1. A training set of labeled facial landmarks on an image. These images are *manually labeled*, specifying **specific** *(x, y)*-coordinates of regions surrounding each facial structure.
2. *Priors*, of more specifically, the *probability on distance* between pairs of input pixels.

Given this training data, an ensemble of regression trees are trained to estimate the facial landmark positions directly from the *pixel intensities themselves* (i.e., no "feature extraction" is taking place).The end result is a facial landmark detector that can be used to **detect facial landmarks in *real-time*** with **high quality predictions**.

The pre-trained facial landmark detector inside the dlib library is used to estimate the location of ***68 (x, y)-coordinates*** that map to facial structures on the face. The indexes of the 68 coordinates can be visualized on the image below:

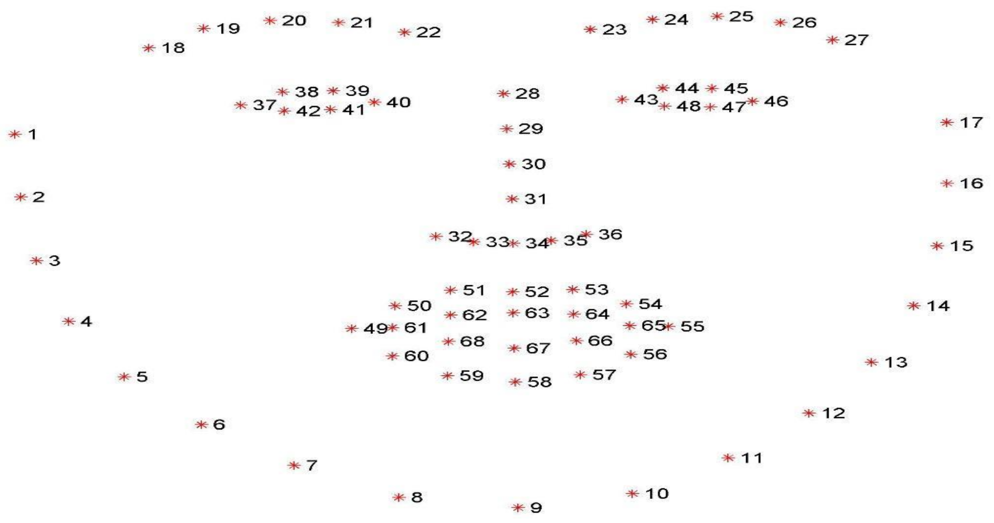


Figure 4.2.2.1: Visualizing the 68 facial landmark coordinates from the iBUG 300-W dataset.

These annotations are part of the 68 point iBUG 300-W dataset which the dlib facial landmark predictor was trained on. The dlib fraework can be leveraged to train a shape predictor on the input training data — this is useful if the developer would like to train facial landmark detectors or custom shape predictors of his/her own. Examining the image, it can be seen that facial regions can be accessed via simple Python indexing (assuming zero-indexing with Python since the image above is one-indexed):

- The *mouth* can be accessed through points *[48, 68]*.
- The *right eyebrow* through points *[17, 22]*.
- The *left eyebrow* through points *[22, 27]*.
- The *right eye* using *[36, 42]*.
- The *left eye* with *[42, 48]*.
- The *nose* using *[27, 35]*.
- And the jaw via *[0, 17]*.

These mappings are encoded inside the FACIAL_LANDMARKS_IDXS dictionary inside face_utils of the imutils library. To *visualize* each of these facial landmarks and overlay the results on an input image, we'll need the visualize_facial_landmarks function, which is already included in the imutils library .

The imultils framework has function called shape_to_np which The dlib face landmark detector will return a shape object containing the 68 *(x, y)*-coordinates of the facial landmark regions. Using the shape_to_np function, it can be converted to to a NumPy array, allowing it to "play nicer" with our Python code. The project uses dlib's pre-trained face detector based on a modification to the standard Histogram of Oriented Gradients + Linear SVM method for object detection.

Eye blink detection: To build our blink detector, we'll be computing a metric called the *eye aspect ratio* (EAR), introduced by Soukupová and Čech in their 2016 paper, *Real-Time Eye Blink Detection Using Facial Landmarks*.

Unlike traditional image processing methods for computing blinks which typically involve some combination of:

1. Eye localization.
2. Thresholding to find the whites of the eyes.
3. Determining if the "white" region of the eyes disappears for a period of time (indicating a blink).

The eye aspect ratio is instead a *much more elegant solution* that involves a *very simple calculation* based on the ratio of distances between facial landmarks of the eyes.

This method for eye blink detection is fast, efficient, and easy to implement.

Understanding the "eye aspect ratio" (EAR):

In terms of blink detection, only two sets of facial structures are needed — *the eyes.*

Each eye is represented by 6 *(x, y)*-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region.

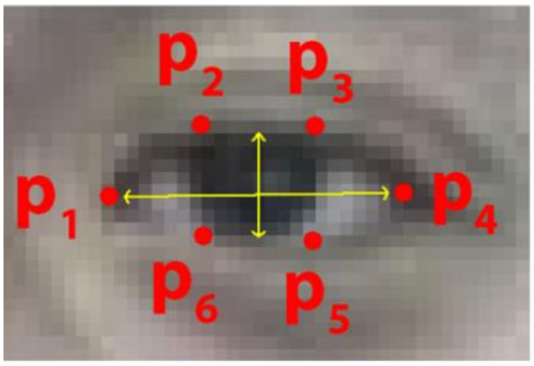


Fig. 4.2.2.2: The 6 facial landmarks associated with the eye.

Based on this image, we should take away on key point: **There is a relation between the *width* and the *height* of these coordinates.**

Based on the work by Soukupová and Čech in their 2016 paper, *Real-Time Eye Blink Detection using Facial Landmarks*, we can then derive an equation that reflects this relation called the *eye aspect ratio* (EAR):

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

Where *p1, ..., p6* are 2D facial landmark locations.

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only *one* set of horizontal points but *two* sets of vertical points.

The eye aspect ratio is approximately constant while the eye is open, but will rapidly fall to zero when a blink is taking place.

Using this simple equation, *image processing techniques* can be avoided and simply rely on the *ratio of eye landmark distances* to determine if a person is blinking. To make this clearer, consider the following figure from Soukupová and Čech:

Drowsiness Detection algorithm can be straightforwardly written like this-

First, we'll setup a camera that monitors a stream for faces:

Next, if a face is found, we apply facial landmark detection and extract the eye regions:.

Then, now that we have the eye regions, we can compute the eye aspect ratio to determine if the eyes are closed.

Finally, if the eye aspect ratio indicates that the eyes have been closed for a sufficiently long enough amount of time, we'll sound an alarm to wake up the driver:

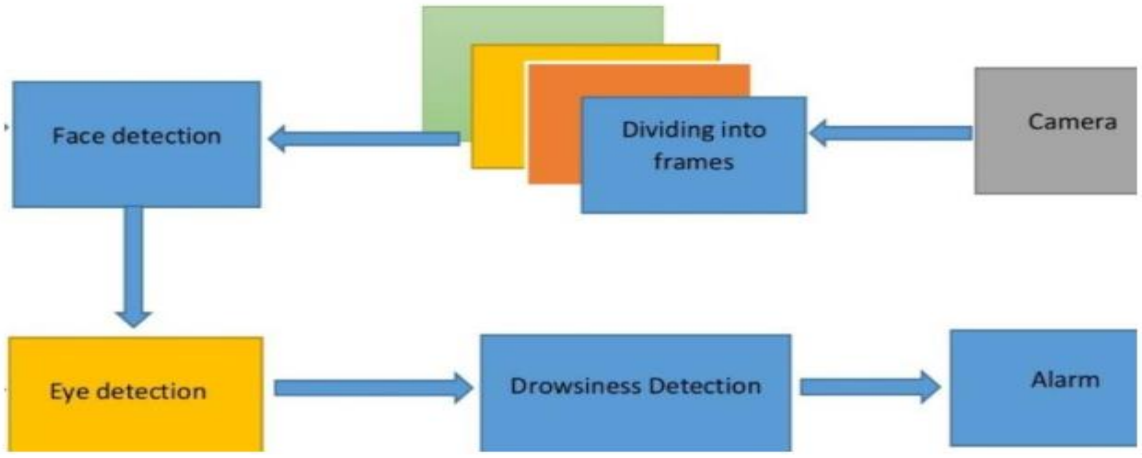


Fig. 4.2.2.3: Process diagram of drowsiness detection system.

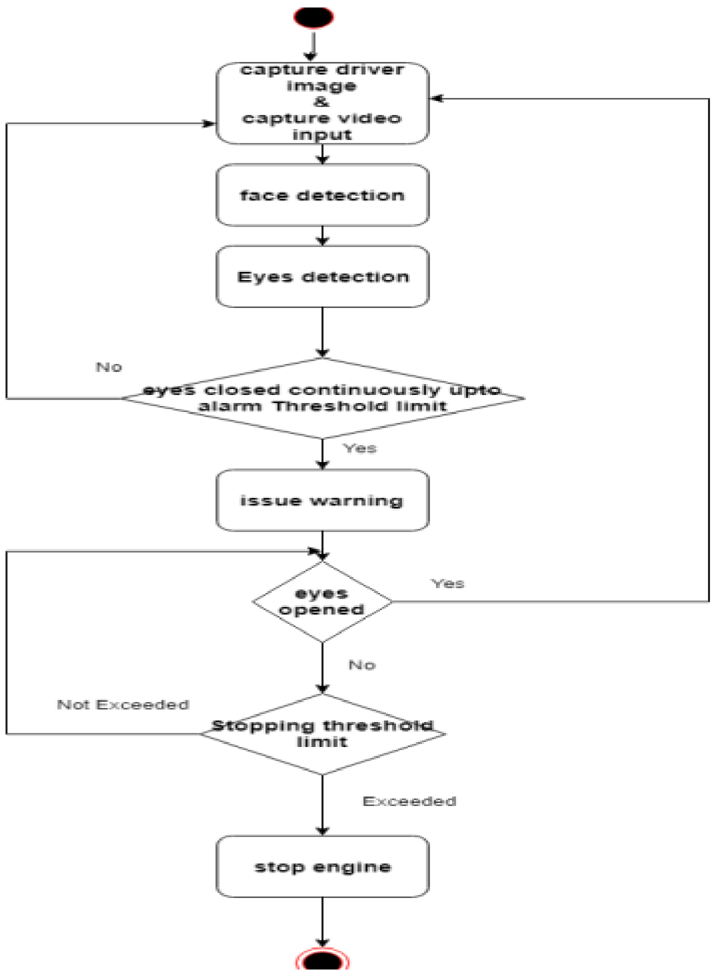


Fig. 4.2.2.4: Flow chart for the drowsiness detecting system

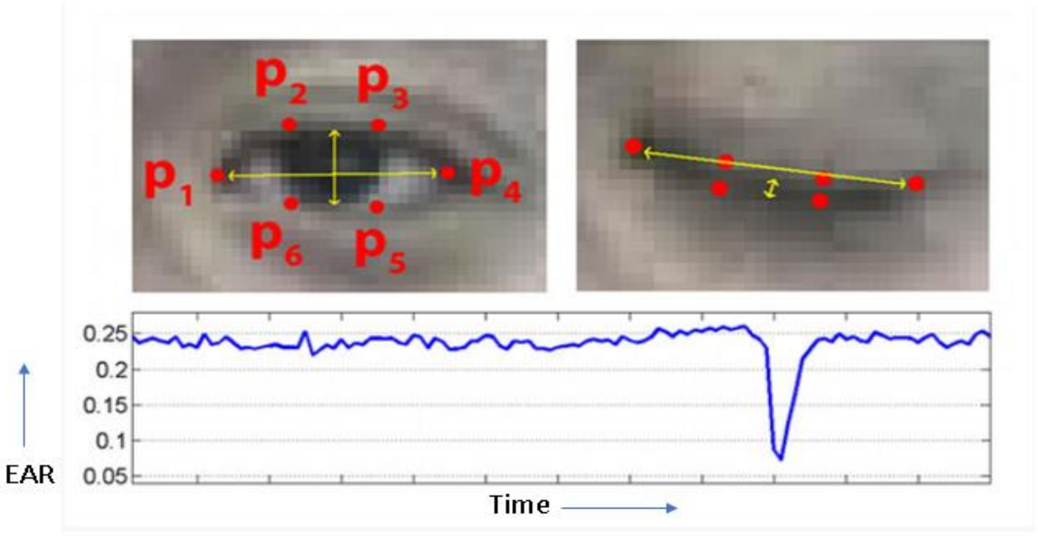


Fig. 4.2.2.5: *Top-left:* A visualization of eye landmarks when then the eye is open. *Top-right:* Eye landmarks when the eye is closed. *Bottom:* Plotting the eye aspect ratio over time. The dip in the eye aspect ratio indicates a blink (Figure 1 of Soukupová and Čech).

On the *top-left* we have an eye that is fully open — the eye aspect ratio here would be large(r) and relatively constant over time. However, once the person blinks (*top-right*) the eye aspect ratio decreases dramatically, approaching zero.

The *bottom* figure plots a graph of the eye aspect ratio over time for a video clip. As we can see, the eye aspect ratio is constant, then rapidly drops close to zero, then increases again, indicating a single blink has taken place.

Now the implementation details will be discussed. If eye aspect ratio drops below or equal 0.3, a blink is indicated. Now if this state sustains for 48 consecutive frames, an alarm sound will be played to alert the driver. Also, location and sleep status will be sent to server to alert the vehicle owner.

The dlib library ships with a face detector along with a predictor — both are instantiated in the code block. The facial landmarks produced by dlib are an indexable list. Therefore, to extract the eye regions only appropriate array indices are need to be known. After instantiating the video stream, looping is done over the continuous frames to monitor the eye status.

Now, in each iteration a frame is read, resized to 450 pixels and converted into grayscale. Now dlib's face detector is applied to detect the face(s) in the captured frame. After finding the face, facial landmark detection scheme is instantiated to localize the important regions of the face, in this case the eyes. Then the returned facial landmark's (x,y) coordinates are transferred into a numpy array. Next array slicing is done to find the left eye and right eye coordinates. Finally, these coordinates are processed to find the eye aspect ratio (EAR). Both left and right eye's EAR are taken and average value is calculated. This is because Soukupová and Čech recommend averaging both eye aspect ratios together to obtain a better estimation.

Then the eye regions can be visualized on the frame that the loop is iterating on. This is done by drawing convex-hull around eye regions. Then contours are drawn around eyes of color green.

Next comes the final part of the code, where checking of drowsiness is done. If the eye aspect ratio is equal or less than 0.3, then a counter variable is incremented. After 48 consecutive frames the counter is greater or equal than 48 then alarm is triggered.

Finally, the counter is set up again and program keeps running. After terminating all the resources are cleared.

4.2.3 LOCATION TRACKING:

The location tracking method is implemented by the json request from **https://get.geojs.io/v1/ip.json** . After receiving the json response it is parsed and location data is extracted. Finally, it is sent to server storage. Firebase real time database is used as storage. The location tracking function is invoked at the same time as driving mode, it is achieved by incorporating the use of multithreading. Python's "Thread" library is used to implement multithreading.

4.3 ANDROID APPLICATION DEVELOPMENT:

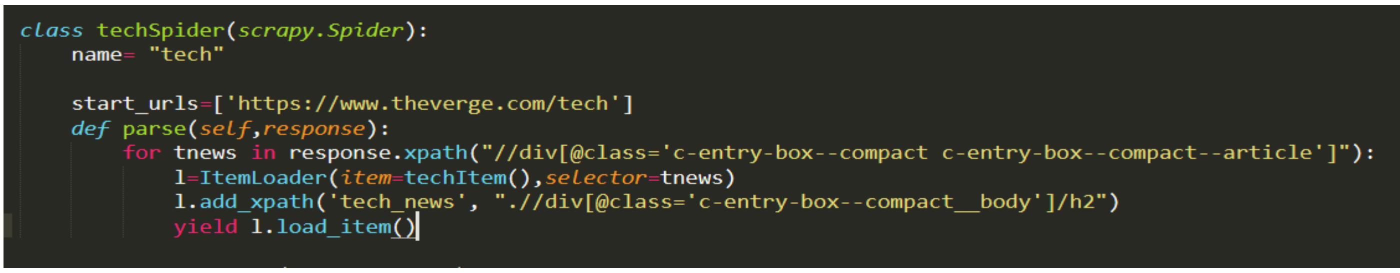
The application is pretty much straightforward with a simple UI. It's only purpose is to fetch data from the firebase real time data base and populate the list view with that data. The UI has the feature to save dummy data to the server and also to load the existing data to the server. The data is loaded from the digital assistant application. A custom adapter class takes the model class and makes a list, finally it uses it to populate the UI. The model class has latitude, longitude, sleeping status and position. The single item view shows them in a simple fashion.

4.4 WEB CRAWLING:

It can provide the user the necessary news that he needs. The types of news it can provide are mainly divided in four types. They are:

1. Tech News
2. World News
3. Cricket News
4. Football News

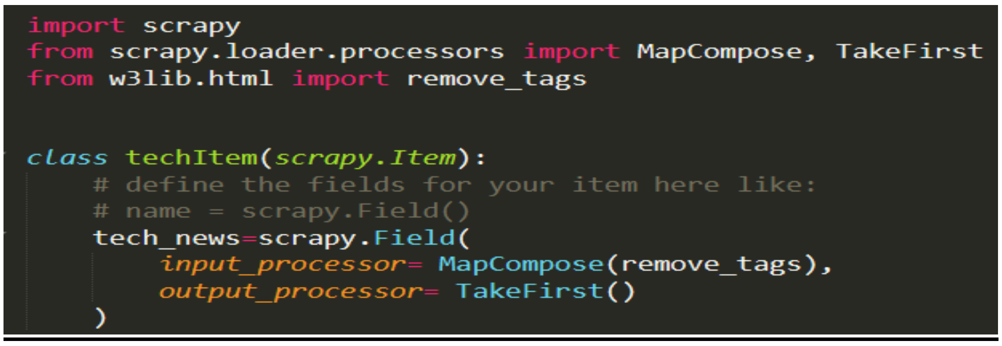
It scraps from different websites depending on the chosen type of news. For tech news 'https://www.theverge.com/tech', for world news 'https://www.reuters.com/news/archive/worldNews', for cricket news 'https://www.espn.in/cricket/' and for football news 'https://www.espn.in/football/' is used as the source. It first crawls the websites and scraps the headline of the pages and save them as a '.csv' file. To scrap the pages spiders are created for each site.



```
class techSpider(scrapy.Spider):
    name= "tech"

    start_urls=['https://www.theverge.com/tech']
    def parse(self,response):
        for tnews in response.xpath("//div[@class='c-entry-box--compact c-entry-box--compact--article']"):
            l=ItemLoader(item=techItem(),selector=tnews)
            l.add_xpath('tech_news', ".//div[@class='c-entry-box--compact__body']/h2")
            yield l.load_item()
```

At first a class is created which inherits scrapy.Spider class which is the base spider class. Then it is named according to the type of news. Then start_urls variable is declared. It tells the spider from which site the spider will start to crawl. Then the parse function is defined. Then the xpath method is called. The xpath method takes the xpath expression as string and returns a selector object. The xpath expression is written in such a way that it can select the portion of the site which contains the headline. To know the structure of the portion the site is inspected using the developer option of the browser Chrome**.** To collect scraped data Item objects are used. Item objects are containers for collecting scraped data. Item loaders are used to populate those item objects.



```
import scrapy
from scrapy.loader.processors import MapCompose, TakeFirst
from w3lib.html import remove_tags


class techItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    tech_news=scrapy.Field(
        input_processor= MapCompose(remove_tags),
        output_processor= TakeFirst()
    )
```

An Item Loader contains one input processor and one output processor for each (item) field. The input processor processes the extracted data as soon as it's received (through the add_xpath(), add_css() or add_value() methods) and the result of the input processor is collected and kept inside the ItemLoader. If no modification is made into these ItemLoaders then the html tags remain in the output. MapCompose method is used to further process the data. MapCompose takes a method as argument and applies it to the scraped data. An utility method remove_tags is passed as the argument of MapCompose. The utility class removes the html tags from the scraped data. The TakeFirst method is called in the output_processor which returns the extracted data as a string.After the necessary data are scraped from the desired site the speak method described in section 4.1 is used and the program read aloud the scraped data. That is to say the news headline that the user wanted to know.

4.5 Web Search Using Voice

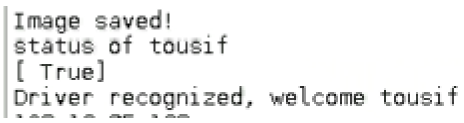
It can search in the default web browser by taking voice input. The input is taken as discussed in section 4.1. If "go to" is in the input then it goes to the specified page. If not it searches in google the query which the user provided as input.

Chapter 5: Runtime simulation and output

The project is implemented and run in the “spyder”(a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts) . Voice input is taken from microphone and parsed to text. Then this text is matched and proper feed back is generated. Responses are generated both in text based format and audio. Below some ideal cases are written-

5.1: Input: Driving mode.

Action: Driver is checked by face recognition, if matched a message says that driver is confirmed.

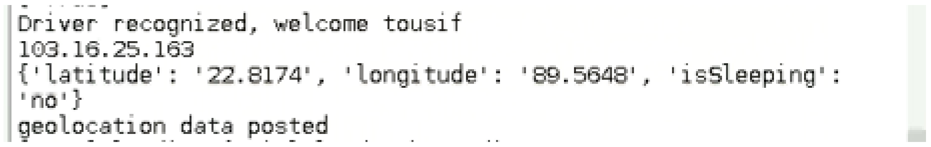


```
Image saved!
status of tousif
[ True]
Driver recognized, welcome tousif
```

Fig. 5.1.1: Driver verification

Next he/she is taken to the initiation of driving mode. This basically invokes drowsiness detection method and location tracking method. Also in a parallel thread location data and sleep status is sent to server. Below some text output are stated:

Driver not sleeping:



```
Driver recognized, welcome tousif
103.16.25.163
{'latitude': '22.8174', 'longitude': '89.5648', 'isSleeping':
'no'}
geolocation data posted
```

Fig. 5.1.2: Sending data to server.

<u>Driver sleeping:</u>

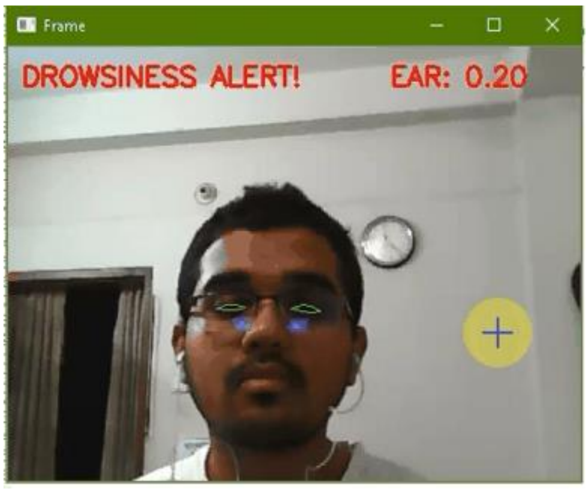
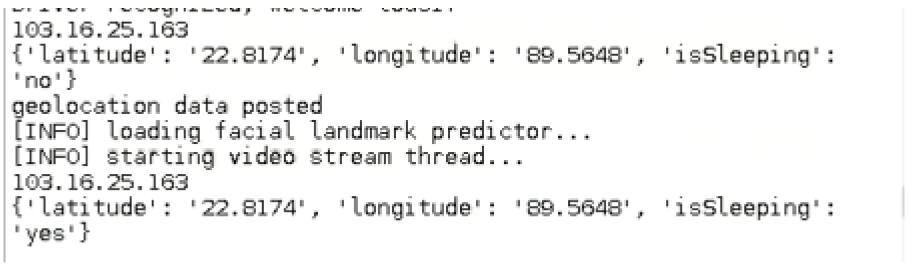


Fig. 5.1.3: Driver sleeping

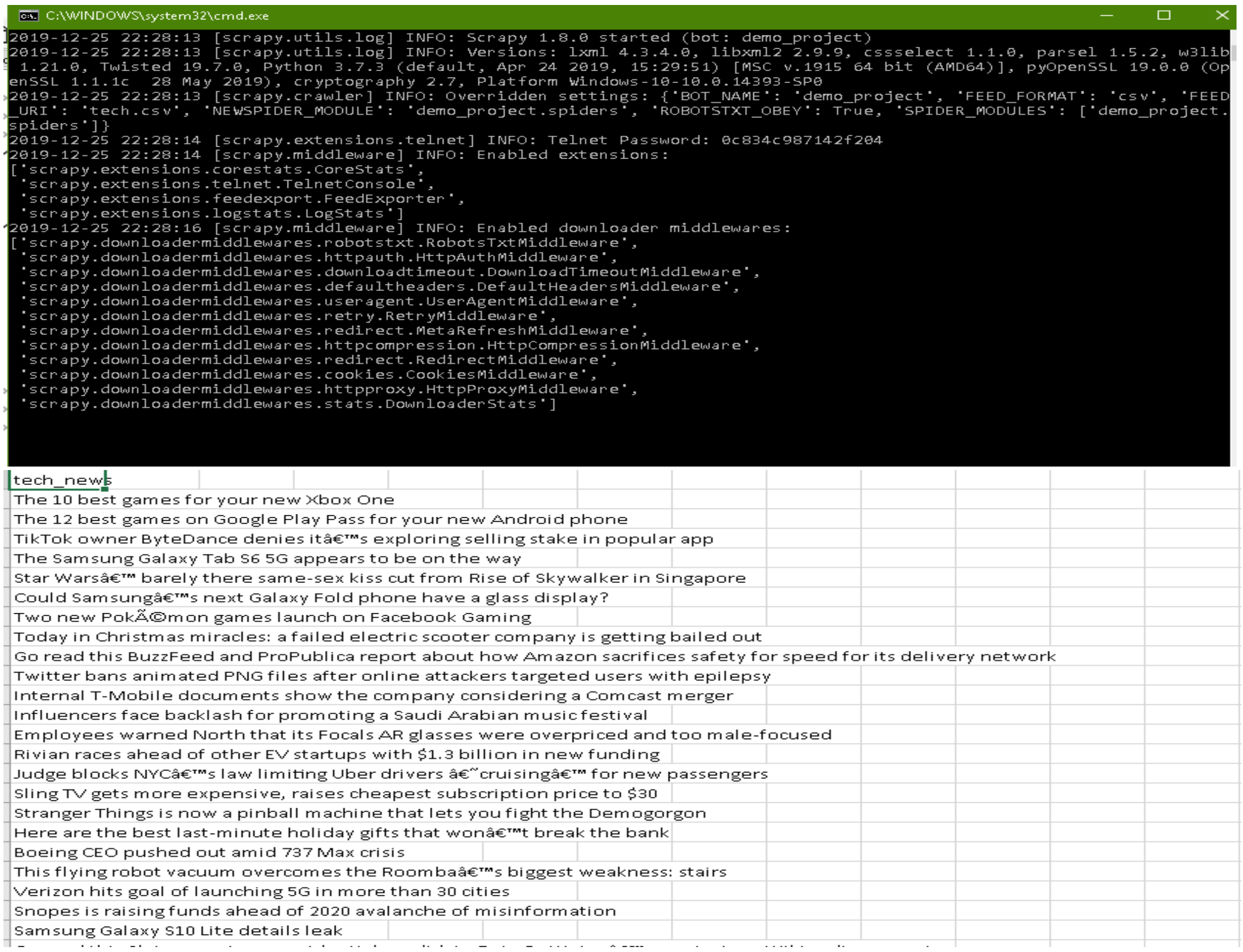


```
[illegible]
103.16.25.163
{'latitude': '22.8174', 'longitude': '89.5648', 'isSleeping':
'no'}
geolocation data posted
[INFO] loading facial landmark predictor...
[INFO] starting video stream thread...
103.16.25.163
{'latitude': '22.8174', 'longitude': '89.5648', 'isSleeping':
'yes'}
```

Fig. 5.1.4: Sending sleep status log and location to server

<u>5.2: Input:</u> News

<u>Action</u>: A prompt is given to the user which news to be heard. After choosing, associating news websites are crawled to find the news and store inside the spreadsheet. Finally it is read aloud to the user.



```
2019-12-25 22:28:13 [scrapy.utils.log] INFO: Scrapy 1.8.0 started (bot: demo_project)
2019-12-25 22:28:13 [scrapy.utils.log] INFO: Versions: lxml 4.3.4.0, libxml2 2.9.9, cssselect 1.1.0, parsel 1.5.2, w3lib
 1.21.0, Twisted 19.7.0, Python 3.7.3 (default, Apr 24 2019, 15:29:51) [MSC v.1915 64 bit (AMD64)], pyOpenSSL 19.0.0 (Op
enSSL 1.1.1c  28 May 2019), cryptography 2.7, Platform Windows-10-10.0.14393-SP0
2019-12-25 22:28:13 [scrapy.crawler] INFO: Overridden settings: {'BOT_NAME': 'demo_project', 'FEED_FORMAT': 'csv', 'FEED
_URI': 'tech.csv', 'NEWSPIDER_MODULE': 'demo_project.spiders', 'ROBOTSTXT_OBEY': True, 'SPIDER_MODULES': ['demo_project.
spiders']}
2019-12-25 22:28:14 [scrapy.extensions.telnet] INFO: Telnet Password: 0c834c987142f204
2019-12-25 22:28:14 [scrapy.middleware] INFO: Enabled extensions:
['scrapy.extensions.corestats.CoreStats',
 'scrapy.extensions.telnet.TelnetConsole',
 'scrapy.extensions.feedexport.FeedExporter',
 'scrapy.extensions.logstats.LogStats']
2019-12-25 22:28:16 [scrapy.middleware] INFO: Enabled downloader middlewares:
['scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware',
 'scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',
 'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',
 'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',
 'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',
 'scrapy.downloadermiddlewares.retry.RetryMiddleware',
 'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',
 'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',
 'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',
 'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',
 'scrapy.downloadermiddlewares.httpproxy.HttpProxyMiddleware',
 'scrapy.downloadermiddlewares.stats.DownloaderStats']
```

tech_news
The 10 best games for your new Xbox One
The 12 best games on Google Play Pass for your new Android phone
TikTok owner ByteDance denies itâ€™s exploring selling stake in popular app
The Samsung Galaxy Tab S6 5G appears to be on the way
Star Warsâ€™ barely there same-sex kiss cut from Rise of Skywalker in Singapore
Could Samsungâ€™s next Galaxy Fold phone have a glass display?
Two new PokÃ©mon games launch on Facebook Gaming
Today in Christmas miracles: a failed electric scooter company is getting bailed out
Go read this BuzzFeed and ProPublica report about how Amazon sacrifices safety for speed for its delivery network
Twitter bans animated PNG files after online attackers targeted users with epilepsy
Internal T-Mobile documents show the company considering a Comcast merger
Influencers face backlash for promoting a Saudi Arabian music festival
Employees warned North that its Focals AR glasses were overpriced and too male-focused
Rivian races ahead of other EV startups with $1.3 billion in new funding
Judge blocks NYCâ€™s law limiting Uber drivers â€˜cruisingâ€™ for new passengers
Sling TV gets more expensive, raises cheapest subscription price to $30
Stranger Things is now a pinball machine that lets you fight the Demogorgon
Here are the best last-minute holiday gifts that wonâ€™t break the bank
Boeing CEO pushed out amid 737 Max crisis
This flying robot vacuum overcomes the Roombaâ€™s biggest weakness: stairs
Verizon hits goal of launching 5G in more than 30 cities
Snopes is raising funds ahead of 2020 avalanche of misinformation
Samsung Galaxy S10 Lite details leak

Fig 5.2.1: Website crawling and fetching news data

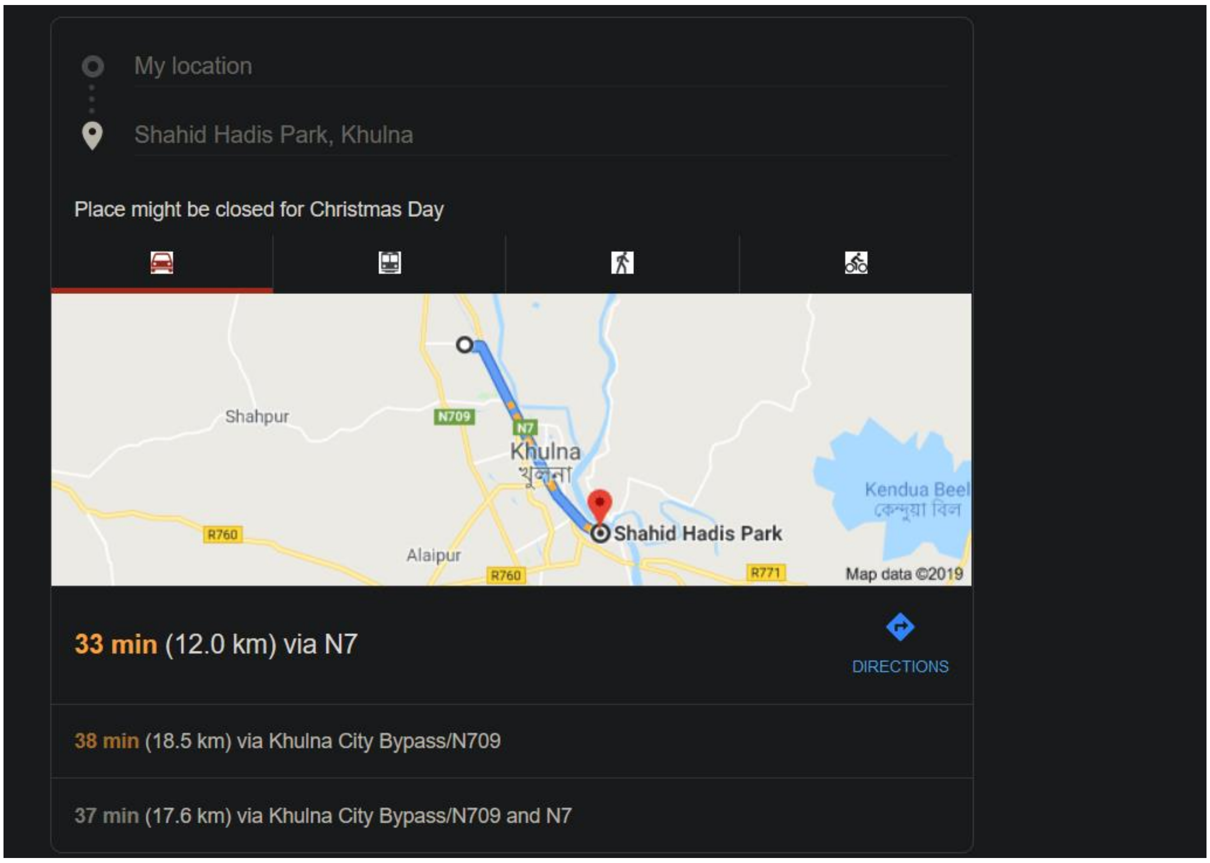
<u>5.3: Input:</u> Status log

<u>Action</u>: A face recognition method is invoked to see how many numbers of passengers are present in the car.

<u>5.4: Input:</u> Search

<u>Action</u>: Browser is invoked and associated things are searched.

```
listening...
Stop
You:  navigate to Hadis Park
```


My location

Shahid Hadis Park, Khulna

Place might be closed for Christmas Day

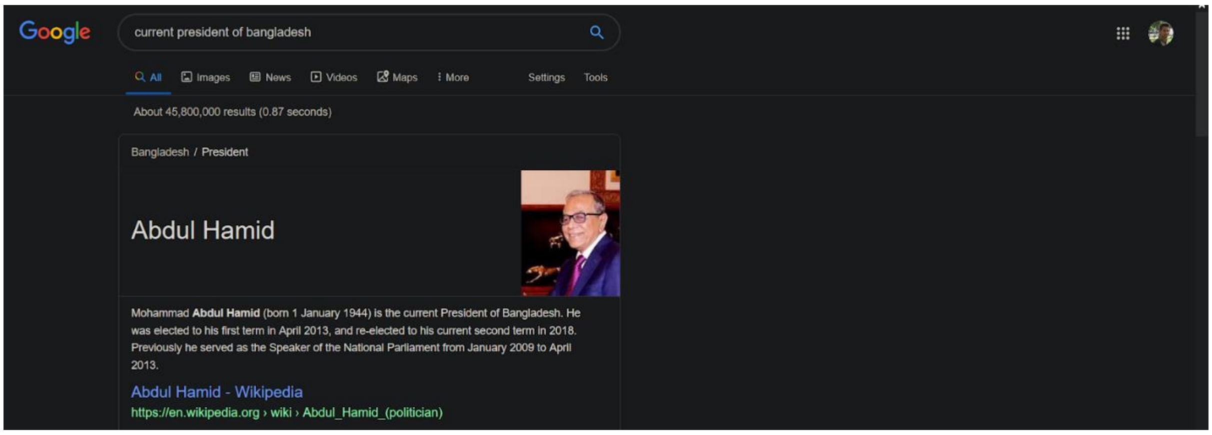
33 min (12.0 km) via N7

DIRECTIONS

38 min (18.5 km) via Khulna City Bypass/N709

37 min (17.6 km) via Khulna City Bypass/N709 and N7

```
You:  find
listening...
Stop
You:  current President of Bangladesh
```


Google current president of bangladesh

All Images News Videos Maps More Settings Tools

About 45,800,000 results (0.87 seconds)

Bangladesh / President

Abdul Hamid

Mohammad **Abdul Hamid** (born 1 January 1944) is the current President of Bangladesh. He was elected to his first term in April 2013, and re-elected to his current second term in 2018. Previously he served as the Speaker of the National Parliament from January 2009 to April 2013.

Abdul Hamid - Wikipedia
https://en.wikipedia.org › wiki › Abdul_Hamid_(politician)

Fig 5.4.1: Usage of web search for navigation and information

Chapter 6: Future work

In future the project can be ported to Raspberry Pi, because it is not practical and convenient to carry a laptop each and every time one drives. As Raspberry Pi has a linux based Operating System and the project is developed on Windows, this will be a challenge. Also the response time of sleep detection algorithm is not up to the mark for real life cases. Raspberry Pi has very limited hardware, the software will run really slow. So, optimizations on current algorithm and even better algorithms need to be used. The drowsiness detection system uses the eye aspect ratio as a quantitative metric to determine if a person has blinked in a video stream.

However, due to noise in a video stream, subpar facial landmark detections, or fast changes in viewing angle, a simple threshold on the eye aspect ratio could produce a false-positive detection, reporting that a blink had taken place when in reality the person had not blinked.

To make the blink detector method more robust to these challenges, Soukupová and Čech recommend:

1. Computing the eye aspect ratio for the *N*-th frame, along with the eye aspect ratios for *N – 6* and *N + 6* frames, then concatenating these eye aspect ratios to form a 13 dimensional feature vector.
2. Training a Support Vector Machine (SVM) on these feature vectors.

Soukupová and Čech report that the combination of the temporal-based feature vector and SVM classifier helps reduce false-positive blink detections and improves the overall accuracy of the blink detector.

An effective conversational UI aka chatbot can be added which will interface with the driver and leverage all the features through it.

The android app design can be more robust, user friendly and include more functionalities.

A tracking system would not be too much out of the context and can be helpful to the driver for tracking unknown areas.

The software is designed in a static way, by integrating software engineering principles it can be dynamic, user-friendly and more realistic in nature.

Chapter 7: References

1. ***https://en.wikipedia.org/wiki/Virtual_assistant***
2. ***https://en.wikipedia.org/wiki/Siri***
3. ***https://en.wikipedia.org/wiki/Bixby_(virtual_assistant)***
4. ***https://en.wikipedia.org/wiki/Google_Assistant***
5. ***https://en.wikipedia.org/wiki/Amazon_Alexa***
6. ***https://en.wikipedia.org/wiki/Interactive_voice_response#Developments***
7. ***https://mycroft.ai/***
8. ***https://jasperproject.github.io/***
9. ***https://pypi.org/project/face_recognition/***
10. ***https://dspace.cvut.cz/bitstream/handle/10467/64839/F3-DP-2016-Soukupova-Tereza-SOUKUPOVA_DP_2016.pdf***
11. *One Millisecond Face Alignment with an Ensemble of Regression Trees*
12. http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html
13. iBUG 300-W dataset
14. Histogram of Oriented Gradients + Linear SVM method
15. *Real-Time Eye Blink Detection Using Facial Landmarks*.

16. **Associated courses:**
    - **Object Oriented Programming**
    - **Data Structures and Algorithm**
    - **Advanced Programming**
    - **Peripherals and Interfacing**
    - **Mobile Computing**
    - **Image Processing and Computer Vision**