Capstone Project

**Drowsy Driving Detection and Alert System**

Sachin Chaudhary, Dhruvin Raval, Parth Brahmbhatt, Ritik Patel, Ruchitkumar Patel

**Problem Statement**

Drowsy driving is feeling sleepy while operating a vehicle. This usually happens when a driver has not slept enough, had alcohol, is under certain medications or has sleep disorders. Research has found that meals that are high in fat, carbohydrates, or calories may increase sleepiness.  
No one knows the exact moment when sleep will come over their body. Falling asleep at the wheel is clearly dangerous, but being sleepy also affects your ability to drive safely, even if you do not fall asleep.

1. Drowsiness may make you less attentive to surroundings.   
2. Slows your reaction time if need to steer suddenly or stop and Worsen decision making.   
3. Along with slower reaction times, your vision can get blurry and sensitive to light when you are tired. It may be harder to tell how far away other cars are from you.

**Problem Importance**

Drowsy driving is a major contributor to motor vehicle collisions. In a Centers for Disease Control and Prevention (CDC) survey, an estimated 1 in 25 adult drivers reported having fallen asleep while driving in the previous 30 days.   
According to the National Highway Traffic Safety Administration (NHTSA), in 2017 drowsy driving led to at least 91,000 crashes, resulting in roughly 50,000 injuries and 800 deaths. In 2020, there were 633 deaths based on police reports.  
However, these numbers are underestimated, and over 6,000 fatal crashes each year may involve a drowsy driver.

**Problem Solution**

As part of this project, we developed a web-based safety product to identify the driver, capture car details, location and detect microsleep. The driver details were stored in the RDBMS database which were captured during the registration process and were matched with the real time image of the driver to identify the driver for authentication to start the trip. The vehicle number is extracted from the vehicle image and stored in the database for tracking. Once the authentication is done, we can start the monitoring system to track the drowsiness. The details will be stored in RDBMS database and once the microsleep is detected the system will generate notification and send the alter to driver.

Analytics is performed on the data to extract insights on microsleep patterns with respect to factors like time, location, and driver.

This product will help rideshare or vehicle rental companies to identify sleep deprived drivers and can prevent fatal accidents before it happens. Alert and notifications will allow them to track the driver details and location and time where the incident happened.

**Related Work**

The implementation of this project is based on python, SQL server, dlib, OpenCV.  
We have used important python libraries like Cv2, face\_recognition, easy\_ocr, scipy spatial, pyttsx3 in our project implementation.

Below is the detailed list of python libraries used:

|  |  |  |  |
| --- | --- | --- | --- |
| Library Name | Installation Command | Documentation Link | Description |
| numpy | pip install numpy | https://numpy.org/ | library for the Python programming language, adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. |
| pandas | pip install pandas | <https://pandas.pydata.org/> | pandas are a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series. |
| pyodbc | pip install pyodbc | <https://pypi.org/project/pyodbc/> | pyodbc is an open-source Python module that makes accessing ODBC databases simple. |
| face\_recognition | pip install face-recognition | <https://pypi.org/project/face-recognition/> | Recognize and manipulate faces from Python or from the command line with the world’s simplest face recognition library. Built using dlib’s state-of-the-art face recognition |
| io | Module is a part of the standard library | <https://docs.python.org/3/library/io.html> | The io module provides Python’s main facilities for dealing with various types of I/O |
| PIL | pip install pillow | <https://pillow.readthedocs.io/en/stable/> | This library provides extensive file format support, an efficient internal representation, and powerful image processing capabilities. |
| easyocr | pip install easyocr | <https://pypi.org/project/easyocr/> | EasyOCR is for Optical Character Recognition (OCR) that allows you to easily extract text from images and scanned documents. |
| cv2 | pip install opencv-python | <https://pypi.org/project/opencv-python/> | OpenCV-Python is a library of Python bindings designed to solve computer vision problems. |
| os | standard library | <https://docs.python.org/3/library/os.html> | This module provides a portable way of using operating system dependent functionality. |
| imutils | pip install imutils | <https://pypi.org/project/imutils/> | A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much easier with OpenCV. |
| re | standard library | <https://docs.python.org/3/library/re.html> | This module provides regular expression matching operations |
| pyicloud | pip install pyicloud | <https://pypi.org/project/pyicloud/> | PyiCloud is a module which allows to interact with iCloud webservice. |
| dlib | pip install dlib | <https://pypi.org/project/dlib/> | A toolkit for making real world machine learning and data analysis applications. It is a general-purpose cross-platform software library written in the programming language C++. |
| pyttsx3 | pip install pyttsx3 | <https://pypi.org/project/pyttsx3/> | It is a text-to-speech conversion library in Python. |
| [scipy](https://scipy.org/) | python -m pip install scipy | <https://scipy.org/> | SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. |
| datetime | standard library | <https://docs.python.org/3/library/datetime.html> | The datetime module supplies classes for manipulating dates and times |
| Flask | pip install Flask | <https://flask.palletsprojects.com/en/2.2.x/> | Flask is a micro web framework written in Python |
| flask\_wtf | pip install Flask-WTF | <https://pypi.org/project/Flask-WTF/> | Form rendering, validation, and CSRF protection for Flask with WTForms. |
| flask\_login | pip install flask-login | <https://flask-login.readthedocs.io/en/latest/> | Flask-Login provides user session management for Flask. |
| urllib.request | pip install urllib3 | <https://docs.python.org/3/library/urllib.request.html> | Python's urllib.request for HTTP Requests |
| Secrets | pip install Flask-Secrets | <https://pypi.org/project/Flask-Secrets/> | A Flask extension to generate random secret tokens. |
| wtforms | pip install WTForms | <https://pypi.org/project/WTForms/> | Form validation and rendering for Python web development. |

Apart from the above list we have used *shape\_predictor\_68\_face\_landmarks.dat* which is a .dat file which can be read using the dlib (deep learning library) of python to recognize the facial points of any person's face. The content of this file is in public domain. More information of this file can be found [here](http://dlib.net/face_landmark_detection.py.html).

**Methods**

**Database**: We have used SQL server Management Studio V18.10 to create a database named **facerecognition** and store user data.   
Data definition language used to create tables can be found here:



**Database architecture**

Diagram

Description automatically generated with medium confidence

**Product Architecture**

Diagram

Description automatically generated

**Webapp**:

To implement the above product, we have created a webapp in flask. Below are the steps taken to create the above shown product.

1. Create HTML web structure for webapp to create Home, Register, Login, Account and Logout page.
2. Home page is the name of the main page of a website where visitors can find hyperlinks to other pages on the site.
3. The Register Page is to register the users, User details such as user credentials, driver detail, vehicle details are captured on this page through the flask form and gets stored in RDBMS database through the help of python and SQL. Once the user is created on Register page, It flashes the user created message and points to the Login page.
4. Login Page is to login the users. User uploads the current driver image and vehicle image for authentication.
   1. The python code for face recognition checks the driver image in database and authenticates if the driver is registered.
   2. The python code for vehicle authentication processes the uploaded vehicle image and extracts the vehicle number from the number plate.
   3. Once the above two steps are done the start trip button appears.
   4. Users can start the drowsiness monitoring process. The process will execute till the user kills the process. Monitoring process continuously keeps tracking the user’s eye and check for drowsiness. Drowsiness whenever detected alert sound is generated.
5. Logout Page is to logout from the profile if user is not ready for trip.
6. Once the data is generated, chart developed on tableau are refreshed with fresh data.

**Drowsiness Detection:**

1. With the help of *shape\_predictor\_68\_face\_landmarks.dat file* create landmark on users’ face.
2. Using the landmark points generated, Calculate the Euclidean distance between landmark on opposite sides of the eye.
3. Calculate the aspect ratio of the eyes and set the ratio. During implementation of this project aspect ratio was taken as 0.15. (This can be changed as per the requirement)
4. Once the ratio is reached below the threshold decided, generate alarm.

**Result**

**Webapp Flow:**

**Registration page**:

Login Credentials, driver and vehicle details are entered.

Graphical user interface, text

Description automatically generated

**Data Recorded for newly registered user:**

Data is inserted in the database.

Graphical user interface, text

Description automatically generated

**Login Page:**

Once the account is created, the website routes to login page**.** Enter the credentials to login.

Graphical user interface, text, application, website

Description automatically generated

**User authentication:**

Front end: Once Login upload driver’s and vehicle’s image.

Graphical user interface, text, application

Description automatically generated

Back End:

Python code running in background checks for driver’s image recognition and vehicle number extraction.

Text

Description automatically generated

Start trip: Once the user is authenticated, start trip button shows up. Start the trip to monitor the driver.

Graphical user interface, text, application

Description automatically generated

**Monitoring:**

The drowsy monitoring process starts, Once the driver feels drowsy, an alarm is generated. Process continues till user shuts the program.

**Graphical user interface, application

Description automatically generated**

**Tableau Analytics:**

Tableau analytics are created on ***facerecognition*** database on SQL Server. Tableau charts are automatically refreshed with every fresh data inserted in database by the webapp product.

Below is the snapshot of dashboard created on tableau:

Graphical user interface

Description automatically generated

**Discussion and Conclusion**

While implementing the project we faced a challenge in extracting the vehicle number from the vehicle’s image. The process of extracting the alphanumeric from the vehicle’s image through EasyOCR is slow. EasyOCR works faster if CUDA is installed and is faster with a GPU. It is also noted that the alphanumeric extracted are not always accurate. We recommend reducing the noise in the image and apply easyocr to check for accuracy. An alternative approach can be taken to test the accuracy, for example tesseract engine.

**Contribution**

|  |  |
| --- | --- |
| Webapp Development | Parth Bhavinkumar Brahmbhatt, Sachin Chaudhary |
| Face Recognition | Sachin Chaudhary |
| Number extraction | Sachin Chaudhary |
| Location capture | Ruchit Alpeshbhai Patel |
| Analytics on the database tables | Ritik Rajeshbhai Patel |
| Drowsiness Detection | Dhruvin KetanKumar Raval |
| Database and code Integration | Sachin Chaudhary, Ruchit Alpeshbhai Patel, Dhruvin KetanKumar Raval, Parth Bhavinkumar Brahmbhatt, Ritik Rajeshbhai Patel |

**References**

https://www.sleepfoundation.org/drowsy-driving

https://docs.opencv.org/3.4/d2/d42/tutorial\_face\_landmark\_detection\_in\_an\_image.html

<https://www.cdc.gov/sleep/features/drowsy-driving.html>

<http://dlib.net/face_landmark_detection.py.html>

<https://docs.python.org/3/>

https://flask.palletsprojects.com/en/2.2.x/tutorial/

**Appendices**

Link to Git hub: <https://github.com/DAB103-2021/proj101>

Project Presentation: 

Tableau Dashboard fie: 