Securo Drive
Cloud Computing Project

Ankush Jain (aj2885), Darshan Solanki (das968), Smit Sheth (shs257) & Vishnu Thakral (vvt223)
New York University
Brooklyn, NY

1. Introduction & Motivation

According to a study conducted by Virginia Tech Transportation Institute in 2013, 20 percent of car crashes are caused by fatigue, with young drivers practically vulnerable to driving while fatigued. This was a significant jump from collected statistics in the preceding years as the attribute to only 2 or 3 percent crashes. According to the National Sleep foundation's survey in 2019 about half of U.S. adult drivers admit to consistently getting behind the wheel while feeling drowsy. About 20% admit to falling asleep behind the wheel at some point in the past year – with more than 40% admitting this has happened at least once in their driving careers. The National Highway Traffic Safety Administration estimates that drowsy driving was responsible for about 100,000 crashes, 71,000 injuries, and 6000+ fatal crashes in 2019.

With numerous other resources to support this claim, it can be stated that drowsiness/tired driving is a prominent contributor towards the number of fatal car crashes annually (AAA Foundation for Traffic Safety). At the same time, the development of drowsiness detection technologies is both an industrial and academic challenge. Some modern examples are Volvos Driver Alert Control. Which warns drivers suspected of drowsy driving by using a vehicle-mounted camera connected to its lane departure warning system (LDWS). Mercedes-Benz collects data drawn from drivers driving patterns through its Attention Assist System. While these solutions are effective they have two major drawbacks, they are costly and are limited to very few cars at this point.

Our project is focused towards reducing the number of crashes by continuously monitoring the driver's facial behavior. The methodology introduces a concept for preventing accidents from drowsiness in a reliable and scalable manner due to complete AWS end-to-end management. It also leverages google cloud platform which is further discussed in Section 3.

2. Architecture

The Amazon Web Services cloud platform has been used to develop this web application. In order to implement a microservices based application architecture, we utilized various AWS offerings for the different functionalities of the application. We have distributed the various processes to get a serverless application development environment.

The Amazon Web Services offerings that have been perused in the development process are as below.

- **Amazon Cognito:** Utilized in implementing the user authentication, session management and service usage authorization.
- **Amazon Rekognition**: Utilized in recognizing the faces of the users in order to add another layer of security.
- **Amazon Kinesis Video Stream**: Used in live streaming the video from the camera inside user's vehicle for drowsiness detection
- **Amazon EC2**: Hosts the artificial intelligence model for performing the drowsiness detection on the live stream video from the user's vehicle
- **Elasticsearch**: Stores the data logs generated from the AI analysis of the video stream in real-time
- **Kibana**: Visualized interactive and realtime graph from the user data logs in Elasticsearch
- **Amazon Polly**: Alerts the user with voice recommendations to ensure that the user is attentive
- **API Gateway**: Two offering of the API Gateway have been utilized in the project
  - **REST API**: Enables creation of all the essential APIs to enable data transfer between frontend and the backend of the web application
  - **Web Socket**: Establishes bidirectional data flow for sending and receiving drowse alerts between EC2 server and the frontend.
- **SageMaker**: Responsible for training the model on user video frames to supplant current models in existence in future.
- **AWS Lambda**: Used to provide the serverless backend logic computation along with API Gateway
- **DynamoDB**: Provides no-SQL data storage for information like users and trips

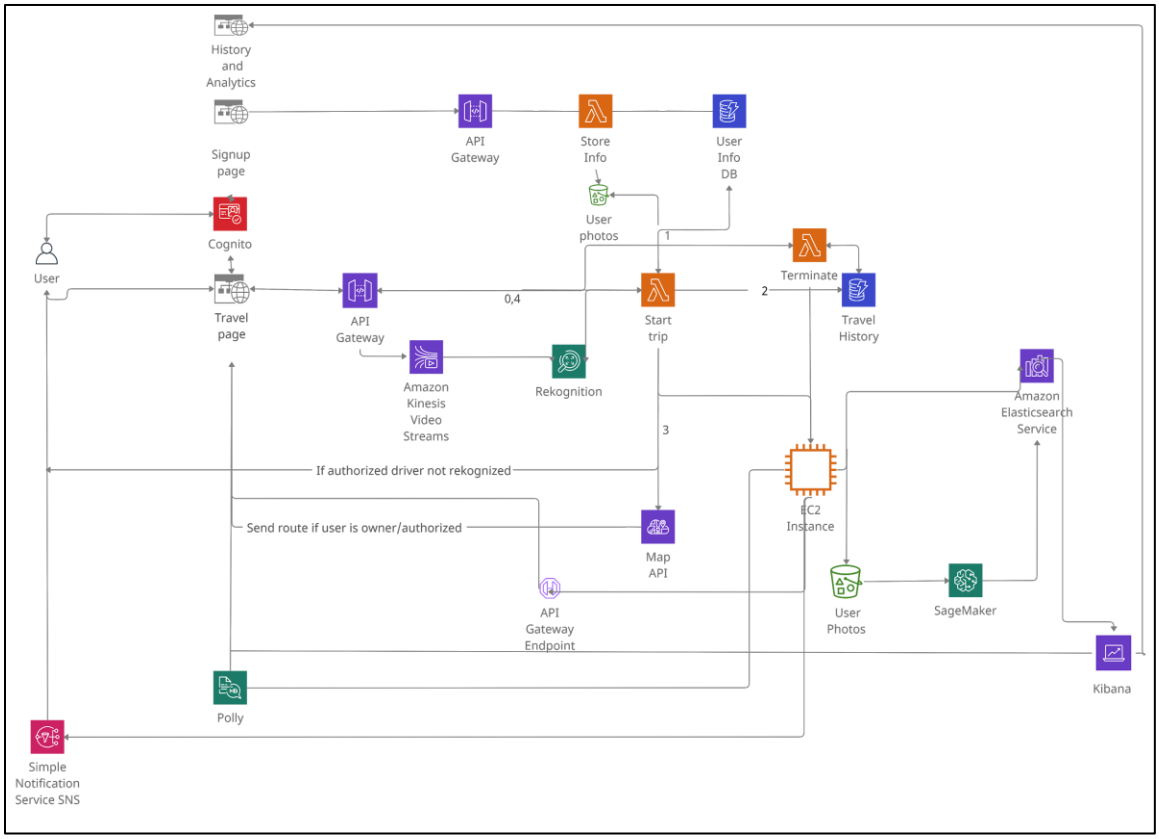

Figure 1: Architecture diagram

3. API Endpoints

The following API endpoints were created in order to allow the data and functionality exchange between the backend, database, and the front end of the web application.

1. /auth/register - POST - project_auth_register : Indexes profile photo to rekognition and stores other info to DynamoDB
2. /auth/update - POST - project_auth_register : Updates the profile photo to rekognition and stores other info to DynamoDB
3. /history/item - POST - project_history_item : Fetches the list of all the historic trips for the user
4. /history/list - POST - project_history_list : Fetches the information about a particular trip
5. /trip/recommendations - POST - project_get_recommendations : Fetches the rest stop recommendations on the basis of the user's current GPS location
6. /trip/route - POST - project_get_route : Fetches the GPS route between the start and destination locations
7. /trip/start - POST - project_start_trip : Used to start the EC2 AI model job instance
8. /trip/end - POST - project_terminate : Used to terminate the EC2 AI model job instance
9. /trip/video - GET - hls_url : Generates a HLS URL for Live Kinesis Video Stream.

4. Work Flow

4.1. Trip Creation

Users first signup with details like name, phone number, email address and upload a photo. All this information is stored in DynamoDB to persist the profile data of the user. Login is enabled with AWS Cognito which allows entry to the homepage after required verification of correct entered details.

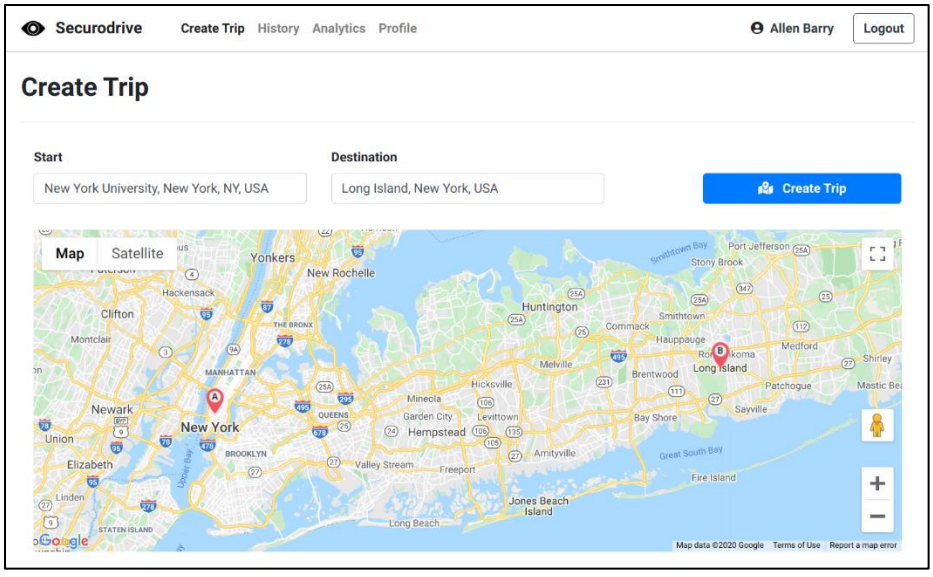

Figure 2: Screenshot of the create trip page

The user can create a new trip with the start and destination location. The maps and directions are obtained with Google Maps. However, we implemented additional security features of face-detection who is taking a drive. When a user creates a trip, AWS Rekognition is triggered implicitly.

If the creator isn't the owner, then their picture is sent to the owner via AWS SNS text. This can prevent any potential car theft attempts.

4.2. Current Trip

If the trip creator was owner then the navigation begins. The AI enabled - live video analysis also begins on an AWS EC2 server. It tracks the yawns and drowsy eyes of the driver with cv2 model dlib discussed in section model prediction.

On UI we also show basic details about the trip like start and end location, estimated time to arrive, trip_id created with a mixture of user_id and random generator to maintain proper partitioning key on DynamoDB, when things scale up.

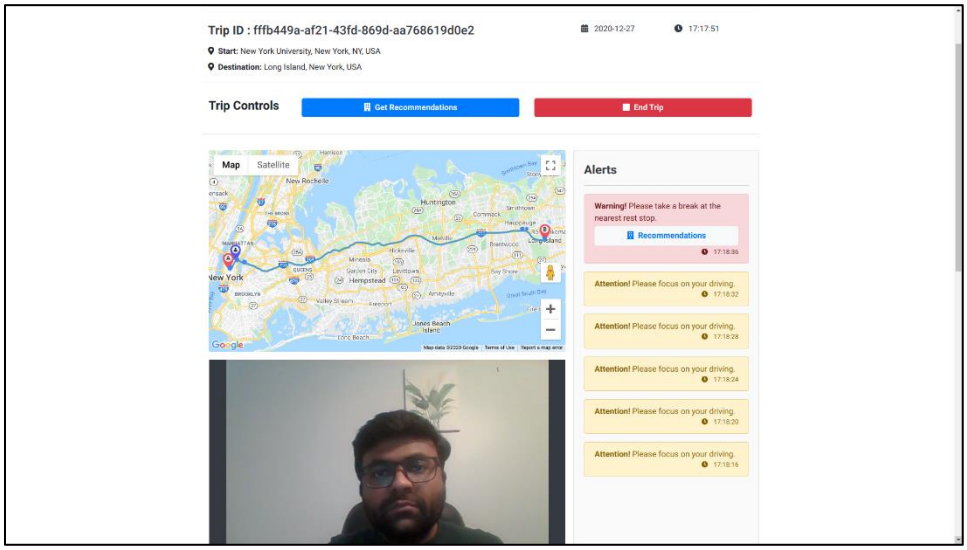

Figure 3: Current trip page

4.2.1. Simulate Navigation

From google maps, we have the user journey coordinates, we simulate the user route with Blue Marker indicating driver's current location along the route. The route given by Google Maps is the best route is what we have assumed for better user experience.

4.2.2. Model Prediction:

Parallelly, when a trip has been created, A live Amazon Kinesis Video stream of the user is also displayed on the UI. From the kinesis video stream deployed on ec2, we process the frames in successive fashion. The pseudo code for this process is as follows.

1) Preprocess the frame and resize to desired size with gray scale image.

2) Applied the dlib's face detection model on individual frames, this detects the bounding box of the face and gives us 68 points called landmark points of the face region as shown in the figure 2.
3) We extract the mouth and eye region from those coordinate calculations.

Once the eye and mouth regions are extracted, the model predicts a score for each of the feature. For eyes, if the score falls under a certain threshold, then drowse is prediicted. It usually means that the eyes are closed. Similarly, for ywans, if the mouth of the person is opened beyond a certain threshold, then it counts as a yawn. Both of these actions should occur for a duration of two seconds or more in order to be classified as a yawn or a drowse.

The snapshots below display the bounding box predictions of the yawns and the drowses. The end user can look at the snapshots at the end of the ride in order to verify the alerts.

Figure 4: Model predicted bounding boxes for Yawn (mouth) and drowse (eyes)

4.2.3. Eye Aspect Ratio (EAR)

Given the frame, we compute the ratio between eye's height to eye's width, Assumption: when we feel drowsy, our eyes tend to get smaller and the ratio of height to width decreases. Formula for Computation: (leftEyeEAR + rightEyeEAR) / 2.For instance leftEAR cal: (|p38 - p42|| + |p39 - p41|) / |p37 - p40| (ref fig 3.1). When this threshold goes below a certain defined threshold we measure the severity of this occurrence over a period of frames. As driver's drowsiness increases our system will react to it by sending alerts to the frontend and also over the websocket API.

4.2.4. Mouth Aspect Ratio (MAR)

Similarly, we go with the assumption when a person yawns, the ratio of height to weight increases, as it surpasses beyond a certain threshold, we monitor this at our end before sending notification to the driver.

This ratio depends from person-to-person and with dlib, we can set only a generic value, say eye_ratio: 0.22 and mouth_ratio: 0.60. When we see this ratio threshold limit has been surpassed, we monitor the occurrence of such events and then give out warning alerts over the websocket API.

For the very first instance of a user taking its trip with a securo drive, we generate the data from dlib where X features would be images coming from kinesis and y label is (ear,mar) ratio given out by dlib model. At

the end of the trip, we trained a deep neural network on this data for user specific eye mouth feature capturing y labels into binary variables “alert” and “no alert” and saved the model in s3 bucket.
When the user travels again, the saved model will be triggered than dlib model. This will overcome the barrier of giving false alerts to the user and retaining customer’s interest to get more Deep Learning based prediction.

4.2.5. Audio Alert:

A notifications box is visible to show any new alerts when the driver appears to be yawning for too long or is drowsing off. When such a warning appears voice alerts are provided within the app by using Amazon Polly. This also makes more flexible notifications and for recommendation users would not be looking at text flashes over the UI.

Then we wait for users to respond to our suggestions, upon affirmative confirmation we suggest users with nearby rest-areas by hitting google place API from current location as simulate above. The user can then select the nearest rest stop and get a new route via the rest stop.

4.2.6. Kibana Live Logs Analytics

At the end of the trip, the user can see the visual summary of their trip as show below. The logs generated by the AI model are sent to Elasticsearch and this live visualization is created with the logs in Kibana. The following visualization shows the amount of eyes opened and closed. If the value crosses a certain threshold then it appears as a yawn or a drowse.
If the mouth opens more than the threshold, then it counts as a yawn. Similarly, if the eyes are shut below a certain threshold, it counts as a drowse. All these actions must occur over a time period of two seconds or more.

Figure 5: Kibana visualization of the live logs generated by the AI algorithm from the live video stream

4.3. Additional Analytics

An analytics module is also present. The user can see a visual summary of their drowsing habits during different times of the day. We plotted the alert counts across all the trips into a single graph for the user and determined as to which hour of the day, the user has drowsed the most. With such visual analytics, a person can determine the suitable hours of the day for driving so that they can drive safely and plan their trips accordingly.

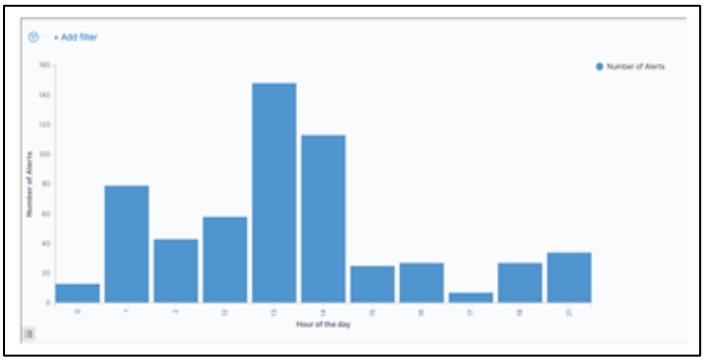

Figure 6: Kibana visualization for hourly distribution of drowse alerts for the user

4.4. History Module

We have also provided a history module so that the user can check out their past trips. The user can access two views. In the first view they get to see the brief summarized information about all the trips that they have made in the past. In the second view, they can expand the trip history in order to obtain the specifics about the trip. The users can also view the drowse and yawn graph for the particular trip. And, they can also access the snapshots of the AI predicted bounding boxes for the instances of yawn or drowse.

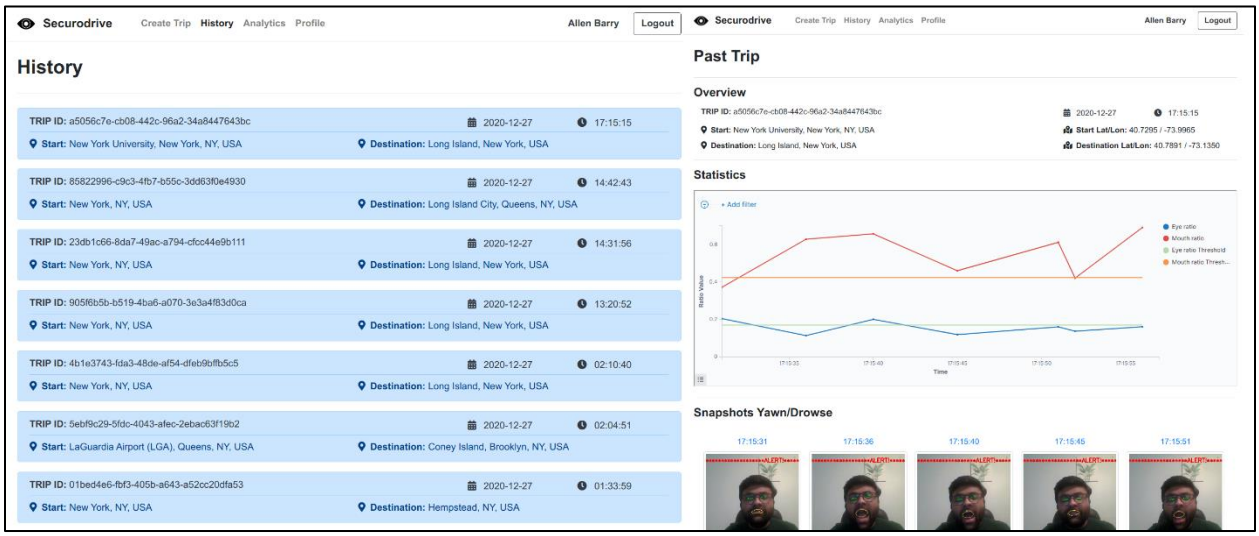

Figure 7: History module snapshot

4.5. Profile Update Module

Since this application utilizes services like Rekognition, Simple Notification Service (SNS), and Cognito for authentication and security, it made sense to give the user more control over authentication related aspects of the application. The users get the capability to update their user profile picture for Rekognition. The visual user authentication is a vital part of this application. Moreover, the user can update their phone number in order to receive the texts about any unauthorized vehicle users.

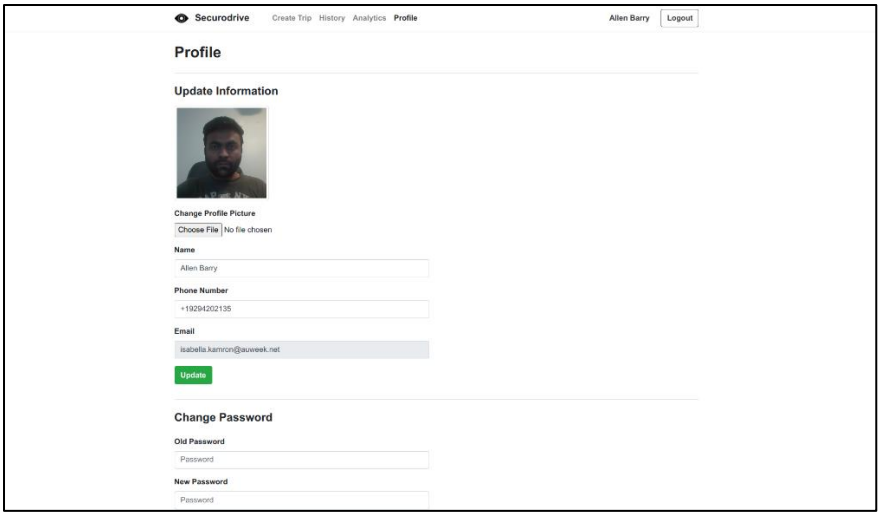

Figure 8: Profile update page

5. Future Work

There are a lot of possibilities in improving this particular project. Firstly, we would like to generate more logs of different varieties from the AI model analysis and ush them to elastic search. We would like to generate more real-time or interactive visualizations with Kibana.

Next, with the Deep Learning model, we can make customized drowsiness detection according to the user's facial features and behavior. With more logs getting pushed from user's trip, we can showcase performance analysis of the user which will be beneficial to decide whether they are fit to take the ride or not.

Lastly, we would also like to study the effects of different lighting conditions and the environment of the driver's video camera setup. As the professor rightly suggested during our demonstration that the effectiveness of this system can change with variation in lightning conditions. We would like to train more robust models with sage make that can handle different light conditions.

6. Conclusion

In conclusion, we successfully managed to deploy the solution proposed in this project by utilizing the scalable, serverless, and microservices enabled Amazon Web Services platform. We were able to come up with a suitable architecture for the web application and utilize the different AWS offering to solve this use-case.

With the Deep Learning model, we can make customized drowsiness detection according to the user's facial behavior. With more logs getting pushed from user's trip, we can showcase performance analysis of the user which will be beneficial to decide whether to take on the ride or not

The overall impact of such a system in a real-world setting would allow the prevention of accidents that occur due to the people drowsing off while driving vehicles. We also try to mitigate the possible accidents

by providing dynamic recommendation and re-routing the drowsy user to the nearby rest-areas. We also made sure to utilize voice alerts and stringent alerting policies in order to gather the driver's attention in case they are sleepy.

References

[1] Walter, Laura. "Wake Up and Drive: Fatigue Causes 20 Percent of Crashes." EHS Today, 5 June 2013,www.ehstoday.com/safety/article/21917988/wake-up-and-drive-fatigue-causes-20-percent-of-crashes.
[2] "Drivers Are Falling Asleep Behind the Wheel." National Safety Council, www.nsc.org/road-safety/safety-topics/fatigued-driving.
[3] Drowsy Driving NHTSA reports. (2018, January 08). Retrieved from https://www.nhtsa.gov/risky-driving/drowsy-driving
[4] Jabbar, Rateb, et al. "Real-Time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques." Procedia Computer Science, Elsevier, 24 Apr. 2018, www.sciencedirect.com/science/article/pii/S1877050918304137.

Appendix

YouTube URL: https://youtu.be/ulE0xlEMt88
GitHub URL: https://github.com/vvthakral/securodrive
S3 Portal URL: http://project-frontend-web.s3-website-us-east-1.amazonaws.com/