Parent Drive Connect: Empowering Parents to Monitor Student Drivers with Ease.

Manasa Durga Sangana

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management

University of North Carolina Wilmington

2024

Approved by

Advisory Committee

_____ _____

Hamed Saeidi Minoo Modarreszadeh

_____

Karl Ricanek Jr, Chair

Accepted By

_____

Dean, Graduate School

TABLE OF CONTENTS

ABSTRACT

Parent Drive Connect: Empowering Parents to Monitor Student Drivers with Ease. Sangana, Manasa Durga, 2024. Capstone Paper, University of North Carolina Wilmington.

Aggressive and speeding drivers may impact not just the speeder but pedestrians, bicycles, and other drivers. Due to inexperience, new drivers are more prone to encounter speeding problems due to poor decision making. Parents have several concerns when their child begins the journey of automobile driving: safety, cost, insurance, etc. New drivers are charged significantly higher rates for insurance; male new drivers are charged higher rates than female drivers. Young new drivers tend to drive above posted speed limits.

Parent Drive Connect, a website that attempts to address the problem of new teenage drivers regarding speeding. It provides a dashboard that allows parents to track driving data, such as location, speed, past route information, and adherence to safety guidelines. Advanced GPS technology, accelerometer data, and car diagnostics can all be used as data for this platform. This telematics system provides parents complete visibility into their inexperienced student drivers' driving habits and development.

Parent Drive Connect uses Google's Geocoding API to improve location intelligence by converting geographic coordinates into addresses humans can understand. This improves user experience and makes it possible to comprehend driving patterns easily.

The web-based platform offers an intuitive user experience that makes it easy for parents to participate when they're on the road. Parent Drive Connect is a state-of-the-art driver monitoring technology that combines intelligent analytics and precision telematics to improve the safety and learning process of new and inexperienced drivers.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Should parents use the proposed Parent Drive Connect and other technology tools to encourage their teenage children to drive responsibly? Studies and surveys have revealed that parents frequently worry about their teenagers' reckless driving habits. In a poll by Students Against Destructive Decisions (SADD) and Liberty Mutual, 76% of teenagers said their parents worry more about them speeding than about them drinking and driving [1]. Teenagers are more inclined than older drivers to speed and to let other cars pass in front of them too close together. These unsafe driving practices seem to worsen in the presence of a teenage male passenger. The National Highway Traffic Safety Administration (NHTSA) reports that in 2020, drivers who were involved in fatal crashes had 35% of male drivers and 18% of female drivers (ages 1 5–20) speeding at the time of the crash [2]. Parental supervision of teenage drivers, including rule-setting and the use of monitoring devices, was linked to decreased rates of dangerous driving behaviors, such as speeding, according to research from the Children's Hospital of Philadelphia [3].

Parent Drive Connect and comparable monitoring programs provide a comprehensive strategy for tackling the problems related to teen driving. Parents should embrace these technologies since they allow them to keep an eye on their child's driving conduct. Through a website linked to the Parent Drive Connect system, parents may obtain comprehensive reports on their adolescent driver's activities, encompassing details of speed and location, which are in compliance with traffic regulations. Additionally, this website can also work as a cooperative platform where parents and teenagers can create objectives and monitor advancement toward safer driving habits. Teens are empowered to take

responsibility for their driving behavior while getting parental support and supervision when clear expectations and goals are set.

This starts with Controller Area Network (CAN bus), a standardized communication protocol used in vehicles. Data interchange between various Electronic Control Units (ECUs) inside a car is made possible via the CAN bus, opening different features like vehicle performance and driving.

This project uses CAN data captured from hardware devices (e.g. OBD devices, custom CAN hardware) that can aggregate the CAN data. The data is then put into an SQLite database and then transformed into a format that may be used, like CSV or a Pandas Data Frame. Processed data is used as inputs to application programming interfaces (APIs) for monitoring speed violations by the driver.

Working with road-related data is made easier with the help of the Google Roads API, a component of the Google Maps Platform [18]. It offers snap-to-road technology, which aligns GPS locations gathered from several sources with the most likely roadways on Google Maps. Applications that use GPS monitoring, such as tracker apps and vehicle movement patterns analysis, can benefit from this.

Additionally, it is possible to investigate how web technologies might be used to display and visualize the outcomes of the APIs on interactive web platforms. Websites can show how the processed data and API results may be dynamically displayed on a website, giving users actionable insights and visualizations by utilizing web frameworks like Django and frontend technologies like HTML and CSS. In addition to the backend and frontend discussed above, parents will expect alert messages for speeding or emergency, which is also integrated into this solution.

This capstone project exemplifies the training received in this graduate program and the application to real-world problems.

Strong business use case with a targeted demographic: detection of speeding to alert parents of teen drivers. This tool may save lives as teen drivers tend to speed and behave recklessly [3].

Novel datasets of vehicle data: used two datasets, one publicly available and the second is a private dataset.

Web application: developed the frontend and backend application that houses the data and access control, examines the CAN data, and compares to posted speeds via Google API, determines speeding events, logs them, displays route and speeding events to maps, and sends email alerts to parents when speeding occurs.

CHAPTER 2: LITERATURE REVIEW AND ANALYSIS

2.1 Controller Area Network

Controller Area Network (CAN) is the most widely adopted in-vehicle network technology by automakers to transmit in-vehicle communications due to its low cost and large requirement within a single unit (up to 500 million chips) [4]. CAN's resiliency is considered acceptable; its noise resistance, fault tolerance, and resistance to external common-mode interference capabilities are provided by its use of unshielded twisted pair lines. CAN transceivers nowadays can detect and report errors in the physical layer.

The operation of motor vehicles is subject to constant monitoring. Electronic sensors obtain these measures and transmit them via the internal vehicle communications protocol (CAN). Dimitrios [5], briefly described the sensors employed for the retrieval of these parameter values. The values obtained from the OBD-II scanner are discussed and shown. Kushiro [6], used a mobile network to log OBD-monitored data and send it to a telematics center. To prevent any component breakdown, this data was used for a prognostics model that is based on correlations among fault codes (cautions/warnings from vehicle sensors).

According to PES Modern College of Engineering [7], their research involves alerting the driver when they become sleepy and sending their current position to their emergency contacts if the alert crosses a threshold. Convolutional Neural Networks are used by the system to evaluate the driver's degree of exhaustion and drowsiness in an efficient manner. A driver sleepiness monitoring system, as shown by Texas State University study [8], could warn drivers and avert catastrophic accidents caused by inattentive driving. In this study, the prospect of utilizing wireless wearables to create an accurate and user-friendly driver sleepiness detection system was investigated.

No cost of entry Web-based mapping was widely used for various purposes, including location-based services, navigation, and situations where instant access to geographic data was required. While some, like Google Maps (GM), were created for profit, others, like OpenStreetMap (OSM), were inspired by volunteer labor. According to research published in June 2020 [9], One of the critical aspects of OSM is the homogeneity and quality level of its information, GM is also heavily consulted, however there is inhomogeneity between densely inhabited and rural areas. For the purpose of automatically assessing internet mapping services, Boottho [10] performed research that makes use of APIs from online map providers. Google Maps is an online mapping tool that uses GPS crowdsourcing to obtain precise traffic data approved by industry and research groups. Premium membership allows access to this data through APIs made possible by Google Maps. This traffic data is also publicly accessible through Google Maps' online interface, albeit with less features and a need for further pre-processing [11]. The current technologies available to enable the use of publicly available traffic data through the Google Maps web interface are either proprietary or lack necessary functionalities.

In June 2023, Jiri Vary [12], along with Michal and Matej, explained generating synthetic vehicle speed along a given route and evaluating the fidelity of the generated output using objective and subjective methods. In their research, they were using OSM as their primary map source for Generating Synthetic Vehicle Speed Records Using LSTM.

2.2 Similar Systems Available in the market

Fleet management products play a crucial role in optimizing vehicle operations, enhancing safety, and improving efficiency for businesses and organizations. ParentDrive, a specialized platform for parental oversight of driving behaviors, offers unique features tailored to the needs of parents monitoring their children's driving activities. This section

of literature review aims to explore existing fleet management products with similarities to ParentDrive, focusing on their functionalities, features, and effectiveness in addressing similar objectives.

A Dynamic Dashboarding Application [13] where the authors introduce a dynamic dashboarding application designed to mitigate the challenges inherent in fleet monitoring scenarios. Central to their solution is the utilization of RESTful Web Things, which are made available through a Web Thing Model compliant gateway.

Fleet Complete [14] is a comprehensive fleet management solution offering real-time GPS tracking, driver behavior monitoring, and vehicle diagnostics. It provides customizable dashboards, geofencing capabilities, and route optimization features to enhance fleet efficiency and safety.

Verizon Connect [15] offers a suite of fleet management tools, including GPS tracking, driver performance monitoring, and fuel management. It features intuitive reporting and analytics tools, customizable alerts, and integration with third-party applications for seamless fleet management. Also, AutoPi [16] is a versatile IoT platform designed for various automotive applications, including fleet management. With AutoPi, fleet managers can monitor and optimize the performance of their vehicles, enhance driver safety, and streamline fleet operations. AutoPi collects and analyzes telematics data from fleet vehicles, including engine diagnostics, fuel consumption, and driver behavior. By tracking metrics such as acceleration, braking, and idling, fleet managers can identify opportunities for fuel savings, maintenance optimization, and driver coaching to improve safety and efficiency.

Parent Drive Connect serves as an extension of conventional fleet management platforms, specifically tailored to address the concerns surrounding teenage drivers. With a primary

focus on over speeding, a critical component of reckless driving behavior, this tool is designed to cater to the needs of parents seeking to monitor their teenage drivers effectively. Through timely alerts and comprehensive monitoring features, Parent Drive Connect empowers parents with the necessary tools to ensure the safety and responsible behavior of their teenage drivers on the road.

CHAPTER 3:DATA

In vehicular research and safety monitoring, Controller Area Network (CAN) data serves as a cornerstone resource, offering insights into vehicle diagnostics, performance, and driving behavior. These datasets are broadly categorized into open-source and commercial options, each presenting unique advantages and considerations. Open-source CAN data, sourced from publicly available repositories or research projects, provides accessibility and flexibility but may lack comprehensive coverage and realism. Conversely, commercial CAN data, obtained from vendors or platforms, offers curated and reliable datasets but may be subject to restrictions and costs. In our project, we encountered challenges in accessing both open-source and commercial CAN data sources, highlighting the complexities of data acquisition in this domain. Despite these challenges, we have leveraged datasets from Vrany and Colorado State University (CSU) in collaboration with Lapetus Solutions, Inc (a global company that is a spinout from University of North Carolina Wilmington) to advance our research objectives in vehicular safety and parental oversight.

3.1 Overview of CAN Data Sources

CAN data serves as a valuable resource for a wide range of applications, including vehicle diagnostics, monitoring, and analysis.

Broadly categorized, CAN data sources encompass both open-source and commercial options, each offering distinct advantages and considerations. Open-source CAN data refers to datasets that are freely accessible and redistributable, often sourced from publicly available repositories and research projects.

However, open-source CAN data may also present limitations, such as variability in data quality, coverage, and representativeness, depending on the source and collection methodology.

Commercial CAN data encompasses datasets provided by commercial vendors or platforms, typically available for purchase or subscription. These datasets are often sourced from proprietary sources, including vehicle manufacturers, telematics providers, or data brokers.

However, access to commercial CAN data may be restricted by licensing agreements, subscription fees, and usage restrictions, presenting challenges for researchers and developers with limited resources or budget constraints.

3.1.1 Open-Source CAN Data Sources

Despite extensive efforts to locate open-source Controller Area Network (CAN) data containing speeding information, no publicly available datasets were found during the research process. It is worth noting that access to such datasets, particularly those containing negative test scenarios where over speeding incidents occur, remains limited in publicly accessible repositories. The absence of publicly available CAN data with speeding information poses a challenge for research projects aiming to incorporate diverse and realistic test scenarios.

As a result, for the purpose of our project, we have opted to utilize the datasets obtained from Vrany [12] as positive test cases. While these datasets lack negative test scenarios, they provide valuable insights into driving behavior and serve as suitable test cases for our experimentation.

Moving forward, future efforts to collect and curate open-source CAN data with speeding information may enhance the availability of diverse test scenarios for research and development endeavors in the field of vehicular safety and monitoring. Collaboration among researchers, institutions, and industry stakeholders could facilitate the creation and sharing of such datasets, thereby fostering advancements in the domain of driving behavior analysis and safety enhancement.

3.1.2 Commercial CAN Data Sources

In attempting to incorporate commercial Controller Area Network (CAN) data into our project, AutoPi emerged as a potential solution. However, our efforts encountered significant obstacles, impeding further progress. AutoPi's platform did not offer provisions for downloading or personal use of CAN data, thereby restricting our ability to access the necessary datasets for our research project. Moreover, concerns surrounding safety and pricing emerged as prominent issues. Notably, the requirement to pay for every logging of alerts or attempt to retrieve CAN data presented a substantial financial barrier and operational constraint.

Consequently, the utilization of AutoPi for our project proved unfeasible, highlighting the complexities and challenges associated with accessing commercial CAN data sources. Moving forward, alternative strategies for acquiring commercially sourced CAN data while mitigating safety concerns and managing costs will necessitate careful consideration.

Exploring partnerships with data providers or exploring other commercial platforms may present viable avenues for overcoming these challenges and facilitating the integration of commercial CAN data into our research endeavors.

3.2 Data Sources Used

3.2.1 Vehicle Speed Dataset by Jiri Vrany

Vrany, J., Krepelka, M., and Chumle, M. [12] conducted a study titled "Generating Synthetic Vehicle Speed Records Using LSTM," which was published in the proceedings of AIAI 2023. The dataset derived from their research is publicly available through IEEE. This dataset comprises a total of 2815 drives, covering 22179.5 kilometers over a duration of 455 hours. The primary focus of their data collection endeavor was to gather route-related information. As a result, the dataset consists of three continuous and three categorical attributes, providing a comprehensive overview of driving behavior across various routes.

In their methodology, the authors addressed missing values in the dataset during preprocessing. They employed interpolation techniques to infer missing values and analyzed whether the route traversed through settlements or remained outside. This preprocessing step ensured the integrity and completeness of the dataset for subsequent analysis and experimentation.

It is noteworthy that the datasets obtained from Vrany lack negative test scenarios where over speeding incidents occur. Consequently, for the purpose of our project, we are utilizing these datasets as positive test cases. Furthermore, as indicated in the dataset description provided by IEEE, the data instances are ensuring a consistent and evenly sampled representation of driving behavior across the dataset.

3.2.2 CAN Dump by CSU in collaboration with Lapetus

For my research project, I have integrated a dataset sourced from Colorado State University, which was collected in partnership with Lapetus Solutions, Inc [17]. This dataset, available in SQLite format, encompasses vehicle-related information such as speed and corresponding GPS coordinates. Additionally, Lapetus Solutions has provided a supplementary test dataset specifically tailored for experimentation purposes, ensuring a comprehensive range of data for analysis.

It is important to note that for this project, we are assuming the test dataset to be devoid of noise, as our primary objective is to explore methodologies for retrieving maximum speed limits based on vehicle location and subsequently generating alerts for parents. This integration of datasets underscores our commitment to utilizing diverse data sources for robust experimentation, thereby advancing research efforts in the domain of vehicular safety and parental oversight.

3.3 Dataset Filtering (CAN Dump by CSU)

In this project, filtering the CAN data is essential to improve the accuracy of the average speed calculation. The raw data, recorded at a high frequency of every 0.1 second, contains a surplus of information that may not be necessary for the specific analysis or application at hand. The CAN data contains noise, which has been attributed to the hardware used to collect and collate the data. To reduce the impact of the noise which can result in false over speeding, a boxcar filter is used. boxcar filtering is a standard technique for minimizing the impacts of spurious noise in a data stream. The ideal boxcar (moving average) filter in real-time is a rolling average of a real-valued input signal over a definite time which extends from a sample at time “now” (t=0) back into the past to a time t = -T, where T is

the length of the boxcar window. Further, the boxcar is used to reduce the number of data points in the data stream. By averaging the speed over a window of consecutive data points, we ensure a more accurate representation of the actual speed.

The graphs comparing the raw data with the filtered data visually illustrate the effectiveness of this approach in obtaining more accurate average speeds. These graphs serve as compelling evidence of the benefits of filtering and accurate averaging and patterns.

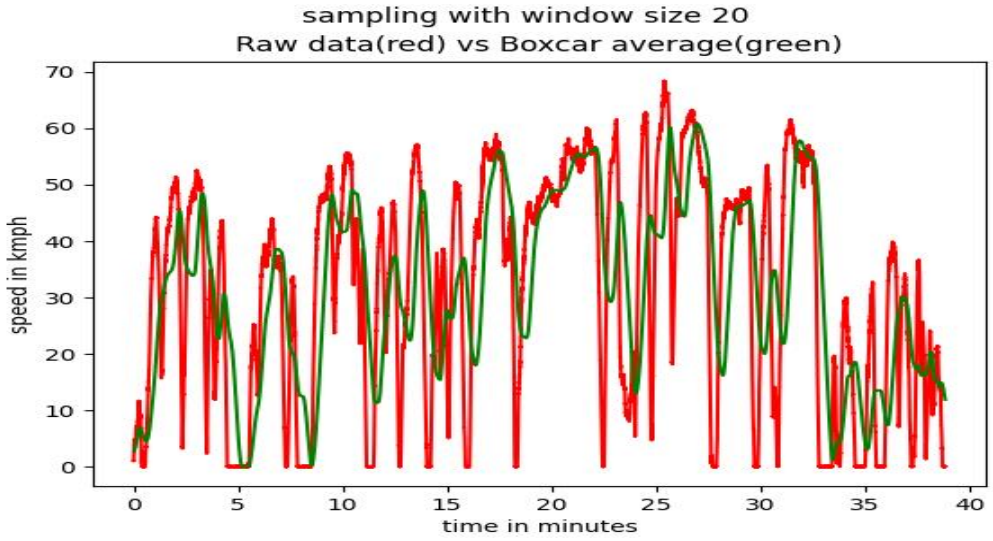

FIGURE 1:FILTERED DATA WITH TIME INTERVAL OF 2 SEC

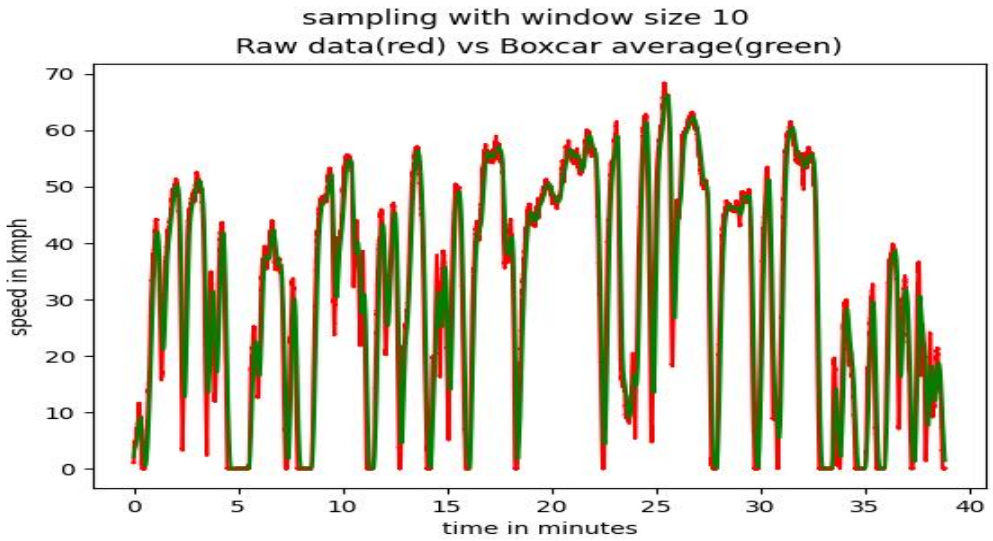

FIGURE 2: FILTERED DATA WITH TIME INTERVAL OF 1 SEC

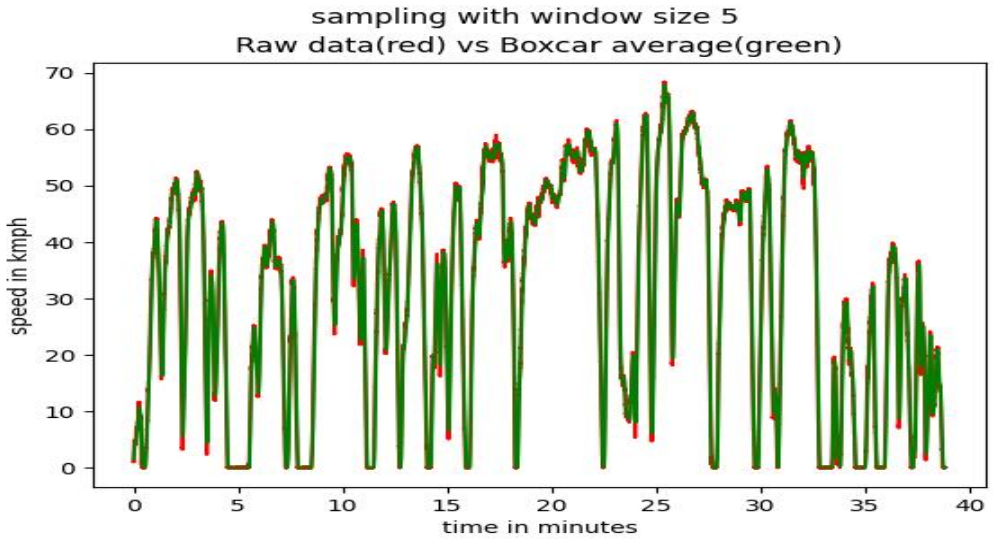

FIGURE 3: FILTERED DATA WITH TIME INTERVAL OF 0.5 SEC

While this filtering technique effectively reduces the noise it also allows to reduce the overall samples, while maintaining essential information about location and speed. Note: caution must be exercised regarding the window size used for the moving average filter because a large filter window will distort the data. For example, a large window sizes, such as 20, may lead to data loss or inconsistency in location information, particularly in scenarios where changes in location are dynamic and influenced by speed as illustrated in figure 15. Therefore, selecting an appropriate window size is crucial to balance noise

mitigation with data accuracy. ensuring the validity of insights drawn from the processed dataset.

3.4 Comparison of Posted Speed Limits with Filtered Data

Overpass API is used to gather posted speed limits for comparison with filtered data, aiming to establish a correlation and identify instances of over speeding through a specific logic. Below are the graphs that focuses on the data from the CSU dataset of two different trips assigned for two users and analyzing the relationship between posted speed limits retrieved from the Overpass API with corresponding filtered data sampled at the same rate.

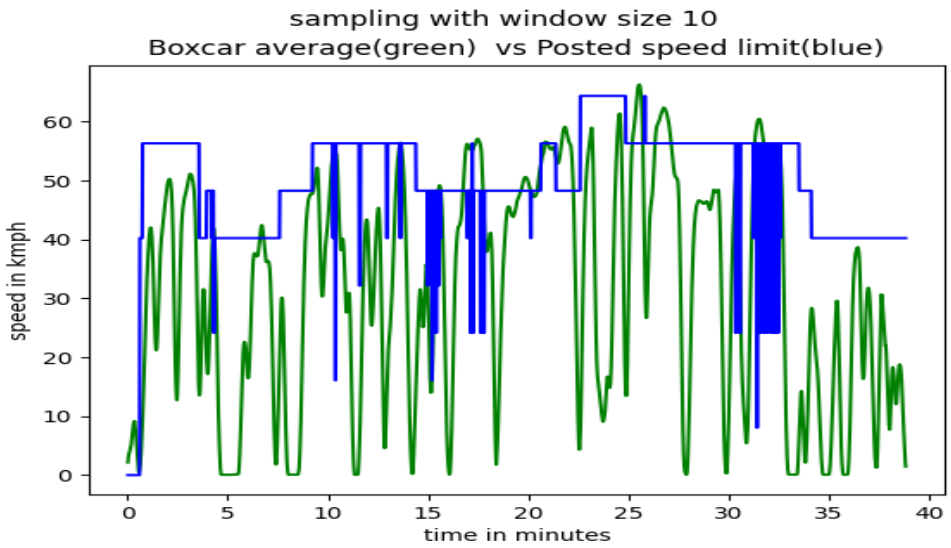

FIGURE 4: GRAPH SHOWING FILTERED DATA VS POSTED SPEED LIMIT FROM OPVERPASS API FOR CSU DATSET 1

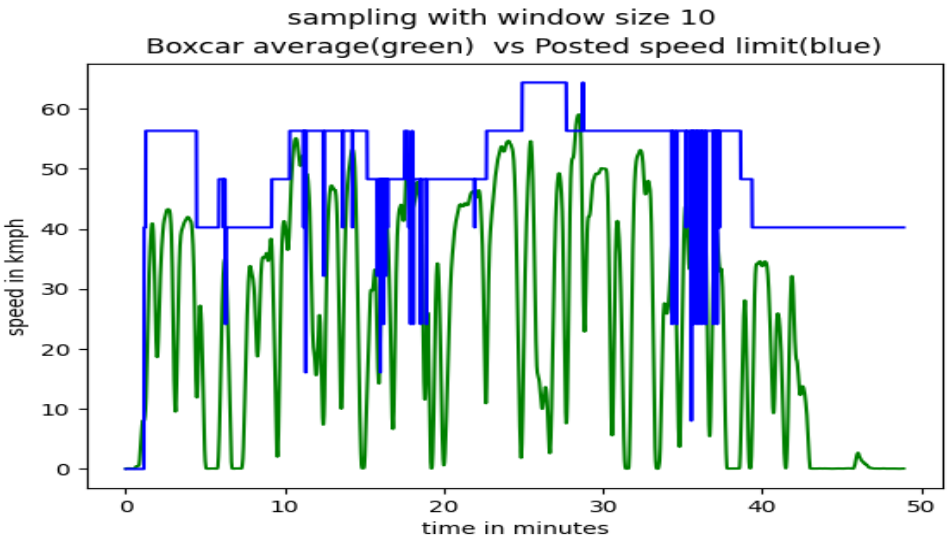

FIGURE 5: GRAPH SHOWING FILTERED DATA VS POSTED SPEED LIMIT FROM OPVERPASS API FOR CSU DATASET2

Figures 4 and 5 illustrate this comparison between the posted speed limits and the filtered data for each user. These figures serve to illustrate the noise present in the posted speed limits obtained from the Overpass API.

3.5 Noise in Posted speed limit from Overpass API

In our analysis of driving data sourced from an API, an intriguing pattern emerges concerning abrupt variations in posted speed limits. Notably, these fluctuatonare often entail sudden reductions from higher speed thresholds, such as 30, 10, or 5 mph, followed by rapid restorations within condensed timeframes. Upon rigorous cross-verification of expansive result datasets, it becomes apparent that these fluctuations predominantly coincide with specific road segments marked by intersections, exits, or the initiation of new routes, commonly identified as turns.

While definitive causation cannot be conclusively established, it is reasonable to hypothesize that the observed decreases in posted speed, especially at intersections or turns, are likely influenced by contextual factors inherent to such spatial configurations. This assumption is derived from the logical inference that regulatory adjustments in speed limits

are often implemented in response to changes in road conditions, traffic patterns, or safety considerations pertinent to these areas. Moreover, factors such as the presence of school zones may also contribute to fluctuations in posted speed limits, given the priority placed on ensuring the safety of pedestrians and children in these designated areas.

latitude	longitude	vehicle speed	address	posted speed
40.5817	-105.0943218	27.49697163	1009-1001 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0941396	28.92916965	1009-1001 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0939549	30.14624902	1009-1001 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0937697	31.2494311	1009-1001 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0935834	32.22864966	1000-1010 W Mulberry St, Fort Collins, CO 80521, USA	25
40.5817	-105.0933953	33.09602144	1000-1010 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0932043	33.85602031	1000-1010 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0930112	34.51554349	1000-1010 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0928167	35.10087497	900-998 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0926214	35.5732412	900-998 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.092426	35.90020662	900-998 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0922316	36.18094204	900-998 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0920362	36.41153282	900-998 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0918387	36.63373509	900-998 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0916396	36.86637639	900-998 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0914401	37.06919188	899-817 W Mulberry St, Fort Collins, CO 80521, USA	5
40.5817	-105.0912402	37.25672165	899-817 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0910391	37.38981932	899-817 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0908375	37.46251973	899-817 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0906365	37.51695183	899-817 W Mulberry St, Fort Collins, CO 80521, USA	15
40.5817	-105.0904372	37.5495738	815-801 W Mulberry St, Fort Collins, CO 80521, USA	35
40.5817	-105.0902386	37.57250239	815-801 W Mulberry St, Fort Collins, CO 80521, USA	35

FIGURE 6: SAMPLE DATA FROM USER1 RESULTS INDICATING NOISY POSTED SPEED DATS AT INTERSECTION AND SCHOOLZONES

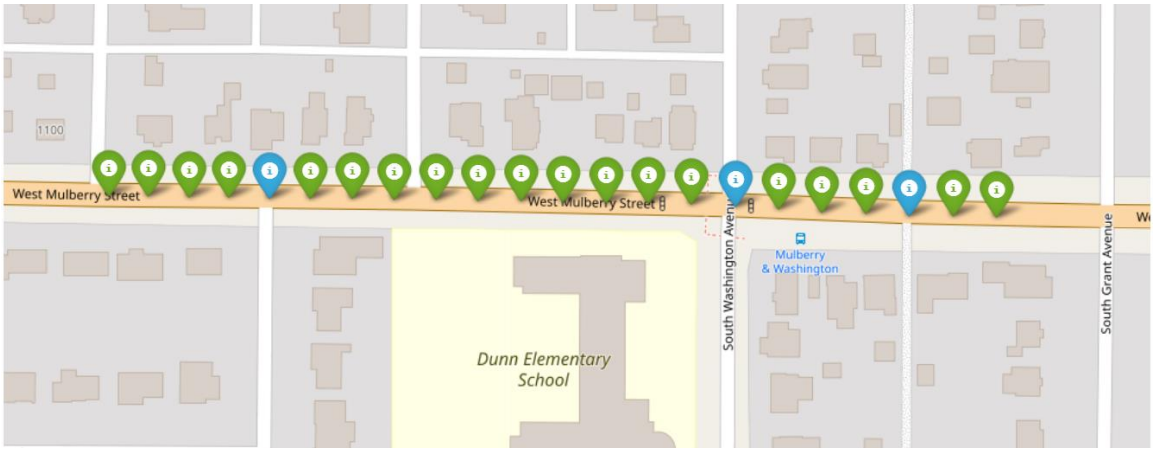

FIGURE 7:GEOMAP OF NOISY POSTED SPEED DATA AT INTERSECTION AND SCHOOL ZONE INDICATED BY BLUE MARKERS

The phenomenon of noise in maximum speed limits, characterized by sudden drops and subsequent increases, above findings suggest a plausible association between decreases in

posted speed limits and intersections, further research and empirical validation are warranted to corroborate this hypothesis conclusively.

It is essential to acknowledge that the date and time of drive entries within the dataset are simulated for testing purposes and are not reflective of real-world vehicle activities. As such, while the over speeding events and alert messages generated provide valuable insights into the system's functionality, they are based on hypothetical scenarios rather than actual driving events. This distinction underscores the experimental nature of the dataset and highlights the need for further validation and refinement once connected to actual vehicle data and undergone comprehensive data processing procedures.

3.5 Over Speeding Response (False) To Noise in Posted Speed Limits

One critical aspect scrutinized in the analysis was the system's handling of over speeding events, especially in scenarios where the Overpass API provided sudden drops in speed limits. A notable finding was the system's ability to accurately flag over speeding events even amidst Noise in speed limits, such as from 30 mph to 15 mph.

It's imperative to note that publicly available datasets may not always include over speeding details, a threshold-based approach to accurately identify instances of over speeding within drive data. By setting the threshold at 5 consequent over-speeding events or a duration of at least 5 seconds, we ensured precise detection of over speeding behavior. This approach was tailored to the sampling rate of the dataset, typically logged per second, thus enhancing the robustness and reliability of our over speeding detection mechanism. For instance, consider a scenario where the speed limit transitions from 30 mph to 15 mph and subsequently returns to 30 mph. Our system successfully discerned this abrupt change in speed limit and maintained its accuracy by not triggering over speeding alerts during such

transient alterations. This capability underscores the effectiveness of our approach in detecting and flagging over speeding events amidst dynamic driving conditions, as illustrated below.

TABLE 1: DEMONSTRATING THE SYSTEM'S OVERSPEEDING RESPONSE (FALSE) TO DYNAMIC CHANGES IN SPEED LIMITS, INCLUDING TRANSIENT DROPS AT INTERSECTIONS

Latitude	Longitude	Vehicle Speed	Route	Posted speed	Over speeding
40.5815	-105.08209	34.04280443	399-323 W Mulberry St, Fort Collins, CO 80521, USA	35	FALSE
40.5815	-105.081908	34.22269133	322-328 W Mulberry St, Fort Collins, CO 80521, USA	35	FALSE
40.5815	-105.081723	34.34609561	322-328 W Mulberry St, Fort Collins, CO 80521, USA	15	FALSE
40.5815	-105.081537	34.40351029	314-320 W Mulberry St, Fort Collins, CO 80521, USA	15	FALSE
40.5815	-105.081353	34.41301727	300-312 W Mulberry St, Fort Collins, CO 80521, USA	35	FALSE
40.5815	-105.081168	34.40742493	300-312 W Mulberry St, Fort Collins, CO 80521, USA	35	FALSE

Furthermore, it's essential to highlight that despite the drop in the maximum speed limit from 30 mph to 15 mph, as indicated in Table 2, we are not designating over speeding as true. By acknowledging and accounting for these variations in speed limits, our approach ensures greater accuracy and relevance in identifying over speeding events while navigating diverse driving environments.

As a result, we mitigate the impact of sudden drop in speed limits, such as those occurring at intersections, by requiring over speeding to persist for a duration of at least 5 consecutive seconds before triggering an alert. By imposing this threshold, we effectively negate transient fluctuations in speed limits and focus on identifying sustained instances of over speeding. This deliberate choice underscores our commitment to developing a reliable and adaptable over speeding detection mechanism.

CHAPTER 4: METHODOLOGY

This chapter describes the analysis of use cases prototyping, design process and database architecture. This chapter gives the user interaction stages and displays the features of the ParentDrive connect website based on stages of SDLC application. The primary advantage of design and diagrams is their ability to help you visualize systems. The importance of visualizing complex ideas is because our brains process images more quickly than other sorts of information.

4.1 Database Design

The seamless translation of class diagrams into object-oriented programming (OOP) languages such as Python, Java, and C++ stands as one of their most significant advantages. Prior to initiating any code development for this project, comprehensive considerations of interactions and class structures were made possible, courtesy of these diagrams. This strategic approach fosters the potential for software that not only functions with greater cleanliness but also demonstrates enhanced performance.

Before coding, it's essential to detail specifications to ensure alignment with design and operational requirements. Creating business-oriented models with class diagrams in Django projects highlights the importance of clear data models for system functionality. This process ensures the resultant code closely adheres to the envisioned design and meets operational needs efficiently.

Through this methodical approach, the ability to effortlessly exchange information and gain a comprehensive understanding of an application's overarching schematics is significantly enhanced. The relationship between Parent Dashboard and Results is modeled as a one-to-many relationship, denoted by the presence of a foreign key in the Parent

Dashboard table. Specifically, the Results foreign key in the Parent Dashboard table establishes a link between instances of Parent Dashboard and multiple instances of Results, indicating that each Parent Dashboard may be associated with multiple records in the Results table.

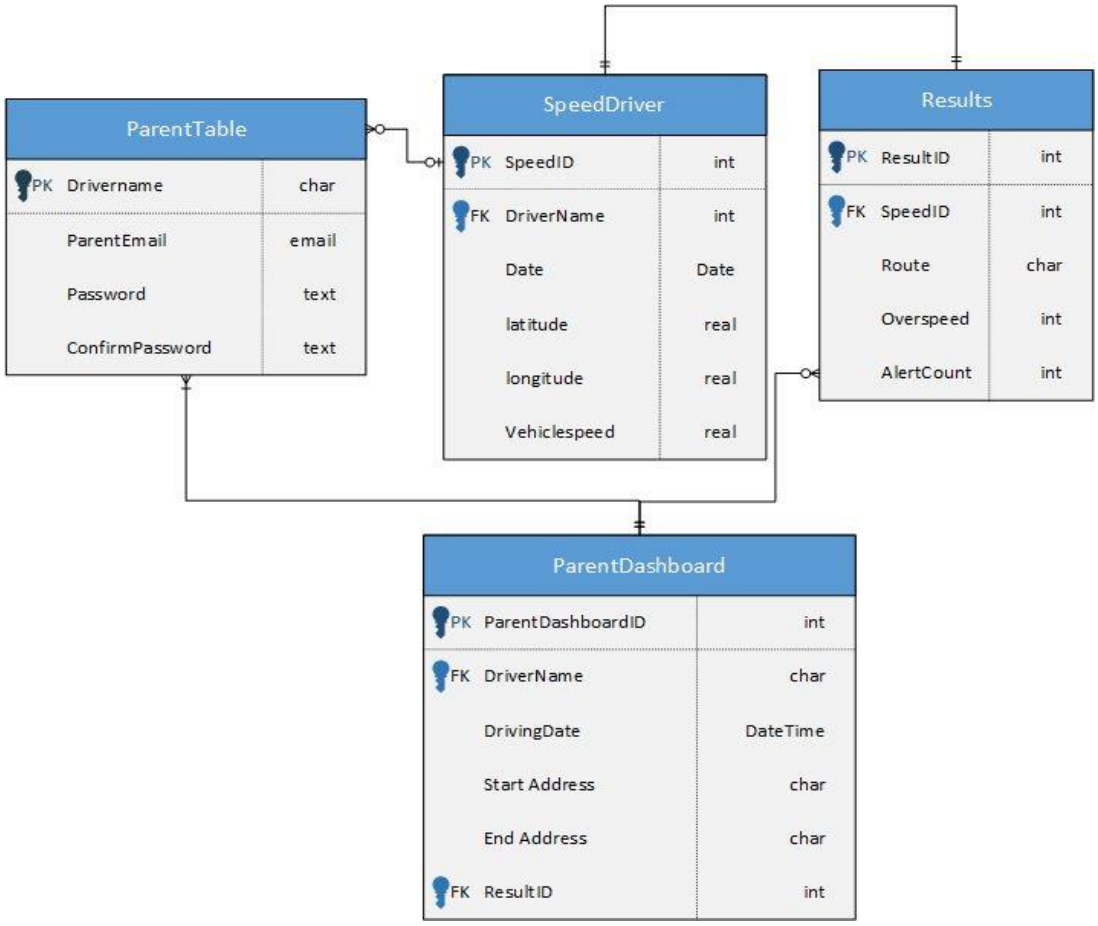

FIGURE 8: CLASS DIAGRAM FOR DATABASE DESIGN RELATED TO PARENTDRIVECONNECT

This relationship is crucial for tracking and analyzing driving-related data. By linking Parent Dashboard to Results, the system can efficiently retrieve and correlate data such as driving duration, start and end addresses, with specific driving instances captured in the Results table. Moreover, the inclusion of the Count of overspeed attribute in the Parent Dashboard table allows for aggregated analysis of overspeed occurrences across multiple driving instances, providing valuable insights for parental monitoring and driver behavior

analysis. Establishment of a one-to-many relationship between Parent Dashboard and Results by foreign key constraints enhances data organization, integrity, and analytical capabilities within the system, contributing to effective monitoring and analysis of driving-related activities.

4.2 Project Flow Diagram

A graphical tool used to illustrate the order of steps to be taken during the project management process is a flowchart of the process. I created a project flow diagram for this project, which will serve as a roadmap for all future initiatives, from inception to completion.

FIGURE 9: USER FLOW DIAGRAM FOR PARENT DRIVE CONNECT USER

The project commences with the creation of a welcome page featuring login and signup functionalities. Upon initiation, a new user is directed to the signup process, followed by login upon successful registration; failure to register prompts an error message. Upon successful login, users gain access to a summary of their trip history across various days. Subsequently, users can delve into detailed trip results via the dashboard, which facilitates navigation to two distinct pages. One page offers visualization of trip data through maps, while the other presents data in a structured format to enhance comprehension.

4.3 User interaction diagram

In this project the user interaction diagram helps in understanding the functional requirements of the system by depicting how users interact with it. This clarity ensures that the development team accurately captures user needs and expectations. The below diagram aids in validating the design of the system by visualizing the flow of interactions between users and the system. It allows stakeholders to ensure that the proposed system design aligns with user expectations and operational needs.

As delineated in the user interaction diagram, the project's interface begins with the index page, serving as the welcome portal, from which users may proceed to either the login or dashboard sections upon a successful login attempt. Alternatively, users have the option to access the signup page if they are new to the system. Specifically tailored for parents of drivers, the system provides functionalities enabling them to monitor instances of speeding through an interactive map display of relevant points. Additionally, parents possess the capability to scrutinize tabulated data for comprehensive insights. Behind the scenes, the system continuously updates its database leveraging driving dataset inputs, ensuring relevance. This systematic approach facilitates seamless user engagement and data-driven decision-making within the context of monitoring and managing driving behaviors.

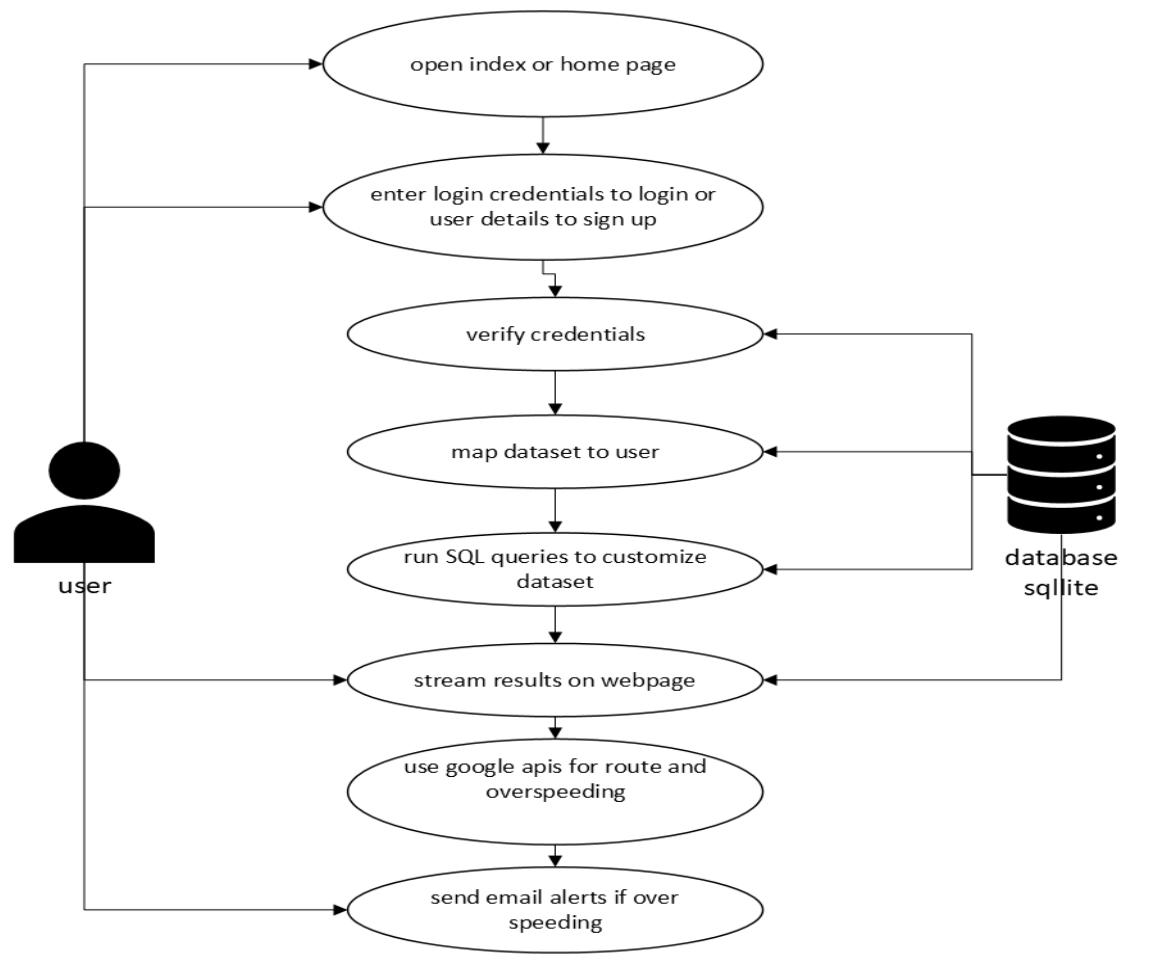

FIGURE 10: USER INTERACTION DIAGRAM

4.4 Django Framework

Django stands as one of the most popular and powerful web frameworks for building dynamic web applications. At its core, Django offers a pragmatic and efficient approach to web development, emphasizing rapid development, clean and maintainable code, and adherence to the DRY (Don't Repeat Yourself) principle. Its flexibility, scalability, and extensive built-in features make it a preferred choice for projects ranging from simple websites to complex web applications.

Key features of Django: One of the key features that sets Django apart is its adoption of the MVC (Model-View-Controller) architectural pattern, although Django itself prefers the

MTV (Model-Template-View) variant. This architecture separates the application's logic (Models), user interface (Templates), and user interaction (Views), promoting code organization and reusability. Additionally, Django boasts a built-in ORM (Object-Relational Mapping) layer that abstracts database access, allowing developers to interact with the database using Python objects rather than SQL queries directly. This simplifies database operations and enhances portability across different database backends.

Django Models and ORM: In Django, Models serve as the backbone of the application's data structure. Models are Python classes that define the structure and behavior of the application's data entities, such as users, products, or posts. Each model class typically corresponds to a database table, with class attributes representing table fields (columns) and methods defining data manipulation logic. The ORM layer translates Python code into database queries, enabling us to interact with the database using high-level abstractions. This abstraction shields from the underlying database implementation details and promotes code readability and maintainability.

Streaming HTTP Response: Instead of waiting for the complete response to be generated, the client receives content in chunks using the Django Streaming HTTP Response, a class-based response. You can send snippets of data to the client as they become available by iterating over a generator function. This is especially helpful because it saves buffering the complete response in memory when working with big datasets or lengthy processes.

Database Support: Django offers support for various relational database backends, including SQLite, PostgreSQL, MySQL, and Oracle. In this project we configure Django to use the preferred database backend based on the project's requirements and constraints.

Django's database migration system automates the process of managing database schema changes. Furthermore, Django provides a powerful admin interface that allows administrators to manage application data without writing custom administrative interfaces, saving time and effort during development.

HTML: HTML (Hypertext Markup Language) is fundamental technologies used in web development, and they play essential roles in the Django framework for building web applications. HTML provides the structure and content of a web page, while CSS defines the visual presentation, layout, and styling of that content. In the context of Django, HTML are used extensively to create dynamic and aesthetically pleasing user interfaces for web applications.

HTML serves as the backbone of web pages, organizing content into a hierarchical structure of elements. These elements range from headings, paragraphs, and lists to forms, tables, and multimedia objects. Each element is represented by tags, which enclose content and provide semantic meaning to it. For example, the <h1> tag denotes the main heading of a page, while the <p> tag signifies a paragraph of text. HTML also supports attributes, which provide additional information or functionality to elements. Attributes can define characteristics such as the source of an image, the target of a hyperlink or the type of input field in a form.

In Django, HTML templates are used to generate dynamic web pages by combining HTML markup with Python code. Django's template engine allows developers to create reusable templates that can be populated with data from the backend. This separation of concerns between presentation (HTML) and logic (Python) promotes code modularity and

maintainability. Django templates use special syntax, such as template tags ({% %}) and filters ({{}}), to embed Python logic within HTML markup. This enables dynamic content generation, conditional rendering, and looping constructs directly within the HTML templates.

The combination of HTML and JavaScript in Django enables developers to create rich, interactive user interfaces for web applications. HTML defines the structure and content of web pages. By leveraging Django's template engine and static files handling, developers can build dynamic, responsive websites that deliver a seamless user experience. Whether it's designing elegant layouts, styling elements with precision, or ensuring cross-browser compatibility, HTML and JavaScript remain indispensable tools in the Django ecosystem. Their versatility, simplicity, and widespread support make them essential skills for any web developer working with Django.

CHAPTER 5: IMPLEMENTATION

5.1 Initial Installations

In the preliminary stages of establishing the infrastructure for the Parent Drive Connect project, the foundational steps entail the installation of Visual Studio, followed by the configuration of the Django framework. Subsequently, the Django static files are integrated into the framework, ensuring smooth operation and accessibility. Additionally, SQLite is incorporated within the Django framework to facilitate database management. To streamline backend operations, SQLite Studio is downloaded and deployed to configure the database effectively. This structured approach ensures a robust foundation for the research project, fostering efficient development and management processes.

In laying the groundwork for the Parent Drive Connect project, the initial steps involve setting up Visual Studio and configuring the Django framework. This includes integrating Django static files and utilizing SQLite for database management. A structured approach is followed, ensuring efficient backend operations by deploying SQLite Studio for effective database configuration. The project's architecture utilizes Django's capabilities for web development, with modules such as user authentication and driver management. Secure access to APIs on Google Cloud Platform (GCP) is ensured through token-based authorization, implemented within the project's code. For a more detailed overview and access to project resources, please refer to the project repository on GitHub:/github.com/manasa2327/PARENTDRIVE_CONNECT.git

5.2 User Interface for Parent Drive Connect

The User Interface for Parent Drive Connect encompasses several crucial elements, each meticulously designed to enhance user experience and functionality. The Welcome Page provides a comprehensive introduction to the platform's features and objectives. Detailed User Signup and Login Pages streamline the registration and the Reset Password feature integrated into the Login Page facilitates hassle-free password recovery. Comprehensive error handling mechanisms are implemented across the webpage to ensure smooth navigation and user interaction. The Parent Dashboard offers a comprehensive overview of relevant information, empowering parents to monitor and manage their teen drivers efficiently. Email Alerts are seamlessly integrated to notify parents promptly about critical events or concerns regarding their teen drivers' activities. Finally, the architecture and functionality of the Parent Drive Connect Database are elaborated upon to highlight its role in ensuring seamless operation and effective data management within the platform.

5.2.1 Welcome Page to Parent Drive Connect

The index page of Parent Drive Connect welcomes users with a message and description, emphasizing its role in empowering parents to monitor student drivers effectively. It features prominent call-to-action buttons for login and signup, encouraging user engagement. A descriptive project overview highlights the platform's significance in addressing teenage driving challenges, particularly speeding issues. Integration of a home button in login and signup pages enhances user navigation, reflecting user-centric design principles backed by research on improving user engagement and satisfaction.

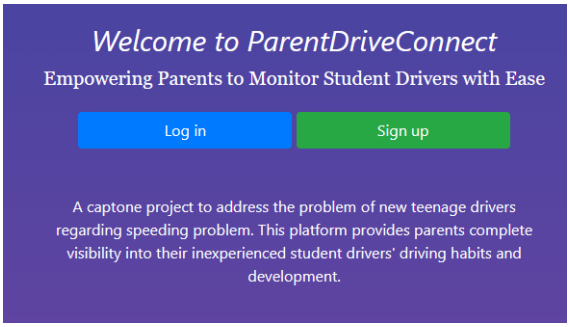

FIGURE 11: WELCOME PAGE OF PARENT DRIVE CONNECT

5.2.2 User Signup and Login Pages

The login and signup pages of Parent Drive Connect offer a seamless user experience with modern design elements and intuitive navigation. The login page features translucent forms over a background image, allowing easy input of username and password, with clear navigation links for accessibility. Upon submission, user input undergoes server-side processing for authentication. Conversely, the signup page provides a comprehensive registration form, ensuring data consistency through server-side validation and prompt error correction. Successful registration prompts a welcome email notification, enhancing user satisfaction. These pages are seamlessly integrated with Django view functions,

prioritizing user experience and streamlining the authentication and registration process effectively.

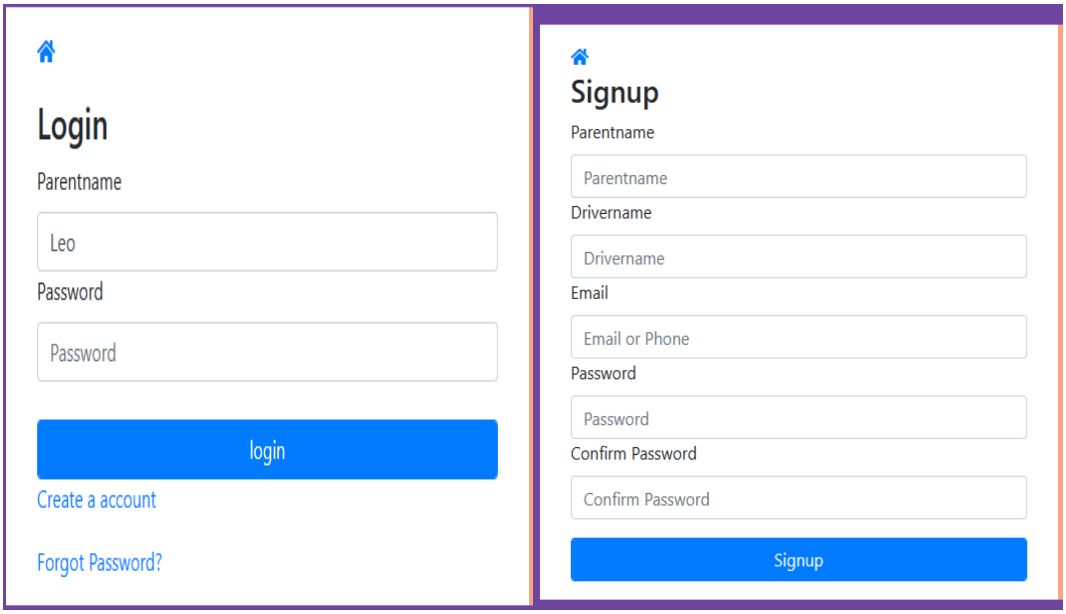

FIGURE 12: LOGIN AND SIGNUP PAGE FOR PARENT DRIVE CONNECT

5.2.3 Reset Password in Login Page:

The password reset feature enhances user experience and security by allowing users to reset their passwords if forgotten. It involves systematic design, development, and integration steps, including backend development with Django for password reset logic, frontend design for a user-friendly HTML page, and email communication for sending password reset confirmation emails. The feature ensures seamless navigation from the login page to the password reset functionality, with clear instructions and links provided in the confirmation emails for secure password resetting.

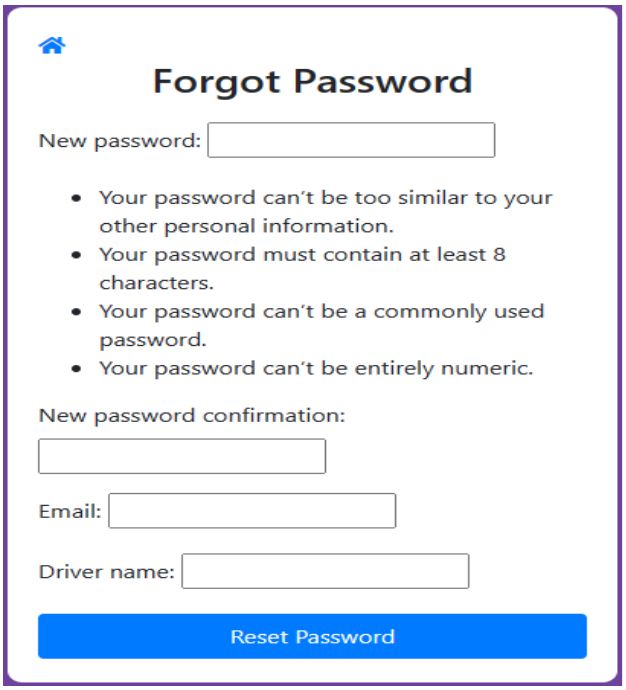

FIGURE 13: PASSWORD RESET WEB PAGE

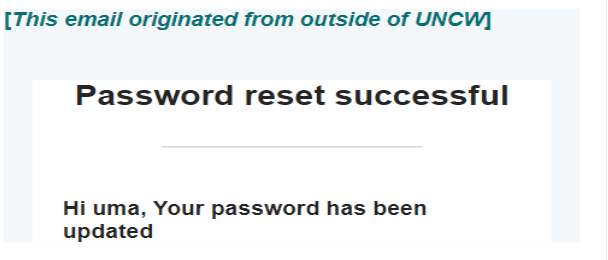

FIGURE 7: CONFIRMATION EMAIL FOR PASSWORD RESET

5.2.4 Handling Errors in Webpage:

The error-handling approach in the Parent Drive Connect web application is structured and user-centric, prioritizing informative feedback for users encountering invalid inputs or authentication failures. Leveraging Django's messaging framework, the system communicates errors effectively, guiding users to resolve issues or retry actions. It emphasizes data integrity and security by enforcing password complexity requirements and verifying user input before processing. In the Signup View, server-side logic validates data, checks for existing users, ensures password match, and enforces complexity requirements.

Successful validation results in database entry creation and a welcome email. The Login view authenticates users and prompts retry upon authentication failure. The Forgot Password View handles password reset requests, validating form data, updating passwords, and sending confirmation emails. Overall, this implementation enhances user experience and security within the authentication flow of the application.

5.2.5 Parent Dashboard

The Parent Drive Connect web application's parent dashboard page offers a user-friendly layout with personalized dynamic content. It begins with a greeting message addressing the parent by name and provides instructions for viewing driving details of a selected driver. Driving details are presented in a tabular format, including date, start and end addresses, and over speeding incident count. Interactive buttons "View Maps" and "View Data" allow access to detailed drive route maps and additional drive data. An "Update calculated results" button triggers result updates. When no data is available for the selected driver, a message indicates this, with an option to calculate results from the current day. Overall, the dashboard provides a comprehensive overview of driving activities and convenient access to relevant information, enhancing the user experience for Parent Drive Connect users.

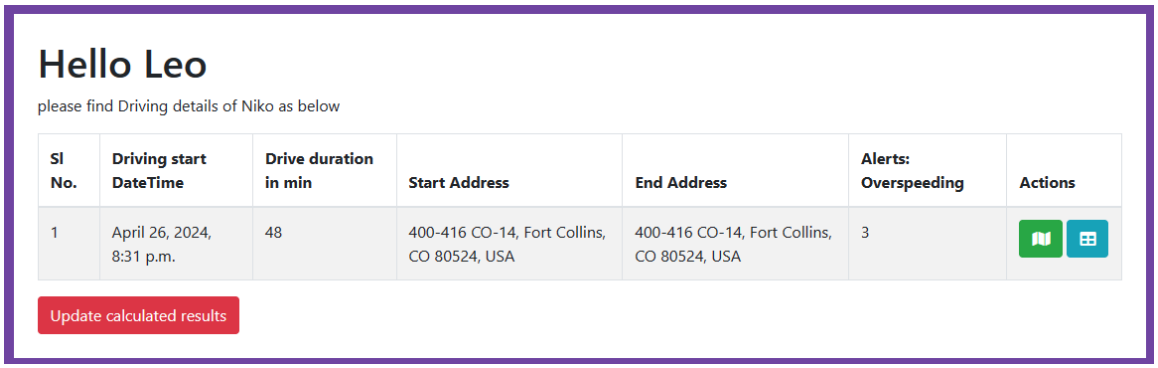

FIGURE 14: PARENT DASHBOARD EXAMPLE

5.2.6 Email Alert to Parents

ParentDriveConnect facilitates communication between parents and drivers through email notifications and monitoring. When a driver exceeds the speed limit, the system sends an email alert to the parent, ensuring their awareness of the incident and their child's safety. Integration with external APIs like Google Maps Geocoding API enhances driver tracking accuracy and route planning efficiency. Configuration adjustments in the setting.py file enable Django projects to automate email sending for various functionalities like user registration and password reset, ensuring seamless email communication within the application. Testing email functionality with Django's SEND_MAIL() function is recommended after updating settings. Parameters like 'EMAIL_HOST', 'EMAIL_PORT', and 'EMAIL_USE_TLS' define SMTP server configuration details, while 'EMAIL_HOST_USER' and 'EMAIL_HOST_PASSWORD' authenticate the application with the SMTP server. Customized email notifications are sent using Django's built-in email functionality, utilizing HTML templates for visually appealing messages. Asynchronous request handling and streaming HTTP responses enable real-time monitoring of driver activities, ensuring timely updates on behavior and safety alerts.

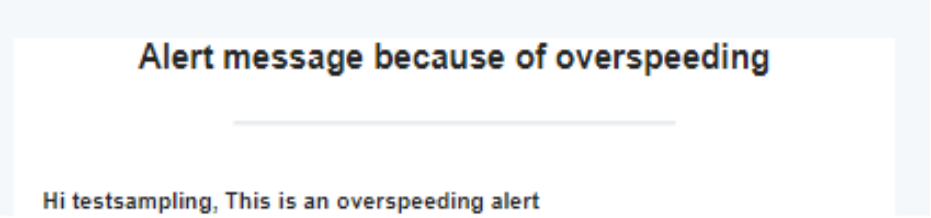

FIGURE 15: EXAMPLE OF ALERT EMAIL

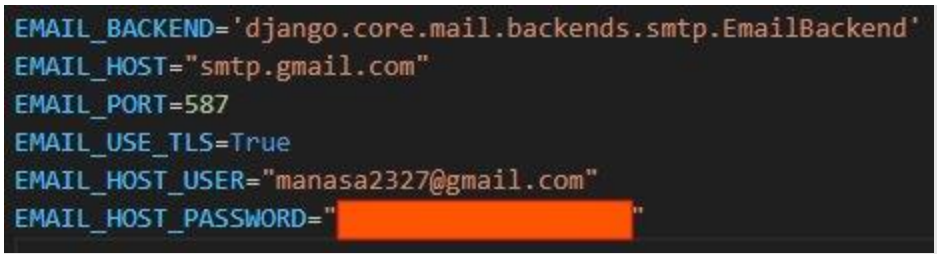

FIGURE 16: BACKEND SETTINGS.PY CONFIGUATION IN DJANGO FOR EMAIL

5.7 Parent Drive Connect Database

The database schema of Parent Drive Connect comprises four interconnected tables: ParentTable for user registration data, SpeedDriver for driving session details, Results for calculated driving statistics, and Parent Dashboard for summarized driving information. ParentTable stores parent details with email as the primary key for authentication. SpeedDriver records driving data including driver's name, location, and speed, facilitating over speeding detection. Results table computes and stores driving statistics for analysis. Parent Dashboard aggregates driving data for concise presentation on the parent interface, enhancing accessibility. Overall, the schema supports seamless user registration, data collection, analysis, and presentation, empowering parents to effectively monitor their child's driving behavior.

These table serves as the primary data source for calculating driving statistics and identifying instances of over speeding. The accompanying figure depicts the schema of a Django database as visualized within SQLite Studio. The database schema provides a structural overview of the database, outlining the composition of tables.

In summary, the database schema facilitates seamless user registration, data collection, analysis, and presentation within the ParentDriveConnect application, enabling parents to monitor and evaluate their child's driving behavior effectively.

5.8 View Over Speeding Along with Routes in Maps

"View Maps" feature of the ParentDriveConnect web application constitutes the HTML template, along with the accompanying Python function. This feature allows users, specifically parents, to visualize the driving route taken by their child, along with any instances of over speeding recorded during the trip.

The HTML template defines the structure and appearance of the route map page. It dynamically generates a map using the Google Maps JavaScript API and plots the driving route and any over speeding locations using markers and polylines. The map's center and zoom level are set to optimize the visualization of the route. Additionally, markers are placed at the start and end points of the journey, denoted by green dots for easy identification. The Python function associated with this feature, view maps, processes the POST request sent from the parent dashboard when the user selects to view the route map for a specific driving session. It retrieves the relevant data from the Results table in the database, including the latitude and longitude coordinates of each location visited during the drive. These coordinates are then used to construct the waypoints array, which forms the basis of the route polyline displayed on the map.

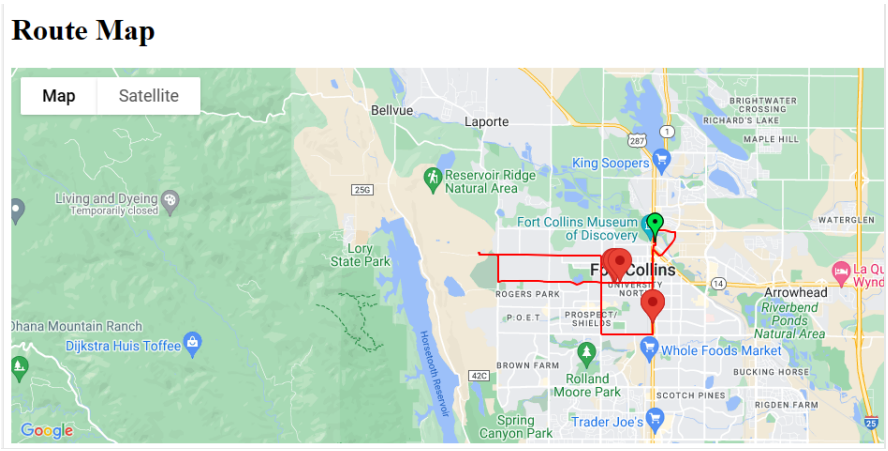

FIGURE 17: MAP VIEW USING GOOGLE API

Moreover, the function identifies any instances of over speeding recorded during the drive and adds corresponding markers to highlight these locations on the map. These markers serve as visual indicators for parents to identify areas where their child may have exceeded the speed limit, enabling them to address potential safety concerns more effectively.

5.9 Streaming of complete results of a driving instance in web page

The view data function within the ParentDriveConnect web application is designed to retrieve and display driving data for a specific driver and date/time combination on a webpage. It operates as a Django view function, handling HTTP POST requests sent from the front-end interface. Upon receiving a POST request, the function extracts the driver's name and the date/time of the driving event from the request data. It then queries the database to retrieve relevant driving data entries matching the provided driver and date/time.

Once the data is retrieved, the function dynamically generates an HTML response to present the driving results on a webpage. It begins by constructing a header section containing a greeting message and a brief description indicating the driver's name and the

date of the driving event. The function then utilizes Django's template rendering engine to populate a template (Driver.html) with the retrieved driving data. This template contains HTML markup and template variables to structure and present the driving results in a visually appealing and user-friendly manner. Below is example of how the data looks like:

Hello

please find driving results for Niko on 2024-04-26 20:31:00

latitude	longitude	vehicle speed	Route	speed limit	overspeeding	Count_overspeeding
40.5929281	-105.0761639	0.033554033999999996	400-416 CO-14, Fort Collins, CO 80524, USA	No posted speedlimit	False	0
40.5929284	-105.076164	0.0454843572	400-416 CO-14, Fort Collins, CO 80524, USA	No posted speedlimit	False	0
40.5929286	-105.0761639	0.038027905199999996	400-416 CO-14, Fort Collins, CO 80524, USA	No posted speedlimit	False	0
40.5929283	-105.0761637	0.047870421839999996	400-416 CO-14, Fort Collins, CO	No posted speedlimit	False	0

FIGURE 18: COMPLETE RESULT IN DATA FORMAT

The function employs a generator (yield statements) to stream the HTML content progressively to the client's browser, enhancing performance by avoiding the need to load the entire response content at once. This streaming approach ensures efficient handling of potentially large datasets while providing users with a responsive and interactive browsing experience.

5.10 Usage of APIs in the Project

The usage of APIs enhances the project's functionality by enabling geocoding, retrieving speed limit information, visualizing driving routes and over speeding incidents on a map interface.

APIs are utilized for several purposes within the project:

5.10.1 Google Maps Geocoding API

The Google Maps Geocoding API [17] is used to convert latitude and longitude coordinates into human-readable addresses. The Google Geocoding API is a web service that provides geocoding and reverse geocoding capabilities, converting addresses into geographic coordinates (latitude and longitude) and vice versa. One key feature of this API is its ability to return formatted addresses in a human-readable format, encompassing street names, locality, administrative areas, and postal codes. It is employed within the generate response function to fetch the formatted address for each set of coordinates retrieved from the SpeedDriver table. The addresses are then stored in the Results and Parent Dashboard tables.

```
https://maps.googleapis.com/maps/api/geocode/json?latlng=40.714224,-73.961452&key=YOUR_API_KEY
```

FIGURE 19: EXAMPLE OF GEOCODING API

The provided example is a URL endpoint [21] for the Google Geocoding API, requesting a JSON response for the geographical coordinates (latitude 40.714224, longitude -73.961452). The "key" parameter should be replaced with a valid API key obtained from the Google Cloud Platform. This request aims to reverse geocode the specified coordinates, retrieving location information such as the formatted address. This illustrates how the Google Geocoding API enables to convert geographic coordinates into human-

readable addresses programmatically, facilitating location-based services and geospatial data analysis.

5.10.2 Overpass API

To get the speed limit this project utilizes the Overpass API [19] to query OpenStreetMap data and retrieve information about the speed limit for a given Route information. This API is invoked to determine the maximum speed limit for a specific location, which is then used to compare against the vehicle speed of the driver to identify instances of over speeding.

To begin, the function initializes an instance of the Overpass () class from the overpy module, establishing a connection to the Overpass API. This connection enables the execution of queries to fetch relevant data from the OpenStreetMap database. The query executed by the function is structured to target ways (roads), The query filters these ways based on the presence of way tag, indicating that they represent roads, and retrieves the associated "maxspeed" tag, denoting the maximum speed limit for those roads.

Upon receiving the query result from the Overpass API, the function extracts the maximum speed limit information from the retrieved data. The speed limit value is then parsed from the result, handling any formatting or conversion necessary to extract the numerical value representing the speed limit. The function employs regular expressions to extract the numerical value of the maximum speed limit from the string format retrieved from the Overpass API. This ensures accurate parsing of the speed limit information, disregarding any additional characters or formatting present in the string.

5.10.3 Google Maps JavaScript API

Within the HTML script block, the Google Maps JavaScript API [20] is utilized to render an interactive map on the webpage. The Google Maps JavaScript API is a powerful tool for integrating interactive maps into web applications. Utilizing this API we can plot locations, visualize geographical data, and create custom map-based interfaces. By leveraging the Google Maps JavaScript API, researchers can effectively present spatial information, such as distribution patterns, geographic trends, or point-of-interest locations, within their research papers. This API offers extensive customization options, allowing researchers to tailor the appearance and behavior of maps to suit their specific needs. Through its intuitive interface and robust documentation, the Google Maps JavaScript API facilitates the seamless integration of dynamic mapping capabilities, enhancing the visualization and analysis of geographical data in research contexts.

```
<script defer
    src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&libraries=visualization&callback=initMap">
</script>
```

FIGURE 20: HTML SCRIPT TAG FOR LOADING THE GOOGLE MAPS JAVASCRIPT API WITH API KEY AND VISUALIZATION LIBRARY.

The provided script tag is a snippet of HTML code used to load the Google Maps JavaScript API into a web page. The initMap function initializes the map, sets the center coordinates, adds markers for start and end points, draws polylines to represent the route, and displays additional markers for over speeding points if available. The API key provided in the script URL is used to authenticate requests to Google's mapping services.

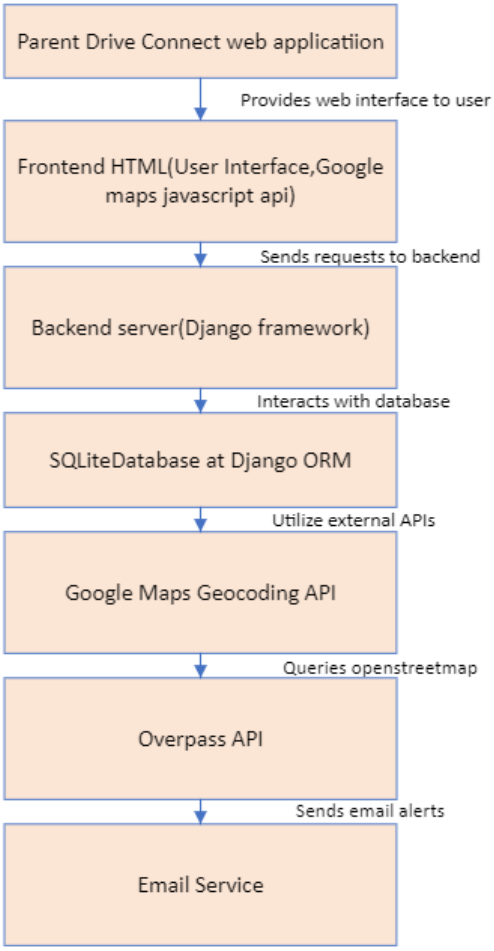

FIGURE 21: BLOCK DIAGRAM OF PARENT DRIVE CONNECT

5.11 Database Integration

In the development journey of Parent Drive Connect, the integration of backend and frontend emerged as a pivotal yet challenging aspect. Combining the functionalities of both components seamlessly required meticulous planning, coordination, and adept handling of various technical intricacies. Particularly noteworthy was the utilization of an internal database, “db.sqlite” necessitating thoughtful strategies for data management and migration within the Django framework.

The integration of backend and frontend components posed significant challenges, primarily due to the need for seamless communication and synchronization between the two layers. Ensuring that data flowed smoothly from the backend to the frontend, and vice versa, demanded careful consideration of data formats, API endpoints, and data retrieval mechanisms. Challenges were encountered in aligning backend data structures with frontend requirements and in establishing efficient data transfer protocols to facilitate real-time updates and interactions.

Parent Drive Connect relied on an internal SQLite database (db.sqlite) for data storage and retrieval, presenting unique challenges in terms of data management and accessibility. To streamline database operations and facilitate data manipulation, it became imperative to leverage external tools such as SQLite Studio.

Effective management of database migrations was critical to maintaining data consistency and integrity throughout the development lifecycle of Parent Drive Connect. Django's migration framework facilitated this process by automating the generation and execution of migration scripts based on changes in the project's data models. By utilizing Django's migrate and make migrations commands could efficiently synchronize database schema changes with the project's codebase, ensuring seamless deployment and scalability while minimizing the risk of data loss or corruption.

CHAPTER 6: TESTING

The testing techniques used to verify the stability and dependability of the Parent Driver Connect functionality in the application are presented in this chapter. A variety of user interactions are covered by the Parent Driver Connect capability, such as account creation (signup), login authentication, dashboard access, and password recovery. The thorough

testing strategy used is to verify the functionality's compliance with the specifications, its usability.

TABLE 2:TEST CASES AND RESULTS

Test Case	Scenario	Expectation	Result	Action
TC001	User attempts to sign up with existing parent name	System should display an error message indicating name duplication	Pass	
TC002	User submits mismatched passwords during signup	System should prompt user to enter matching passwords	Pass	
TC003	User leaves mandatory signup fields blank	System should prompt user to fill in all mandatory fields	Fail	Updated logic in both login and sign-up pages. provides error message while no inputs.
TC004	User enters password with insufficient length	System should request a password of minimum 8 characters	Fail	Ensure system enforces password length requirement as expected
TC005	User sets a password without	System should require at least one alphabetic	Pass	

	alphabetic characters	character in the password		
TC006	User sets a password without numeric characters	System should require at least one numeric character in the password	Pass	
TC008	User successfully logs in and accesses dashboard	System should grant access to the parent dashboard upon successful login	Pass	
TC009	User requests password reset with valid email	System should reset password and send confirmation email	Pass	
TC011	Parent logs in and accesses the dashboard	System should display the parent's personalized dashboard containing relevant information and options	Pass	
TC012	Different parents access their respective dashboards	Each parent should see only their own data and settings, ensuring data isolation and privacy	Pass	

TC013	Parent views drive schedules for different dates	The dashboard should display drive schedules for the selected dates, allowing parents to plan accordingly	Pass	
TC014	Parent views drive schedules with different timings on the same date	The dashboard should accommodate multiple drive timings for the same date, providing a comprehensive view	Fail	Updated the system models accurately to present all drive timings for the selected date,
TC015	Parent views maps displaying over speeding markers based on drive data	Maps should visually represent drive routes with markers indicating instances of over speeding	Pass	
TC016	Parent views tabular data corresponding to map data	Tabular data should mirror the information presented on the maps, providing detailed drive statistics	Pass	
TC017	Parent creates drive entries with	The system should allow parents to input drive entries with	Fail	created drive entries with flexible timings,

	identical dates but different timings	identical dates but varying timings		accommodating diverse scheduling needs
TC018	Parent creates drive entries with different sampling rates	Parents should be able to customize the sampling rate for drive data, enabling granular analysis and tracking	Pass	

CHAPTER 7: ANALYSIS OF RESULTS AND USER FEEDBACK

The primary objective of the project was to develop a website capable of serving diverse users with varying driving log data, subsequently analyzing instances of over speeding through API integration and facilitating automated message alerts. The efficacy of the website's data ingestion mechanism was examined to ensure seamless integration of driving log data from two different data sources available.

7.1 Parent Dashboard Insights

A key emphasis was placed on evaluating the customization and personalization options available within the Parent Dashboard, as well as the effectiveness of alert messages. Below are the results of different users with different over speeding alerts, duration of drive time.

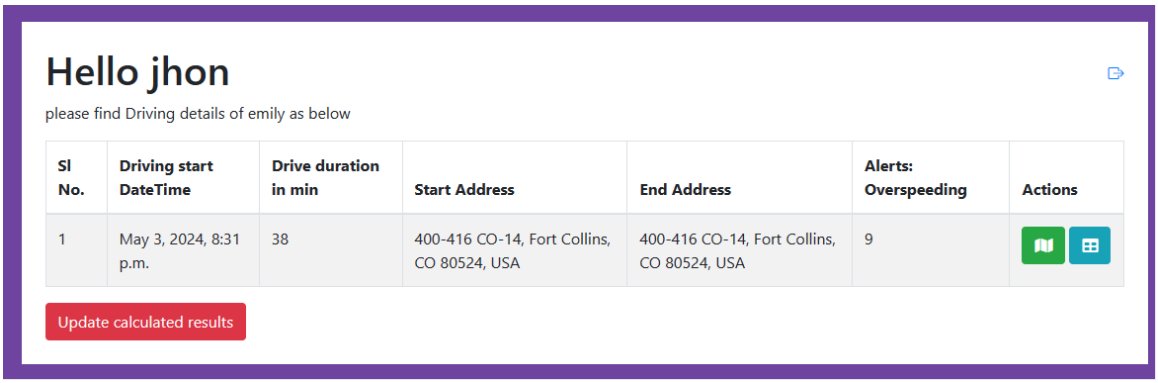

FIGURE 22:USER 1 WITH CSU DATASET1 WITH SAMPLING RATE1SEC

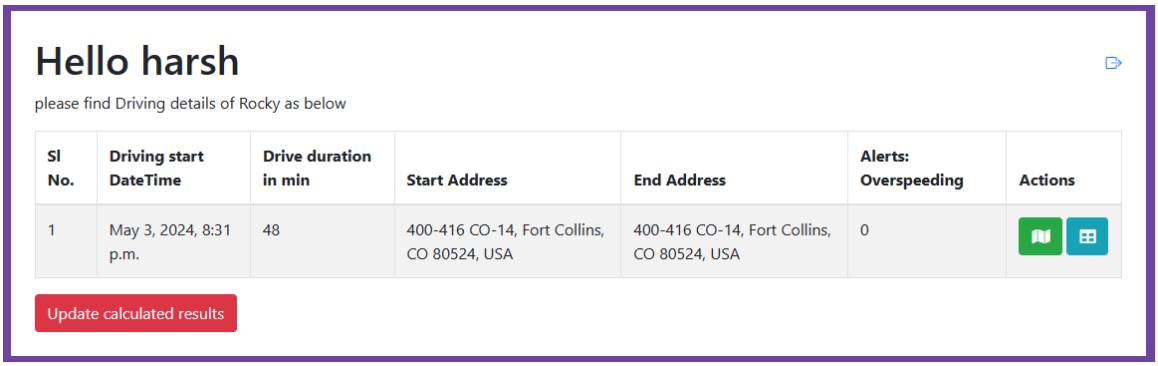

FIGURE 23:USER 2 WITH CSU DATASET2 WITH WINDOW SIZE 10 (1sec)

The evaluation of the Parent Dashboard's customization and personalization options, along with the effectiveness of alert messages, revealed a significant versatility in catering to the unique needs of different users, even amidst varying locations and driving behavior. Despite diverse driving environments and behaviors, the dashboard consistently provides relevant and insightful data, as inferred from the graphs. This reaffirms the expectation that the dashboard effectively accommodates diverse user profiles, ensuring a personalized and informative experience.

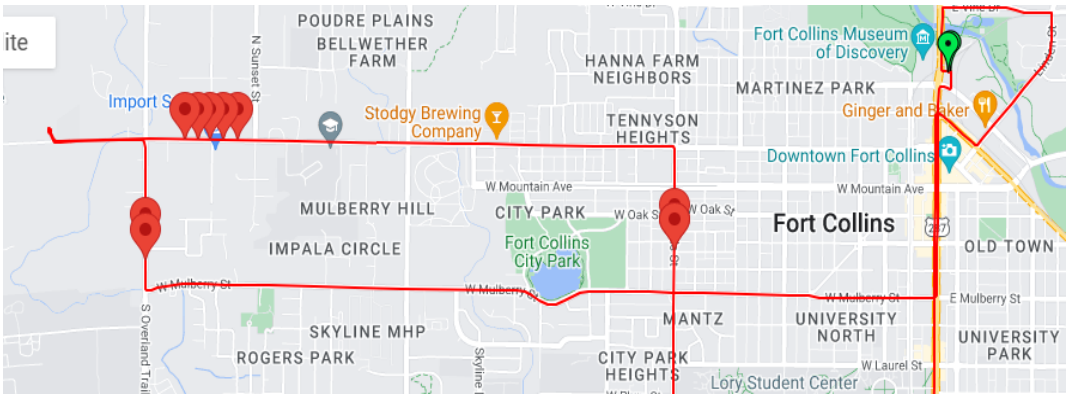

FIGURE 24: PRESENCE OF NINE OVERSPEEDING EVENTS PINPOINTED ON THE MAP CORRESPONDING TO USER 1'S ACTIVITY.

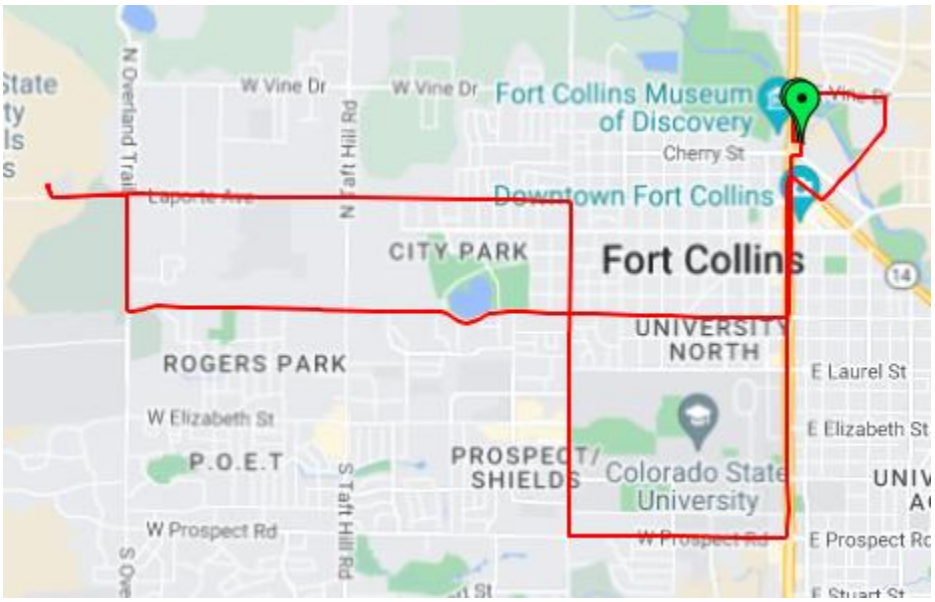

FIGURE 25: ABSENCE OF ANY OVERSPEEDING EVENTS CORRESPONDING TO USER 2'S ACTIVITY

In Figure 24, the map displays nine over speeding events marked for User 1, as reported by the dashboard. Conversely, Figure 25 depicts zero over speeding events for User 2. These figures visually represent the over speeding incidents detected by the dashboard for different users, showcasing marker points indicating the areas where these events occurred. This project not only quantifies the number of alerts presented on the dashboard but also visually confirms their occurrence through marker points on the map interface. Such

detailed mapping of over speeding events enhances the dashboard's effectiveness in providing actionable insights to parents regarding their children's driving behavior.

The analysis underscores the Parent Driver Connect feature's efficacy in detecting and addressing over speeding events, contributing significantly to enhanced road safety and user awareness. Despite potential limitations in publicly available datasets, the systems over speeding detection mechanism operates reliably within the provided dataset's constraints. By delivering timely alerts and personalized messages, the system fosters a proactive approach to addressing safety concerns, ultimately enriching the user experience, and promoting responsible driving habits among users.

7.3 User Feedback

Three users provided feedback on the Parent Driver Connect feature, offering valuable insights into its usability, performance, and potential enhancements.

TABLE 3 : USER FEEDBACK

User	Email id	Feedback
Ashish	ag8564@uncw.edu	The UI could be more intuitive, but it was easy to navigate different sections. Maps with over speeding points were easy to understand. Overall, I think it serves its purpose well
Regan Harlacker	Rjh8987@uncw.edu	I like this webpage. I don’t have any negative reviews and liked the forgot password and over speeding feature.

Madhav	mm8760@uncw.edu	Even though the data is huge, there were no delays in loading pages or accessing certain features and there were quick responses. Expects the driver to slow down based on speed limits. Alerts can include route details along with speeding messages. It would be a good tool for parents looking to monitor their children's speeds, routes, and driving times.

User feedback suggests that while the UI is functional, there is room for improvement in intuitiveness. Users found it relatively easy to navigate between different sections of the application, indicating a satisfactory level of usability. However, they highlighted the importance of enhancing UI intuitiveness to further improve the user experience. Additionally, users appreciated the inclusion of maps displaying over speeding points, noting that they were easy to comprehend and provided valuable insights into driving behavior.

Performance-wise, users were generally satisfied with the feature's responsiveness and stability, even when dealing with large datasets. They reported no delays in loading pages or accessing various features, indicating efficient data processing and system performance. However, one user suggested incorporating alerts that provide route details alongside over speeding messages, enabling parents to have a comprehensive understanding of their children's driving behavior. This feedback underscores the importance of providing users with detailed information to facilitate informed decision-making and enhance the feature's effectiveness.

Overall, user feedback indicates that the Parent Driver Connect feature effectively serves its intended purpose of facilitating parental oversight and promoting safer driving habits. While there are areas for improvement, such as UI intuitiveness and alert customization, users recognize the feature's utility and potential for further enhancement. Incorporating user feedback into future iterations of the feature can lead to a more user-centric and robust solution, aligning closely with user expectations and requirements.

CHAPTER 8: CONCLUSION AND FUTURE WORK

In conclusion, the Parent Drive Connect project represents a significant advancement in enhancing the safety and monitoring capabilities of driving experiences, particularly concerning parental oversight. By leveraging technologies such as Django, HTML, and JavaScript, the project successfully provides a platform for parents to track and analyze their children's driving behaviors. Through the integration of APIs and database management, crucial data regarding driving patterns, locations, and speed limits are efficiently collected, processed, and presented to users via intuitive user interfaces.

Moving forward, there are several avenues for future work and improvement within the Parent Drive Connect project:

a) Enhanced Data Analytics: Implementing advanced data analytics algorithms to derive deeper insights from the collected driving data, such as identifying patterns of risky behavior or optimizing driving routes for efficiency and safety.

b) False Positives: In future work, it is imperative to address the issue of false positives arising from the reliance solely on static speed limits provided by maps. For instance, if a traffic signal indicates green and a driver maintains consistent vehicle speed, yet encounters a junction with lower speed limits, it may lead to erroneous

detections of compliance. Another example occurs within school zones where the speed is reduced during drop-off or pickup. However, the maps show the transient lower speed limit. Therefore, a false positive may occur when the driver is driving at the posted speed in a school zone outside of the transient speed period. The maps do not record multiple speed limits and their conditions. The maps will require an update to accommodate transient speed reduction periods. To mitigate such occurrences, incorporating real-time data sources, such as live traffic conditions could significantly enhance the accuracy and reliability of the system. Additionally, integrating advanced sensor technologies and machine learning algorithms capable of recognizing contextual cues, such as road signs and traffic patterns, may further refine the decision-making process.

c) Integration of Machine Learning: Incorporating machine learning models to predict and prevent potential accidents or hazardous driving situations based on historical driving data and external factors like weather conditions or traffic patterns.

d) Mobile Application Development: Developing a companion mobile application for the Parent Drive Connect platform, allowing parents to receive alerts and notifications about their children's driving activities directly on their smartphones.

e) Incentive Programs: Introducing gamification elements and incentive programs to incentive safe driving practices among young drivers, such as rewarding points or discounts for adhering to speed limits and driving responsibly.

f) Collaborations with Automotive Manufacturers: Partnering with automotive manufacturers to integrate the Parent Drive Connect platform directly into vehicle systems, enabling seamless data collection and monitoring capabilities built directly into the vehicle's onboard systems.

By pursuing these future work points, the Parent Drive Connect project can further evolve and expand its capabilities, ultimately contributing to improved road safety and enhanced parental oversight in the realm of teen driving.

CHAPTER 9: LESSONS LEARNT

The lessons learned from the Parent Drive Connect Django Project offer valuable insights for developers embarking on similar endeavors in web-based application development and data management. By prioritizing data display, dynamic map plotting, speed threshold alerts, separate access key acquisition, and data table maintenance, projects can enhance user experience, and improve overall system performance. These lessons underscore the importance of continuous learning and adaptation in the ever-evolving landscape of web development. This paper outlines five critical lessons learned, including the utilization of streaming HTTP for data display, the integration of JavaScript with HTML for dynamic map plotting, the implementation of speed threshold alerts, the importance of acquiring separate access keys for Google Maps APIs, and the necessity of maintaining distinct data tables for result visualization and driving dataset input.

9.1 Utilization of Streaming HTTP for Data Display:

One of the primary lessons learned from the Parent Drive Connect Django Project was the significance of employing streaming HTTP for displaying data on web pages. Traditional HTTP requests often result in latency issues, particularly when dealing with dynamic data streams such as vehicle tracking. By utilizing streaming HTTP protocols, such as Server-Sent Events (SSE) or WebSocket, the project achieved seamless data updates on the user interface, enhancing the overall user experience.

9.2 Integration of JavaScript with HTML for Dynamic Map Plotting:

Another critical lesson pertained to the integration of JavaScript with HTML for dynamic map plotting instead of relying solely on frameworks like Folium. While Django offers robust backend capabilities, leveraging JavaScript libraries such as Leaflet.js or Mapbox.js empowered the project to create interactive and customizable maps directly within the web browser. This approach facilitated visualization of driving routes and speeding incidents, enhancing user engagement and data interpretation.

9.3 Implementation of Speed Threshold Alerts

The project highlighted the importance of implementing speed threshold alerts to notify users when vehicle speeds exceeded predefined limits. As driving conditions vary, it is essential to establish appropriate thresholds for issuing alerts, balancing the need for safety with practical considerations. By incorporating speed threshold alerts, parents could receive timely notifications, enabling them to intervene and address potential safety concerns effectively.

9.4 Acquisition of Separate Access Keys for Google Maps APIs:

A key lesson learned was the necessity of acquiring separate access keys for Google Maps APIs to access geolocation services. This practice ensures compliance with usage limits and facilitates more efficient management of API calls. By obtaining dedicated access keys for specific project requirements, the Parent Drive Connect Django Project maintained reliability and scalability in accessing geospatial data for mapping and location-based functionalities.

9.5 Maintenance of Different Data Tables for Results and Input Driving Dataset:

Finally, the project emphasized the importance of maintaining separate data tables for storing results and input driving dataset. This organizational structure streamlines data management processes, facilitating data retrieval, analysis, and visualization. By segregating result data from the original driving dataset, the project ensured data integrity and optimized database performance, contributing to a more robust and scalable system architecture.

9.6 Integration of Database with the Project:

One of the initial challenges in integrating a database with a Django project is selecting an appropriate database management system (DBMS) and configuring it within the Django framework. Django supports multiple database backends, including PostgreSQL, MySQL, SQLite, and Oracle. Choosing the right database backend entails considerations such as scalability, performance, compatibility, and project requirements. Additionally, configuring database settings in the Django project's settings.py file, including specifying database engine, host, port, username, password, and other parameters, requires careful attention to ensure proper connectivity and functionality.

9.7 Conversion of Vehicle Speed Measurements:

Additionally, the project required the need to convert original vehicle speed measurements from kilometers per hour to miles per hour. Aligning with OpenStreetMap's provision of speed limits in miles per hour, this conversion ensured consistency and accuracy in speed-related analysis and alerts within the platform.

REFERENCES

[1] https://www.libertymutualgroup.com/about-lm/news/articles/liberty-mutual-insurance-and-sadd-teen-driving-survey
[2] National Highway Traffic Safety Administration (NHTSA). Traffic Safety Facts 2020 Data: Young Drivers (Report No DOT HS 813 313). Washington, DC: U.S. Department of Transportation, National Highway Traffic Safety Administration, National Center for Statistics and Analysis; June 2022.
[3] The Children's Hospital of Philadelphia Center for Injury Research & Prevention (2009). Driving through the eyes of teens a closer look, second in a series of teen driver safety issues. https://www.teendriversource.org/ Accessed 11-25-2020.
[4] W. Zeng, M. A. S. Khalid and S. Chowdhury, "In-Vehicle Networks Outlook: Achievements and Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1552-1571, third quarter 2016, Doi: 10.1109/COMST.2016.2521642
[5] Dimitrios Rimpas, Andreas Papadakis, Maria Samarakou, OBD-II sensor diagnostics for monitoring vehicle operation and consumption, Energy Reports, Volume 6, Supplement 3,2020, ISSN 2352-4847,https://doi.org/10.1016/j.egyr.2019.10.018.
[6] Noriyuki Kushiro, Yusuke Oniduka, Yoichi Sakurai, Initial Practice of Telematics-Based Prognostics for Commercial Vehicles: Analysis Tool for Building Faults Progress Model for Trucks on Telematics Data, Procedia Computer Science, Volume 112, 2017,, ISSN 1877-0509,https://doi.org/10.1016/j.procs.2017.08.244.
[7] A. V. Sant, A. S. Naik, A. Sarkar and V. Dixit, "Driver Drowsiness Detection and Alert System: A Solution for Ride-Hailing and Logistics Companies," *2021 IEEE Pune Section International Conference (PuneCon)*, Pune, India, 2021, pp. 1-5, Doi: 10.1109/PuneCon52575.2021.9686546.
[8] B. Warwick, N. Symons, X. Chen and K. Xiong, "Detecting Driver Drowsiness Using Wireless Wearables," *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, Dallas, TX, USA, 2015, pp. 585-588, Doi: 10.1109/MASS.2015.22.
[9] Franzini, M.; Annovazzi-Lodi, L. and Casella, V. (2020). Assessment of the Completeness of OpenStreetMap and Google Maps for the Province of Pavia (Italy). In Proceedings of the 6th International Conference on Geographical Information Systems Theory, Applications and Management - GISTAM; ISBN 978-989-758-425-1; ISSN 2184-500X, SciTePress, pages 270-277. DOI: 10.5220/0009564302700277
[10] P. Boottho and S. E. Goldin, "Automated evaluation of online mapping platforms," *2017 International Electrical Engineering Congress (iEECON)*, Pattaya, Thailand, 2017, pp. 1-4, Doi: 10.1109/IEECON.2017.8075809
[11] S. Mostafi and K. Elgazzar, "An Open Source Tool to Extract Traffic Data from Google Maps: Limitations and Challenges," *2021 International Symposium on Networks, Computers and Communications (ISNCC)*, Dubai, United Arab Emirates, 2021, pp. 1-8, Doi: 10.1109/ISNCC52172.2021.9615680.
[12] Vrany, J., Krepelka, M., Chumlen, M. (2023). Generating Synthetic Vehicle Speed Records Using LSTM. In: Maglogiannis, I., Iliadis, L., MacIntyre, J., Dominguez, M. (eds) Artificial Intelligence Applications and Innovations. AIAI

2023. IFIP Advances in Information and Communication Technology, vol 675. Springer, Cham.

[13] Vanden Hautte, S., Moens, P., Van Herwegen, J., De Paepe, D., Steenwinckel, B., Verstichel, S., Ongenae, F., & Van Hoecke, S. (2020). A dynamic dashboarding application for fleet monitoring using semantic web of things technologies. *Sensors, 20*(4), 1152. https://doi.org/10.3390/s20041152

[14] Fleet Connect. "Optimizing Fleet Management " 2023. https://www.fleetconnect.com

[15] Verizon Connect. (2024). Fleet Management Solutions. Retrieved April 26, 2024, from https://www.verizonconnect.com/solutions/fleet-management/

[16] AutoPi: Telematics IoT Infrastructure for Fleets and developers - Built on Raspberry Pi. (n.d.). AutoPi.io. https://autopi.io/

[17] Lapetus Solutions Inc: www.LapetusSolutions.com

[18] Google. (2024). Geocoding API. Retrieved April 26, 2024, from https://developers.google.com/maps/documentation/geocoding/start

[19] OpenStreetMap Contributors. (2024). Overpass API,from https://wiki.openstreetmap.org/wiki/Overpass_API

[20] Svennerberg, G. (2010). Beginning Google Maps API 3. In *Apress eBooks*. https://doi.org/10.1007/978-1-4302-2803-5

[21] Google Developers. (n.d.). Geocoding API. Retrieved from https://developers.google.com/maps/documentation/geocoding/overview.