Integrating an AI-Driven Chatbot for Natural Language Traffic Safety Analysis within the SAFE-AL Dashboard

by

Keanna Tennyson

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 8, 2026

Keywords: SAFE-AL Dashboard, AI chatbot interface, conversational analytics, natural language processing (NLP), large language models (LLM), traffic safety analytics

Approved by

Xiao Qin, Chair, Alumni Professor of Computer Science and Software Engineering
Laurence Rilett, Co-chair, Distinguished Professor of Civil and Environmental Engineering
Adrian Cottam, Assistant Research Professor of Civil and Environmental Engineering
Cheryl Seals, Professor of Computer Science and Software Engineering

Abstract

This thesis presents the design and implementation of an AI chatbot by integrating into the existing Safety Analytics For Evaluating Alabama (SAFE-AL) dashboard to give users another way to interact with and understand transportation data. The chatbot functions as a natural language query interface, helping users interact with and navigate traffic safety datasets without requiring technical knowledge of dashboard filters, SQL queries, or geospatial tools.

Through natural language inputs such as “Show me Jefferson County in January 2025?”, the chatbot interprets the user’s request through a combination of both natural language processing and rule-based parsing and large language model use to navigate, understand and inform the user on such topics as time, location, and roadway. The pulled filters are then reviewed and sent through dashboard’s existing callbacks module, stored procedures, and user interface to produce results. Rather than creating SQL queries or directly accessing the database, the chatbot operates as a navigational and conversational tool for the SAFE-AL dashboard. The chatbot remains read-only and uses the existing dashboard logic instead of creating new queries or changing the transportation data. In addition to text responses, the chatbot can automatically zoom to selected counties, highlight queried results on the dashboard map, and open related charts leveraging the existing callback logic already present in the dashboard.

The research uses quantitative evaluation methods. Quantitative analysis compares Google Gemini and OpenAI ChatGPT using filtering accuracy, consistency with dashboard results, intent and understanding accuracy, hallucinations, and response time. The study also compares chatbot response times with manual dashboard interactions and reviews the cost effectiveness of both models. The study analyzes whether conversational analytics can improve how users interact with transportation safety data while maintaining reliability, ease of use, and performance.

Artificial Intelligence (AI) Use Disclosure Statement

In the preparation of this thesis, the following Artificial Intelligence (AI) tools were used: ChatGPT. This tool was used primarily to assist with punctuation, LaTeX formatting, citation formatting and formatting review. Google Gemini and OpenAI ChatGPT as research models are not disclosed under this statement because they are the subjects of the research rather than tools used in the preparation of this thesis. The author acknowledges full responsibility for the intellectual content of this work and has ensured that all AI-assisted sections have been reviewed and revised for accuracy and appropriate academic style. All AI-generated content was reviewed and validated for relevance, appropriateness, and accuracy before incorporation into the final document to maintain the scholarly integrity of this research.

Digital Accessibility Disclosure Statement

In the preparation of this thesis, the following digital accessibility tools were used to ensure this document complies with federal requirements: LuaLaTeX and the PDF Accessibility Checker (PAC). The author acknowledges full responsibility for the intellectual content of this work and has made a good faith effort to comply with digital accessibility requirements in publishing, wherein the nature of the content does not significantly change in order to do so. Furthermore, all content has been reviewed and revised to meet these requirements prior to final publication.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In transportation, one of the most critical roles is the roadway system because it supports the economy and daily travel. Although transportation systems and vehicle technologies continue to improve, traffic crashes remain a significant safety concern. In the United States, 39,254 people were killed and an estimated 2,422,195 people were injured in motor vehicle traffic crashes during 2024 [1]. One major factor contributing to these crashes is speeding because of its severity and frequency, making it one of the most important concerns for transportation and law-enforcement agencies [2].

To help combat these challenges, agencies have now focused on data driven safety improvements and interactive tools to explain transportation data to better understand roadway conditions and identify unsafe locations. The implementation and development of web-based dashboards have made it possible for users to visualize and interact with large transportation datasets. These tools help transportation professionals, researchers, and law enforcement agencies by showing patterns in speeding, travel time reliability, and crash activity and severity in multiple locations and time periods, helping the user to make more informed decisions. Transportation data becomes easier to interpret when it is presented through interactive visualizations that allow users to explore trends across different locations and time periods [3]. The SAFE-AL dashboard was designed to help further research by incorporating National Performance Management Research Data Set (NPMRDS) data and Alabama crash data into a single platform that handles both geospatial and modular analysis [2, 4].

A key limitation of current web-based dashboards is the requirement for manual selection of all filters and settings, making it difficult to quickly obtain insights. This process can be time-consuming for first-time users because they must manually use the filters, maps, and charts to find the transportation data they need. As transportation datasets continue to expand and become more complex, there is an increasing need for more natural and efficient ways to interact with these systems, to reduce the learning curve.

One possible solution is the integration of artificial intelligence, specifically a natural language-based chatbot that allows users to request dashboard data using natural language. Instead of manually adjusting multiple filters, a user could ask questions such as "where does the crash data come from?" or "Show the county with the highest speeding in 2025." The system would then interpret the request and return results through the existing dashboard architecture.

This research proposes the implementation and development of an AI chatbot that integrates with the existing SAFE-AL dashboard to give users a natural language method for interacting with transportation data. The chatbot is built using a structured pipeline, where user input is first processed through a natural language parser to pull key information such as date, time, county, and roadway type. The request is then sent through a controlled backend architecture that reuses the dashboard's existing filtering logic, stored procedures, and visualization callbacks. This design allows the chatbot to remain read-only, preventing unauthorized query creation or changes to existing data while maintaining consistency with validated dashboard outputs.

In addition to creating text responses, the chatbot can also trigger dashboard actions such as updating filters, zooming to selected counties, highlighting queried map sections, and opening charts through the same approved callback logic already used by the dashboard. The system is also provided context regarding the current state of the dashboard, allowing users to ask follow-up questions based on what is displayed on the screen. This allows users to interact with transportation data through natural language while continuing

to use the existing dashboard logic for filters, maps, and charts.

To test the reliability and accuracy of the chatbot's response, this study uses quantitative evaluation methods. The quantitative analysis compares Google Gemini and OpenAI ChatGPT using filtering accuracy, intent and understanding accuracy, hallucinations, consistency with dashboard outputs, and response time. This study evaluates how natural language interaction can be integrated into the existing SAFE-AL dashboard while maintaining dashboard accuracy and comparing model performance.

Chapter 2

Literature Review

Recent research in artificial intelligence, natural language processing, and transportation analytics has improved how users interact with transportation data and dashboards. As transportation datasets continue to grow in size, dashboards and manual query systems can become difficult for beginner or non-technical users to use. Because of this, researchers have explored the use of AI models, natural language interfaces, and interactive analytics tools to make raw complex data easier to access and understand. This chapter reviews prior work related to artificial intelligence in transportation dashboards, traffic analytics, natural language understanding, text-to-SQL systems, conversational analytics, and the trustworthiness and accuracy of AI systems.

2.1 Artificial Intelligence in Transportation Systems

Artificial intelligence has become an important part of transportation research because it can help research teams analyze large amounts of data, predict traffic conditions, improve safety, and support finding viable solutions. Iyer [5] explains that AI is used across intelligent transportation systems for traffic management, public transportation, and safety management. The paper also shows that AI can support tasks such as predicting congestion, routing, signal control, monitoring driver behavior, and transportation planning [5]. Jevinger et al. [6] give a more in-depth overview of artificial intelligence in public transportation. Their mapping study shows that most AI applications in this area are used for prediction, current state estimation, planning, scheduling, and resource allocation. Only

two of the reviewed studies focused on automation, while the others used AI to provide different types of decision support [6]. This suggests that AI is mainly used to support human decisions rather than replace them, allowing transportation professionals to make better use of the large amounts of data generated within these systems [6]. Saki et al. [7] go into further detail by explaining artificial intelligence, machine learning, and deep learning applications across advanced transportation systems. Their work covers traffic management, public transit optimization, smart parking, freight logistics, safety, and infrastructure monitoring. They also point out common challenges such as data quality, privacy, scalability, bias, and model generalization. Saxena [8] also explains how artificial intelligence can improve traffic management through applications such as traffic prediction, incident management, and route recommendations. The study also highlights challenges with data quality, privacy, cybersecurity, and the integration of AI with existing transportation systems. Fahim et al. [9] also discuss how artificial intelligence can be used for traffic management, vehicle monitoring, accident detection, and other applications that support safer and more efficient transportation systems.

These studies focus on the use of artificial intelligence for prediction, traffic management, safety, and other transportation applications. These applications use AI to support different transportation tasks, but do not focus on using natural language to interact with an existing transportation dashboard. The SAFE-AL chatbot builds on this research by adding natural language interaction to the existing dashboard.

2.2 Transportation Dashboards and User-Interface Analysis

Transportation dashboards are useful because they allow users to explore roadway data through maps, charts, and filters. The paper written by Tennyson et al. [2] is the most directly related source because it gives a basis for the SAFE-AL dashboard. The paper also describes a low-cost analytics dashboard that combines NPMRDS speed and travel time data with Alabama crash data using open-source Python tools such as Dash

and Plotly. The existing dashboard supports geospatial analysis, roadway speed exploration, travel time reliability, and safety performance metrics. However, users still have to manually interact with the filters, maps, and charts. The artificial intelligence chatbot builds on the existing system by giving users a natural language way to interact with the same dashboard logic [2].

Other transportation dashboard research also focuses on how users interact with and understand the data being displayed. Tantillo et al. [10] explain that transportation dashboards should be designed around user needs and the operational questions they are trying to answer. Other research focuses on how dashboards can help users identify traffic patterns, abnormal conditions, and trends across different locations and time periods [3]. Zerafa et al. [11] focus more specifically on helping users understand traffic incident data. Jung et al. [12] developed a web-based platform that displays traffic simulation results through an interactive dashboard. Although these systems use different types of transportation data, they each use visualization tools to help users explore and understand the information being displayed. The research in this section shows how transportation dashboards can help users navigate and understand different types of transportation data. The Regional Integrated Transportation Information System (RITIS) is one such platform and provides interactive dashboards that incorporate maps, charts, and filtering tools for exploring transportation data [13]. Users still need to interact with these visualizations and filters to locate specific information, which gives the SAFE-AL chatbot another way to interact with the existing dashboard.

2.3 Natural Language Understanding for Dashboard Queries

Understanding natural language is one of the most important things when working with a chatbot, because the chatbot has to understand what the user wants before it can update the dashboard. Weld et al. [14] explain that two major parts of natural language understanding are intent detection and slot filling. Intent detection identifies the user's

goal, while slot filling identifies the information needed to complete that goal [14]. For the SAFE-AL chatbot, this information can include details such as location, time, roadway, or vehicle type. Both tasks relate to the SAFE-AL chatbot because the system needs to identify what the user wants to do and pull the information needed to update the dashboard. Dharmireddi et al. [15] also support the use of modern language models by explaining how GPT-based systems improve contextual understanding, ambiguous phrasing, and conversational responses. This relates to the SAFE-AL chatbot because users may ask for the same dashboard information using different wording or ask follow-up questions based on the current conversation.

Natural language interfaces have been studied as a way to help users ask data questions without writing code. NL4DV (Natural language-Driven Data Visualization) is a Python toolkit that converts natural language queries into visualization specifications. It is especially useful because it translates natural language requests into structured visualization information, including data attributes, analytic tasks, and visualization specifications, as described by Narechania et al. [16]. This is similar to the SAFE-AL chatbot, which also converts the user's natural language request into structured information before updating the dashboard. Mitra et al. [17] also supports this idea by focusing on conversational interaction in visualization systems, including follow-up questions and handling unclear requests. Chen et al. [18] also show how natural language can be used to generate interactive visualization interfaces. These studies combine natural language interaction with data visualization in different ways. The SAFE-AL chatbot uses a similar approach but applies natural language interaction to the existing filters, callbacks, and stored procedures of a transportation safety dashboard.

2.4 Text-to-SQL and Conversational Dashboards

Text-to-SQL research is related to this study because both systems have to understand a natural language question and convert it into a structured action that can retrieve data.

Yu et al. [19] used Spider to evaluate how well models translate natural language questions into SQL across different database schemas. Later research focused on improving different parts of this process. RAT-SQL improved schema encoding and the connection between user questions and database structure Wang et al. [20], while PICARD improved constrained decoding to reduce invalid SQL generation Scholak et al. [21]. Further improvements were introduced when Sen et al. [22] also showed that natural language systems can handle more complex business intelligence queries. While Gao et al. [23] evaluated large language models for Text-to-SQL tasks by comparing different prompting strategies on effectiveness, efficiency, and cost. Their study also shows that understanding natural language questions and generating correct SQL queries remain challenges for these systems. The Text-to-SQL research discussed in this section shows that converting natural language into database queries remains challenging because of ambiguity in user requests, differences in database structures, and the generation of invalid SQL queries. It also recognizes approved filters from the user's question and sends them through the existing SAFE-AL dashboard logic. This gives users natural language interaction while keeping the AI model separate from direct database access.

2.5 Trustworthiness, Accuracy, and Data Analysis

Trustworthiness is important for the SAFE-AL chatbot because the user needs to know that the generated response matches the validated data shown by the dashboard. Narechania et al. [24] explain this issue through their do it yourself system, which helps users determines whether answers from natural language-to-SQL systems are correct. Their work focuses on giving users additional information to help them understand and evaluate how the system produced an answer. Zhong et al. [25] also show that evaluating text-to-SQL systems is difficult because surface level accuracy can miss deeper semantic errors. This relates to the evaluation of the SAFE-AL chatbot because understanding the user's question is only one part of its performance. The chatbot also needs to return results that

match the existing dashboard and respond within a reasonable amount of time. These challenges are also important to the design of the SAFE-AL chatbot because incorrect results could affect how users understand speeding patterns, crash activity, or roadway performance. The chatbot remains read-only and uses the existing stored procedures and dashboard callbacks to provide natural language access without changing the existing data or creating new invalidated queries. This approach helps minimize ambiguity, prevents invalid SQL generation, and ensures that the chatbot uses only approved filters and existing dashboard logic.

Chapter 3

System Architecture

3.1 Existing Dashboard Application

The existing dashboard infrastructure is built on a combination of large-scale transportation datasets, structured database design, and interactive web-based visualization tools. The system primarily relies on the National Performance Management Research Data Set (NPMRDS), which provides detailed speed and travel time data at the Traffic Message Channel (TMC) level within five-minute intervals [4]. This dataset is combined with crash data from the Alabama CARE system to support a more complete understanding of roadway performance and safety across different locations and time periods [26]. These datasets serve as the foundation of the dashboard and allow users to explore patterns in speed, travel time reliability, and crash activity across the state [2]. By showing transportation data through maps and interactive visualizations, geospatial dashboards help users navigate roadway conditions and better understand patterns in transportation data [3].

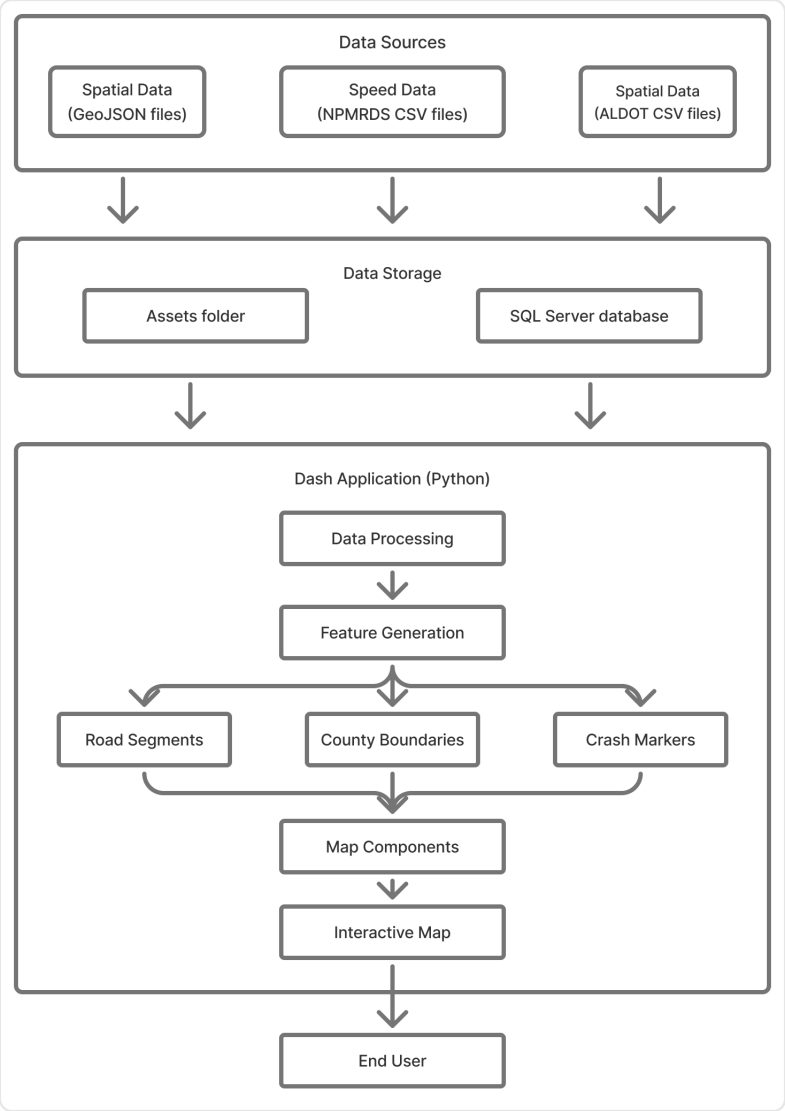

Figure 3.1: SAFE-AL Dashboard Architecture

Figure 3.1 Is the existing SAFE-AL Dashboard Architecture, the figure displays the three main layers of the SAFE-AL dashboard, including the data sources, data storage, and the Dash application layer, and shows how each layer connects to show the interactive

map displayed to the user.

To handle the size and complexity of this data, a Microsoft SQL Server Database is used for storage and retrieval. The raw data is first imported from CSV files into structured tables, including separate tables for commercial motor vehicles (CVs), non-commercial vehicles (CMVs), and combined traffic data, along with crash and reference tables. To improve performance, the system uses aggregated summary tables that stores daily, weekly, monthly, and yearly averages. These tables, along with indexed queries and stored procedures, improve the speed of real-time calculations and allow the dashboard to respond quickly to user inputs. Precomputed date ranges are also used in certain parts of the dashboard to reduce loading times for map queries. This design makes it possible to work with very large datasets while still maintaining fast and reliable performance [2].

On the front end, the dashboard is built using Dash and Plotly, which allow for building interactive applications entirely in Python. The layout defines the structure of the interface, while callback functions connect user inputs, such as dropdowns and date filters to back-end processes. When a user makes a selection, the application queries the database and updates the visualizations automatically. Geospatial features are supported using Dash Leaflet, which renders map layers from GeoJSON files, including roadway segments and county boundaries. The dashboard also maintains multiple client-side stores that help with the current filter state, selected counties, selected roadways, and readily available crash data for the user to interact with. This setup allows users to explore the data through both maps and charts, making it easier to identify trends and problem areas. Overall, the system connects the chatbot to the existing SAFE-AL dashboard while keeping the dashboard's original data procedures and filtering logic. [2].

3.2 Chatbot Integration Layer

The chatbot integration layer is the connection between the user's natural language request and the existing SAFE-AL dashboard. Rather than users having to manually ad-

just filters, select troops, counties, or roadways, or navigate between maps and charts, the chatbot lets users interact with the dashboard using natural language. The goal of this chatbot integration layer is not to replace the existing dashboard logic but to improve it by allowing users to access the same information through natural language questions and dashboard requests. Research on transportation dashboards show that users typically need two things from a dashboard: the ability to make real-time decisions and the ability to ask deeper analytical questions that reveal patterns and performance across the network Tantillo et al. [10]. The AI chatbot was designed with these needs in mind, allowing users to update dashboard filters through the use of natural language while also being able to ask insight questions about speeding, crash and travel time reliability.

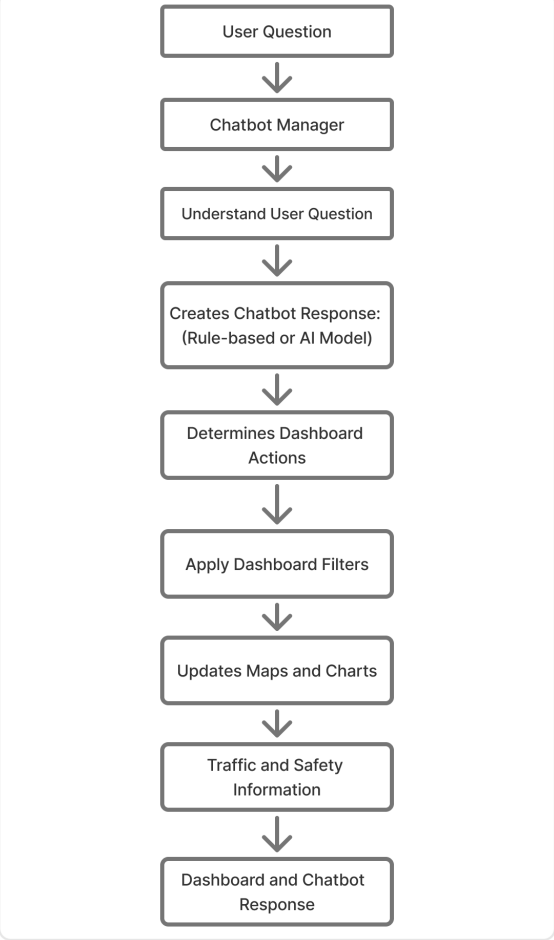

Figure 3.2: Chatbot Architecture

Figure 3.2 displays the Chatbot Architecture and the steps the chatbot takes from the user's question to returning a dashboard and chatbot response, including understanding the question, creating a response based on either the rule-based parser or the selected AI models, then determining dashboard actions, and finally applying the necessary filters to update either the map or charts.

When a user submits a question, the chatbot first processes the message through

a rule-based parser that pulls pre-existing information without the need for an external API call, but it has clear limitations. It can only pull information it was programmed to look for, and it cannot understand indirect phrasing, context, or questions that combine multiple intentions in a single sentence. Natural language dashboards rely on finding important information from the user's question, such as attributes, filters, and intent, before outputting a response Narechania et al. [24]. In the SAFE-AL chatbot, this information filters such as county, roadway, date range, time range, vehicle type, which are then used to determine the next actions for the dashboard. This is where the LLM becomes useful because it can handle requests that the rule-based parser cannot. When a user asks a question that uses vague phrasing, references something already on screen, or uses language that does not map directly to a named filter, the LLM reads the full question along with the conversation history and the current dashboard state, and determines how to respond to the user. The LLM also creates all of the natural language responses that appear in the chat panel, including explanations of what metrics mean, descriptions of what the user will see after a filter update, and answers to general questions about the data. Without the LLM, the chatbot would be limited to confirming that a filter was applied with no way to explain it, and it would fail on any question that did not match a known pattern in the parser. The rule-based parser and LLM have different roles in the chatbot. The parser handles structured questions and known filter patterns, while the LLM processes questions that require more natural language understanding,

If the user selects one of the models in the drop-down, the question is run through the LLM along with the full conversation history and a real-time description of the current state of the dashboard. This gives the LLM the information it needs to understand follow-up questions, and interpret language that does not map to a filter.

The full text of the SAFE-AL research paper is also loaded when the application starts to provide the LLM with additional information about the dashboard's structure, purpose, and supporting datasets. This gives the LLM a detailed understanding of the dashboard's

data sources, methodology, and metrics without requiring it to query any database. As a result, the LLM can answer questions about how speeding is calculated, what LOTTR means, where the NPMRDS data comes from, and what the difference between CMV and non-CMV vehicle types represents, all using information pulled into its context rather than getting this information from an unreliable source. This is important because users should be able to select either CMV or non-CMV vehicle data when interpreting the results, as previous research has shown that statewide speed estimates can be influenced by commercial vehicle traffic [27]. This design allows the chatbot to be used as both a navigation tool and a conversational assistant within the same interface.

The LLM does not interact with the database directly at any point. Instead, it returns a structured JSON object that specifies which dashboard filters to apply, such as county, troop, date range, time range, vehicle type, highway, analysis type, or crash status. This output is then checked in a validation process before anything is written to the dashboard. The JSON is first checked against an approved schema to remove any unrecognized values, then checked to confirm that the target page and filter structure are correct, and finally checked against allowlists that restrict filter values to those only used in the existing dashboard. Counties and roadways are matched against approved state data, date and time values are checked against the approved ranges in the dashboard, and any unrecognized or unsafe values are rejected before they can affect the dashboard. If information is missing, such as a date range that is needed before the map can load, the chatbot asks a follow-up question and waits for the user's response before updating the rest of the filters. In this design, the LLM handles natural language while the existing dashboard logic controls how filters and dashboard actions are applied.

3.3 Chatbot Data Process

The chatbot does not create, modify, or delete queries as this is an intentional decision that protects the transportation data and ensures nothing is changed within the database.

Instead, it works by using the existing dashboard's system, such as stored procedures, summary tables and preprocessed datasets, that already existed in the dashboard before the chatbot was added. The chatbot's role in this process is to interpret the user's natural language question into the approved filter values that those existing processes already accept, and to pass those values through the same dashboard logic that a user would trigger manually.

For most questions involving filtering by county, troop, roadway type, vehicle category, date range, time range, or analysis type, the chatbot pulls the key values from the user's question and updates the filters accordingly. Because these filtering options were part of the pre-existing dashboard design, the chatbot can apply them without requiring any additional database use. The same callbacks and stored procedures that run when a user manually selects a county from the dropdown or moves the time slider are the same ones that run when the chatbot sets those values. When the chatbot correctly identifies and applies the requested filters, the dashboard uses the same existing logic as when a user manually applies those filters. This is an important part of the tool intended for use in transportation safety research and law enforcement.

For questions requesting roadway performance measures, speeding metrics, and travel time reliability analysis, the dashboard continues to use the same callback logic, and summary tables. When a user asks the chatbot to show travel time reliability data for a specific year, for example, the chatbot sets the analysis type and date range and the existing dashboard system handles the data and populates it on the map using the same path it always has. This consistency ensures that the outputs can be comparable to those created when updated manually, which is important in a testing aspect.

The chatbot can also answer insight-based questions by evaluating the data that is already loaded into the dashboard rather than needing to create a new query. When a user asks which county currently shows the highest speeding activity, or which area has the most crashes, the chatbot reads the GeoJSON feature data that is already loaded in

the dashboards Dash stores, compares the results, and returns a summary as well as updating the map if needed. Because the chatbot applies the filters for the user, the user can request transportation data without manually interacting with the dashboard filters.

For more knowledge-based questions about the dashboard, such as metrics, data sources, and methodology, the LLM uses information from the SAFE-AL research paper to interpret the context and explain it to the user. This helps the LLM to answer questions about how speeding is calculated, what the LOTTR metric is and how to interpret it, where the NPMRDS and Alabama CARE data come from, and what is the difference between CMV and Non-CMV vehicle types. The combination of live dashboard data for insight questions and the integrated paper helps the LLM to answer questions and allows the chatbot to aid in wide range of tasks the user may need. While staying a read-only system.

Overall, this design allows the chatbot to reuse the existing dashboard system while preventing the AI models from directly creating queries or modifying the transportation data. Because all outputs work based on approved dashboard actions and with no queries being made, users receive consistent results whether they interact with the dashboard through the chatbot or manually toggle the filters.

3.4 The Chatbot Response

When the chatbot has finished interpreting and processing a user's question, the results are shown as two outputs that work together to give the user both a visual update and a natural language summary of what they are seeing. The first is a text based response shown in the chat box that is created by the LLM to explain the dashboard results through natural language. Rather than confirming that a filter was updated, the reply explains what the user will now see on the map or chart, as well as what the color scale represents, how the data is calculated, and how the current filters relate to the question that was asked. Conversational questions, such as asking what a specific calculation means or where the data comes from, the chatbot uses the SAFE-AL research paper as additional context to

help reduce inaccurate responses.

When the chatbot identifies a question pertaining to filtering, it confirms the approved filter values to the dashboards existing component stores, which updates the same callback logic that a manual filter interaction would produce. This means the map reloads, the chart updates, or the analysis type updates the same way it would, if a user manually toggled the same filters. Keeping consistency throughout the chatbot and dashboard interactions.

Another use for the chatbot is its understanding of the current dashboard state. On every message submission, the system reads the live values of all active filter components and passes that information into the LLM's context before giving the user a reply. This means the LLM always knows what is currently shown on screen regardless of whether the most recent state was set by the chatbot or by the user manually, which is important for follow-up questions where context from earlier in the conversation needs to be used to answer another question.

Overall, the chatbot improves the SAFE-AL dashboard by adding a conversational AI assistant that allows users to explore transportation data, and understand results through natural language. Because all visual updates continue to rely on the dashboard's pre-existing system, the chatbot uses the same filtering logic and dashboard actions as manual user interactions.

Chapter 4

Methodology

The methodology for this study is designed to evaluate the chatbot's performance within the SAFE-AL dashboard. The chatbot is not built as a separate analytics tool. Instead, it works directly with the existing dashboard by using its current filtering logic, stored procedures, and visualization callbacks. Because of this, the evaluation focuses on how accurately the chatbot processes requests, how closely the results match the dashboard, and how quickly the chatbot responds.

The system follows a structured process where a user enters a question using natural language through the dashboard interface. The request is then passed through the natural language parser, which identifies important details such as county, roadway type, date range, time range, and analysis type. The parser also confirms and processes the values before they are passed into the dashboard's system. When an LLM model is selected, the chatbot uses the parsed request along with the current dashboard state to help create structured filter outputs. The interpreted request is then sent through the approved dashboard tools and the existing callback logic to update filters, maps, charts, and visualizations.

Rather than creating new SQL queries or changing existing data, the chatbot uses the same approved stored procedures and filtering process already used by the dashboard. The chatbot does not directly access the database or create SQL queries. This allows the chatbot to return answers that match what a user would receive if the same filters were selected manually through the interface. The system also includes multiple validation

layers that check the pulled filters and approved actions before any update is applied to the dashboard.

4.1 Quantitative Analysis

The quantitative analysis focuses on measuring how well the chatbot performs by comparing Google Gemini and OpenAI ChatGPT using the same set of transportation-related prompts. These models were selected because they support structured outputs and can process the SAFE-AL dashboard information and conversation history [28, 29]. Both models will be tested using common dashboard tasks such as finding peak rush hour periods, analyzing speeding activity, filtering by county, comparing vehicle types, and selecting roadway categories. Additional prompts will include follow-up questions, incomplete requests, and navigational tasks such as opening charts, and toggling the interactive map. Using the same prompts for both models allows for a fair comparison under the same testing conditions. Each question was tested ten times per model to account for variability in LLM responses and to get a percentage score rather than a single pass or fail result. Since both were tested ten times using the same questions, it was important that the responses were able to vary naturally between runs. To keep the testing condition consistent, both Google Gemini 2.5 Flash and OpenAI GPT-4.1-mini were tested using a temperature setting of 0.5. This let each model have small differences within its responses while remaining consistent enough to evaluate accuracy, response times, and hallucinations. Using the same temperature setting for both models also helped provide a fair comparison throughout testing.

4.1.1 Manual Testing

Before chatbot testing, all filter-based questions were first tested manually once by the user by toggling each filter in the dashboard. The response time was then recorded for

each question. This is the baseline which is used as a comparison to the chatbot response times, to evaluate whether the chatbot was faster or slower than a user manually toggling the filters within the dashboard. The goal of manually testing each question once instead of manually across all ten runs is to record a single reference time for each question rather than to test for variability. This comparison shows how the chatbot's response time compares to manually applying the same filters within the dashboard.

4.1.2 Filter Accuracy

One of the main areas being measured is how accurately the system identifies the correct filters from a user's request. This includes understanding items such as county, roadway type, date range, time range, vehicle category, and analysis type. Since the chatbot depends on understanding the question before returning results, correctly identifying these filters is an important part of the evaluation. Each run was scored using a three-point scale. A score of one was given when both the filters and the description populated correctly and the map updated. Half of a point was given when either the description did not populate or the filters did not update, meaning that one of the two outputs failed. A score of zero was given when neither the filter nor the description populated, and the map did not update. The total score across all the ten runs was then divided by ten to get the percentage score for each question and each model.

4.1.3 Intent and Understanding

Another area being tested is intent and understanding, which assesses how well the chatbot works with conversational type questions. This includes questions about what the user is currently looking at on screen, what specific calculations mean, and where does the data come from. For each question we also tested for hallucinations. A reply from the chatbot was marked as a hallucination if the chatbot gave a response to the user, but the response did not correlate with the output of the map. By loading in the research paper

into the LLM's module. Each of the ten runs per question was checked and then marked as either accurate or a hallucination based on the question. Filter based questions were not included in the hallucination count since those responses are created directly from data already loaded in the dashboard rather than from the LLM.

4.1.4 Response Time

The final area being measured is response time. This looks at how long it takes for the chatbot to process a request and return a final answer. Since dashboards are interactive tools, response time is measured to determine how quickly the chatbot can process requests and return results. Response times were recorded for both the manual user baseline and the chatbot were taken for all ten runs per question, which was used in the calculations for mean and standard deviation to show both the average speed and variability between runs. This helps to show the comparison between the user manually toggling the filters and the chatbot updating the map, to show the difference in chatbot's average response time per question against the user's manual testing. The final results from Google Gemini and OpenAI ChatGPT will be compared across these areas to determine which model performs better for conversational transportation dashboard interaction.

Chapter 5

Implementation

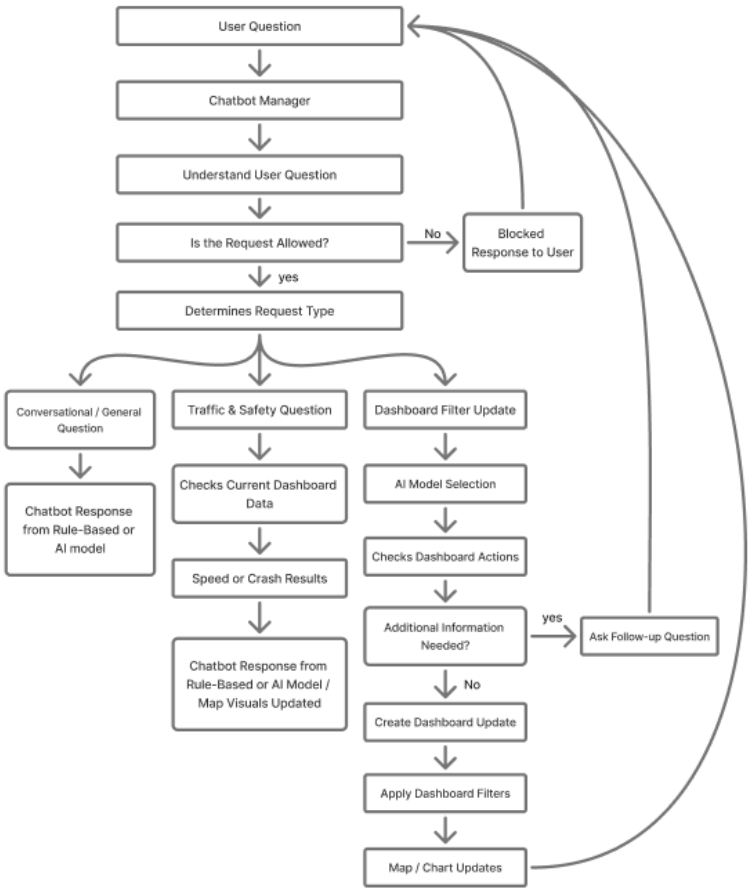

Figure 5.1: Chatbot Flow Diagram

Figure 5.1 shows the overall workflow of the chatbot architecture after it has been integrated into the SAFE-AL dashboard. The chatbot then determines the request type, and either creates a conversational response, explains the current state, or applies dashboard filters, and displays it to the user. The loop-back arrows show how this is a continuous process for user interaction through follow-up questions.

5.1 Development Environment

The chatbot was built using the same development environment as the existing SAFE-AL dashboard, which is written in Python, and the database being written in SQL. The main framework used within the dashboard is Dash by Plotly, which handles the layout, the interactive filters, and the callback processes. Because the chatbot uses the same callback system, it was easier to connect directly with the existing dashboard without needing to use a different programming language, or any additional tools.

With the use of LLM providers, two libraries were incorporated. The Google GenAI library connects to Gemini 2.5 Flash, and the OpenAI Python client connects to GPT-4.1-mini. Both are called when the user toggles the dropdown box to select either one of those models in the chat box. Both models were chosen because they give a good balance between performance and cost while covering the needs of the chatbot. Before replying to the user's question, the models receive the SAFE-AL research paper, the current dashboard state, and the chatbots history, giving them the information needed to understand both the dashboard and the users request. Because each request includes a large amount of information the chatbot requires models that could handle that context while still returning structured JSON for updating the dashboard. Both models met these needs while being cost-effective to use during testing. API keys for both LLM providers are stored in a local environment file and loaded at startup using Python-dotenv, which keeps the API keys safe and secure out of the main source code. The database connection continues to use the existing data handler through pyodbc, which works best as there are no new

database connections needed for the chatbot.

5.2 Chatbot File Architecture

The chatbot is organized into four main Python files, with each having their own use. Chatbot_parser.py has the rule-based parser, which handles all keyword and pattern matching, as well as a reference list, and intent detection before calling either LLM provider. Chatbot_LLMs.py handles the connection to both LLM providers Google Gemini, and OpenAI, which creates the chatbot prompts, formats the conversation history, and processes the model's response. It also imports the rule-based parser only if the API call fails notifying the user as well, after notifying the user the parser result would be used in its place. Dashboard_tools.py controls approving certain actions, formatting, and approving of the filters, blocking certain actions. Chatbot_callbacks.py is the main connecting file that handles all Dash callback functions for the chatbot, the user question processing function, and also handles insight, parsing date ranges, and the visual chatbot bubble components. It is the only chatbot file connected via register with the Dash application when the website is in use. Layout.py was the only other module that was modified, which is a pre-existing file from the original dashboard setup, for the visual components. The chatbot drawer component was added which is the side panel where the chatbot, and conversation history is displayed, as well as the text box, send button, and the model dropdown box. Other components such as some additional dcc.Store components were added to align with the callback calls. Also, an AI assistant chatbot button was added to the left side of the visual layout to give the user access to the chatbot.

5.3 Conversational Management

All conversation data is stored explicitly using dcc.Store components which was added to layout.py. The chatbot-convo-store holds the conversation history, but caps the amount

of messages to twenty. The chatbot-history-by-email-store uses the local browser storage to save the users chatbot history which is connected to the users email at the time of application startup, the email log modal window populates, this is then sent to the LLM for use in new conversations. The chatbot-filter-store keeps the most recent filter updates from the chatbot, which is useful to the LLMs for explaining the current state, even if the user manually toggles the filters. The chatbot-pending-request stops the chatbot response and waits for the user to input additional information if needed such as date range.

5.4 Chatbot Security

Since the chatbot is used for transportation safety and law enforcement use, there have been several validation layers placed within the four modules to make sure that there is no main connection from the chatbot to the database to make sure there is no creation, deletion, or modification of any kind. First is the use of keyword blocking in the parser, where the user's question is compared with a list of blocked keywords, then there is schema approval for the LLMs output, where the JSON output is compared with the approved schema, and any unclear or unrecognized key values get removed before the output is approved and sent to the user. After that is the action labeled as allowlist where the action must match one of the three approved operations, or it is blocked, and the user is notified. Finally, the filter values are also checked against allowlist because it is sent to any Dash store. Together, these layers create a read-only environment where the chatbot can apply approved dashboard actions without directly modifying the transportation data.

Chapter 6

Results

6.1 Filter Accuracy Summary

The filter accuracy results are shown in Table 6.1, which shows the scores for each question across ten total runs for both models. OpenAI's overall average of 87% compared to Gemini's 77%, with both models performing better for clear filter requests. Questions updating filters of specific counties, or switching to the travel time reliability map, consistently performed highest between both models, with several questions scoring 100% for both models. The questions that scored the lowest were intent and filtering based questions because the chatbot reads from data already loaded on the screen rather than applying a direct filter because of the chatbot not being connected to the database. "Where do people speed the most in Alabama," scored the lowest overall at 30% for Gemini and 55% for OpenAI, and "which county has the most crashes", scored 60% for Gemini and 75% for OpenAI, showing that both models had the least amount of success with intent and filtering style questions. For questions that scored correctly, the filters were updated by the chatbot, similar to how a user would manually toggle the filters within the dashboard, which shows that the chatbot's output stayed consistent with the existing dashboard logic for all successful results.

Table 6.1. Filter Accuracy Summary		
Questions	**Gemini percentile**	**OpenAI percentile**
Q1. Show me jefferson county in january 2025?	0.75	0.8
Q2. Can you show me crash data for mobile county for 2024?	1	1
Q3. Show troop b, cmv only, jan 01 - 30 2024	0.95	1
Q4. Filter to calhoun county morning hours?	0.8	1
Q5. Monthly march 2024	0.8	0.75
Q6. Can you show me a scatter chart for calhoun county march 2024?	1	1
Q7. Where do people speed the most in alabama?	0.3	0.55
Q8. Which county has the most crashes?	0.6	0.75
Q9. Can you switch to the travel time reliability map 2023?	0.75	0.9
Q10. Show me i-65 N in january 2025?	0.7	0.9
average overall	**0.77%**	**0.87%**

Table 6.1: Filter Accuracy Summary

Table 6.1 The filter accuracy results showed that OpenAI performed better than Gemini, with overall scores of 87% and 77%. Both models performed better for direct filter requests and struggled more with intent-based questions.

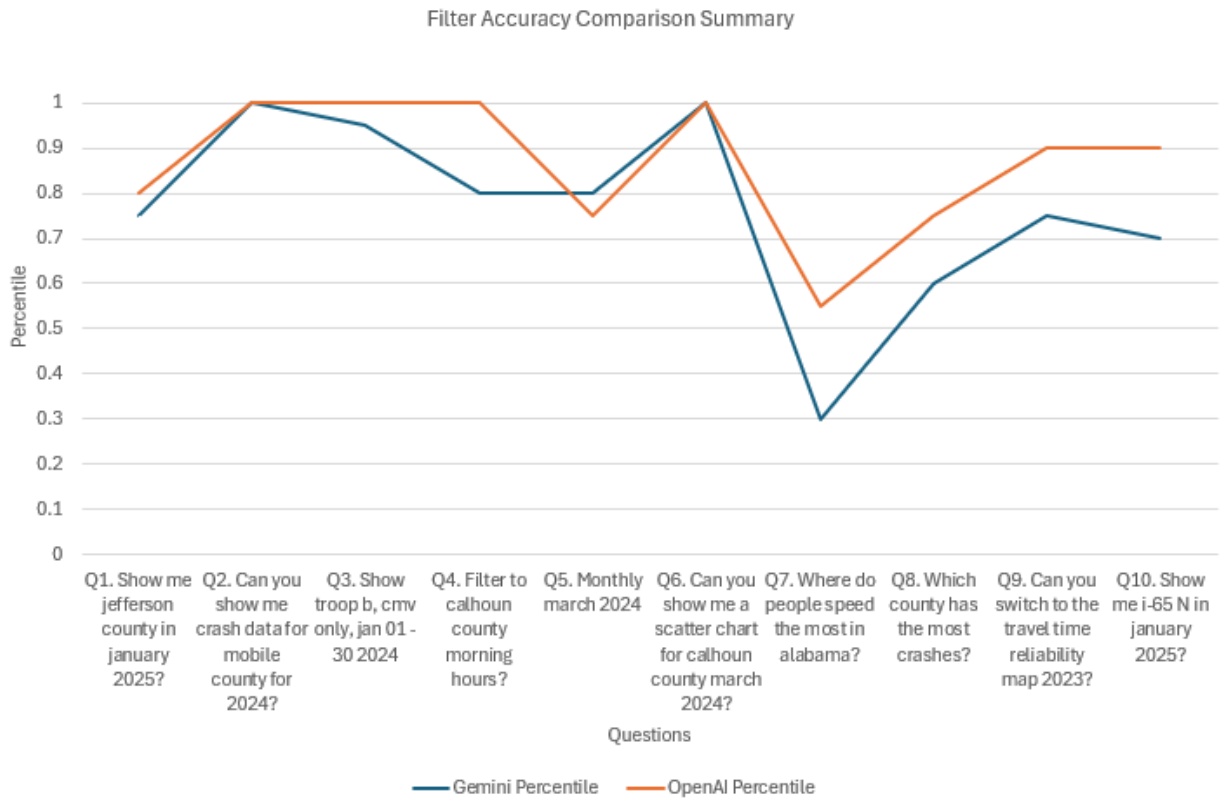

Figure 6.1: Filter Accuracy Comparison Summary

Figure 6.1 This figure shows the filter identification accuracy percentage scores for Google Gemini and OpenAI ChatGPT over all ten filter style questions, showing that OpenAI's model outperformed Google's model, and also showed the largest gap between the models.

6.2 Intent and Understanding Summary

The intent and understanding results are shown in Table 6.2, which tests how both models handle insight questions and how often each model hallucinated an output. OpenAI performed better as it averaged 87% accuracy with only a 13% hallucination percentage, while Gemini's model averaged 77% accuracy with a 23% hallucination percentage.

Both models scored 100% with a 0% hallucination percentage for clear and direct questions that could be answered from the SAFE-AL research paper, such as what the red dots mean, where the crash data comes from, and what LOTTR is. The weakest results for both models were on questions such as “how many segments are there” with scoring a 50% in accuracy as well as a 50% in hallucination percentage for both models. The questions “how is that calculated” had the largest hallucination variability between the two models, with Gemini hallucinating at 40% and OpenAI at 0%.

Table 6.2. Intent and Understanding Summary				
Questions	Gemini Accuracy	Gemini Hallucination	OpenAI Accuracy	OpenAI Hallucination
Q1. What am I looking at?	0.9	0.1	0.8	0.2
Q2. What do the colors mean?	0.9	0.1	1	0
Q3. What do the red dots mean?	1	0	1	0
Q4. Where does the crash data come from?	1	0	1	0
Q5. Is this in mph or kph?	0.9	0.1	1	0
Q6. How is that calculated?	0.6	0.4	1	0
Q7. What is lottr?	1	0	1	0
Q8. What am I looking at on this highway?	0.8	0.2	0.7	0.3
Q9. How many segments are there?	0.5	0.5	0.5	0.5
Q10. What does cmv mean?	1	0	1	0
average overall	**0.77%**	**0.23%**	**0.87%**	**0.13%**

Table 6.2: Intent and Understanding Summary

Table 6.2 Is a visual representation of intent and understanding summary. This table shows the accuracy and hallucination percentages for Google's model and OpenAI's model across ten runs per question.

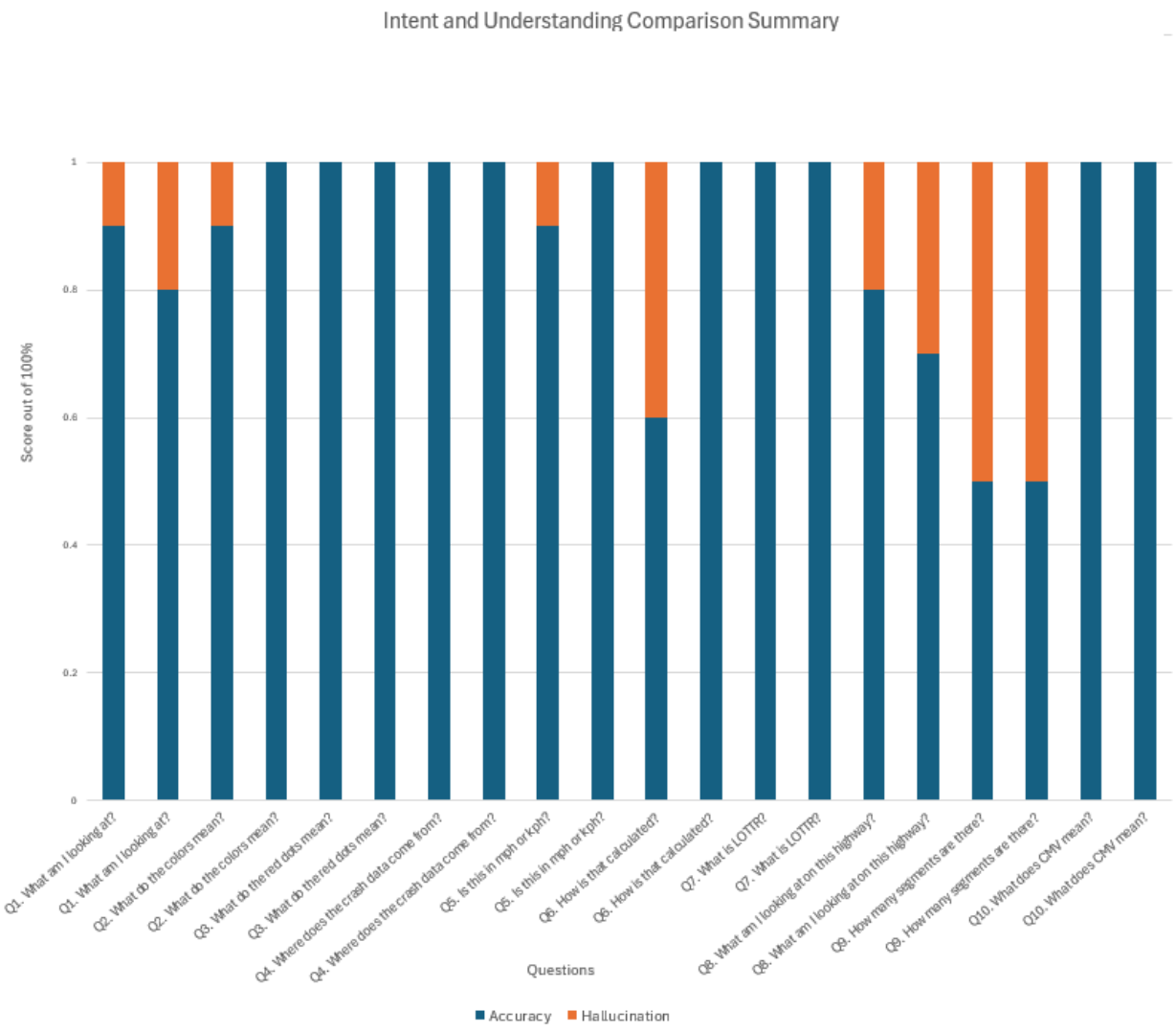

Figure 6.2: Intent and Understanding Comparison Summary

Figure 6.2 This figure shows the accuracy and hallucination scores of Google's model and OpenAI's model across all ten intent based questions, showing a side-by-side comparison of each model's performance per question.

6.3 Response Time Summary

The response time results are shown in Table 6.3, and the addition of a box plot, which shows the mean and standard deviation of 0.81 seconds, while Gemini averaged 3.81 seconds with a standard deviation of 1.49 seconds. The boxplot shows that OpenAI's response times were grouped in a closer range compared to Gemini which showed a wider spread grouping, and more variability between runs. The slowest questions for Gemini were filtering questions such as "show me Jefferson county in January 2025" and "show me i-65 N in January 2025," which averaged 5.18 seconds and 5.43 seconds. Overall, OpenAI was faster and more consistent across almost all questions, and most standard deviations being under 0.90 seconds.

Table 6.3. Average Response Time Summary						
	Questions	Gemini Mean (Sec)	Gemini STD Dev.	OpenAI Mean (Sec)	OpenAI STD Dev.	Manual (Sec)
Q1.	show me jefferson county in january 2025?	5.18	1.76	3.27	0.87	15.30
Q2.	what am I looking at?	3.13	0.52	2.21	0.30	N/A
Q3.	what do the colors mean?	2.88	0.63	2.13	0.53	N/A
Q4.	can you show me crash data for mobile county for 2024	4.41	1.45	2.55	0.33	23.72
Q5.	what do the red dots mean?	2.56	0.32	1.89	0.37	N/A
Q6.	where does the crash data come from?	2.42	0.45	1.93	0.39	N/A
Q7.	filter to calhoun county morning hours?	5.03	1.32	3.48	0.67	30.00
Q8.	monthly march 2024	5.03	1.32	3.56	0.60	8.42
Q9.	can you show me a scatter chart for calhoun county march 2024	5.91	1.37	2.54	0.43	40.77
Q10.	where do people speed the most in alabama?	3.94	1.02	2.87	0.47	N/A
Q11.	how is that calculated?	3.00	0.46	1.95	0.26	N/A
Q12.	which county has the most crashes?	3.95	1.83	2.53	0.43	N/A
Q13.	can you switch to the travel time reliability map 2023?	4.08	0.52	3.80	0.96	12.35
Q14.	what is lottr?	2.27	0.22	2.55	0.33	N/A
Q15.	show me i-65 N in january 2025	5.43	1.38	3.32	0.63	25.48
Q16.	what am I looking at on this highway?	3.06	0.34	2.15	0.35	N/A
Q17.	how many segments are there?	3.81	1.55	2.55	0.68	N/A
Q18.	show troop b, cmv only, jan 01 - 30 2024	4.69	1.12	3.64	0.72	30.35
Q19.	what does cmv mean?	2.72	0.54	2.50	0.74	N/A
Q20.	is this in mph or kph?	2.73	0.46	1.91	0.34	N/A
	average overall	**3.81**	**1.49**	**2.67**	**0.81**	**23.30**

Table 6.3: Response Time Summary

Table 6.3 This table shows the mean response time in seconds and standard deviation for Google Gemini and OpenAI ChatGPT across ten runs per question, along with the manual response times for applicable dashboard filter questions.

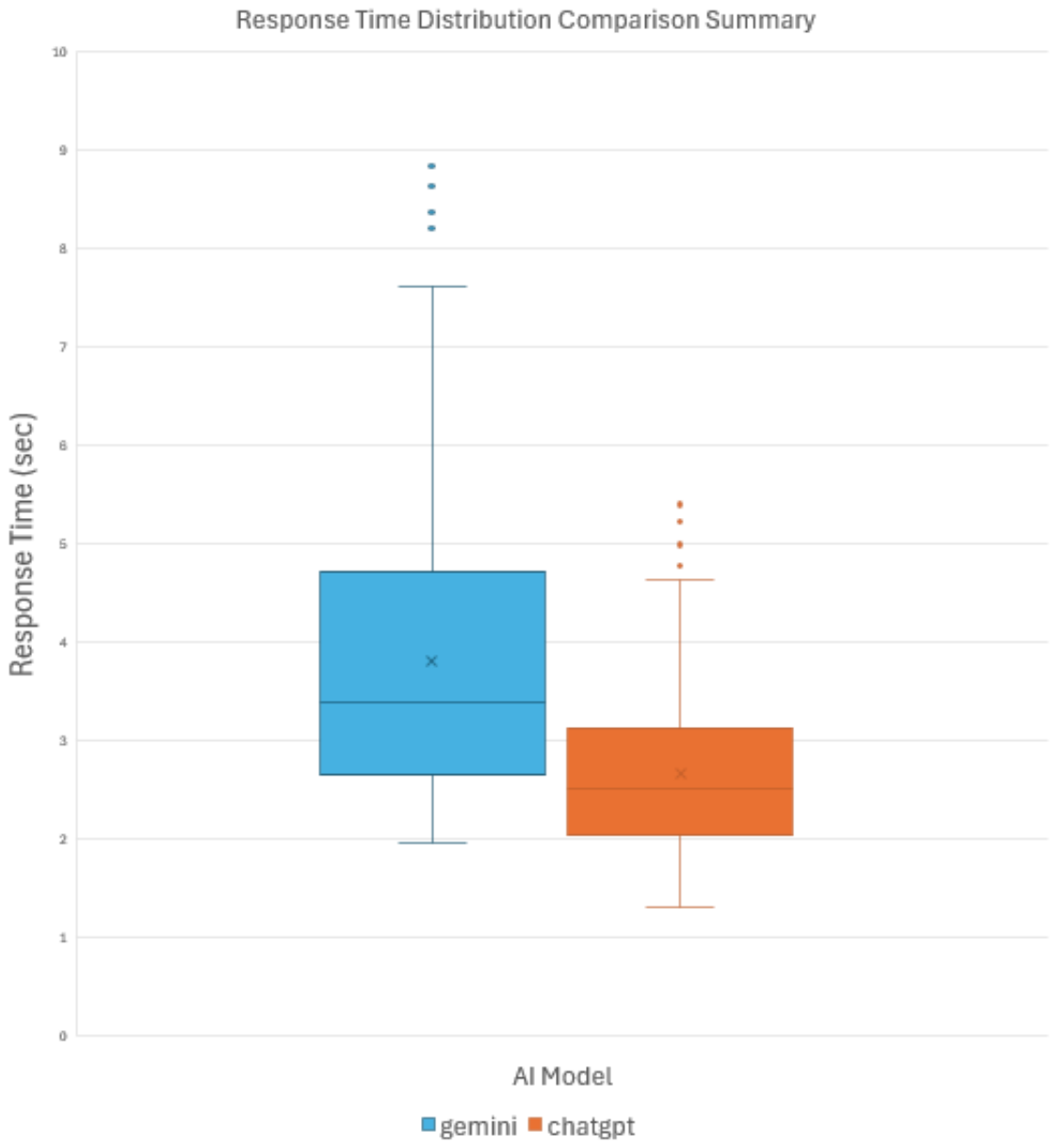

Figure 6.3: Response Time Box Plot

Figure 6.3 Is a visualized representation of distribution for Google Gemini and OpenAI ChatGPT. The figure shows the variability of response time in seconds across all twenty questions, across ten runs per question.

6.4 Manual Response Time Summary

The manual response time results were compared against the average chatbot response times to determine the percentage of time saved for each question. Overall, both models minimized the amount of time needed to apply filters within the dashboard, with OpenAI showing a decrease in time across most questions.

Table 6.4. Manual Response Time Summary

	Questions	Gemini vs Manual (%)	OpenAI vs Manual (%)
Q1.	Show me jefferson county in january 2025?	66.15	78.61
Q4.	Can you show me crash data for mobile county for 2024	81.40	89.23
Q7.	Filter to calhoun county morning hours?	83.24	88.40
Q8.	Monthly march 2024	40.27	57.77
Q9.	Can you show me a scatter chart for calhoun county march 2024	85.51	93.78
Q13.	Can you switch to the travel time reliability map 2023?	66.95	69.25
Q15.	Show me i-65 N in january 2025	78.68	86.97
Q18.	Show troop b, cmv only, jan 01 - 30 2024	84.53	88.02

Table 6.4: Manual Response Time Table

Table 6.4 This table shows the percentage of time saved by Google Gemini and OpenAI ChatGPT compared to manually applying the same dashboard filters, showing the improvement in user efficiency across the applicable filter based questions.

Figure 6.4 and Equation 6.1 show the percentage of time saved by the chatbot compared to a user manually toggling the same filters through the dashboard. Because

each question had a different manual completion time depending on the number of filters needed, Equation 6.1 was used to maintain consistency when comparing all questions. Both models showed a reduction in time compared to manually interacting with the dashboard, with OpenAI saving a higher percentage of time across most questions, which is consistent with its faster response times shown in Table 6.3.

$$\text{Time Saved (\%)} = \left(1 - \frac{T_{\text{LLM}}}{T_{\text{Manual}}}\right) \times 100 \tag{6.1}$$

where T_{LLM} is the chatbot's average response time (seconds), and T_{Manual} is the manually recorded response time (seconds) for the same question.

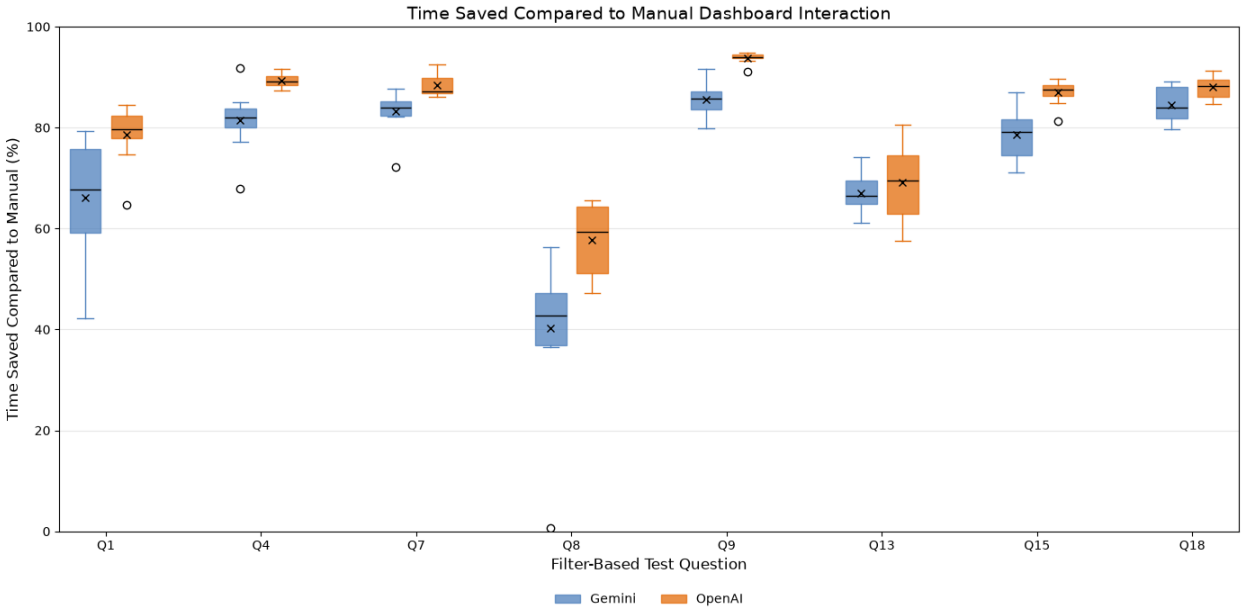

Figure 6.4: Time Saved Box Plot

Figure 6.4 This visualized representation of Percentage of time saved compared to manual dashboard interaction. This figure compares Google Gemini and OpenAI ChatGPT across the filter-based test questions using the percentage of time saved relative to manually applying the dashboard filters.

6.5 Cost Effectiveness

Cost Effectiveness was included as an additional evaluation metric. The total cost for each model was calculated using the total number of requests during testing and the pricing information provided by each model provider [28, 29]. The cost effectiveness of both AI models have been reviewed after testing for comparison. OpenAI GPT-4.1-mini had a total cost of $0.31, averaging $0.10 per day. Google Gemini 2.5 Flash had an estimated total cost of $0.70 over the same time period, averaging $0.23 per day, with each model having a total of 152 requests. The difference between both models was $0.39 in total, making Google Gemini 2.5 Flash approximately 2.3 times more expensive than OpenAI GPT-4.1-mini when testing the same set of questions for each model. OpenAI's model has shown to outperform Google's model in testing as well as cost effectiveness when integrating a chatbot into the SAFE-AL dashboard.

6.6 Overall Results

After testing both models across three different tests with a total of ten runs each, both models were able to handle the majority of the questions. OpenAI scored higher than Gemini in filter accuracy, intent and understanding and response time, while also having the lowest results for hallucinations. Both models equally struggled with the filter and intent questions. Cost Effectiveness was also included as part of the evaluation to compare the overall cost of each model within the SAFE-AL dashboard. Overall, both models correctly handled most of the filter questions included in the testing, with OpenAI achieving a higher overall filter accuracy than Gemini.

Chapter 7

Discussion

7.1 Summary of Results

The testing results across all three tables showed that both models were able to handle the majority of the questions during testing. For accuracy in Table 6.1, questions that involved strictly filtering questions such as showing crash data for a specific county or switching to the travel time reliability map consistently produced the highest results from both AI models. These types of questions are simple and clear for filtering on the map within the dashboard, which makes it easier for both models and the rule-based parser to pull the correct values from the question and apply them to the filters. Questions such as “can you show me crash data for mobile county for 2024” did significantly well compared to others as it scored 100% for both models, showing that the chatbot is very efficient and reliable in handling clear and direct filtering.

For the intent and understanding testing in Table 6.2, both models performed best with questions that could be answered directly from the SAFE-AL research paper which loads into the LLM’s context when the application launches. Questions such as “what do the red dots mean,” “where does the crash data come from,” and “what is LOTTR” all performed well as they each scored 100% on accuracy with no hallucinations from either of the models. The results show that the models were able to correctly answer most of the tested questions using the SAFE-AL research paper as additional context, and using verified information and reducing the chances of outputting incorrect responses.

7.2 Gemini vs ChatGPT Comparison

Across all three tables, OpenAI GPT-4.1-mini consistently outperformed Google Gemini 2.5 Flash. For filter accuracy in Table 6.1, OpenAI achieved an overall average of 87% compared to Gemini's 77%. For intent and understanding in Table 6.2, OpenAI had scored 87% accuracy with a 13% hallucination rate, while Gemini scored a 77% accuracy with a 23% hallucination rate. The most noticeable difference between the two models was for the question "how is that calculated" with Gemini hallucinating 40% compared to OpenAI's 0%, showing that Gemini is more likely to output incorrect information or a vague response.

For response time in Table 6.3, Open AI was not only faster but also more consistent than Gemini. OpenAI averaged 2.67 seconds per question and with a standard deviation of 0.81 seconds while Gemini averaged 3.81 seconds with a standard deviation of 1.49 seconds. The box plot shows that OpenAI's response times were more consistent with grouping in the lower range, while Gemini had a wider margin with more variability between runs. The questions with the highest variability for Gemini were "show me Jefferson County in January 2025" and "show me i-65 N in January 2025" both of which have multiple actions for filtering which shows that with questions such as these Gemini takes longer to process and is less consistent when the question is more complex. Based on the results from all three tables, OpenAI GPT-4.1-mini performed better than Google Gemini 2.5 Flash in many areas.

7.3 Limitations Specific to the System

While both models performed well on filtering and simple questions, the testing also highlighted areas where the chatbot was slower, and less consistent. The insight question "where do people speed the most in Alabama" was the lowest for both models where Gemini had 30%, and OpenAI doing slightly better at 55%, neither model scored high

enough to show consistency or reliability when it came to certain questions because the insight function reads what is currently on the screen and not the entire database. This is a limitation of the system, where the conversational questions are restricted to the data that is already loaded into the dashboard at the time the question is asked. As a result, the chatbot's outputs are based on the current dashboard state.

The question "how many segments are there" also did poorly as both models scored a 50% in accuracy and a 50% in hallucination. The number of segments visible on the map changes depending on the filters, and the chatbot does not have a way to reliably count and output that number consistently within the dashboard. This shows that the chatbot performs better with clear and consistent filtering questions, rather than interpretation of current state data visualized on the interactive map.

Chapter 8

Future Work

With the chatbot being implemented into the Safe-AL dashboard which adds a conversational interface, and there are several areas within the chatbot that could be improved and added on to enhance the users experience. These additions are based on the current status of the chatbots and systems limitations, which can be added on to the existing dashboard.

8.1 User Feedback and System Improvement

One improvement which the chatbot could benefit from is a user feedback feature, added into the chatbot side panel. After the end of a conversation, a prompt with buttons could populate, giving the user the option to select a thumbs up button or thumbs down button to show whether the chatbot is responding accordingly or could use further improvement. For filter-based replies if a bad response was created and could be recorded as well as what filters were triggered, and adding the response to a dataset which can be used to identify patterns in how the parser fails for future system updates for the LLMs. Since the dashboard already tracks user sessions by email address, the feedback information could be stored with that information, which could be used for further updates.

8.2 PDF Report Creation

Another improvement for the chatbot would be to have access to a summary of what is currently displayed on the dashboard in the form of a PDF report. Currently the chatbot can explain what the user is looking at, but currently there is no way to export this information for further use. Giving the user the ability to create formatted PDF that includes the current filters, an explanation of the summary, and other on-screen metrics from the map or charts. This would be very useful for data scientists and law enforcement agencies. Because the chatbot is already connected to the dashboards current state, and can produce natural language summaries, further implementation would be connecting the output to a report generation library, creating a button to toggle within the chatbot panel and adding an additional callback.

8.3 Side-by-Side Comparison Map

Currently, the dashboard supports only a single interactive map and chart view at a time, meaning that users have to manually toggle between filters to compare different maps and charts based on different time ranges, vehicle types, or counties. A side-by-side comparison function would let the chatbot display to display two interactive maps or charts side by side, which can be used to compare speeding data based on the toggled filters. This feature could also be used to support before and after studies to evaluate the effectiveness of speeding countermeasures.Integrating this feature into the pre-existing dashboard for manual filter updates, and can be used with the chatbot for a more beginner-friendly use.

Chapter 9

Conclusion

After completing the design, implementation, and testing phases of the SAFE-AL chatbot within the pre-existing dashboard, the results showed that the chatbot could successfully interpret user requests and update the dashboard. The chatbot was built to be a user-friendly natural language interface for interacting with the transportation data and filtering logic within the dashboard, without requiring users to have prior knowledge of how the dashboard works. By combining a rule-based parser with a large language model using either Google Gemini 2.5 Flash or OpenAI GPT-4.1-mini, the chatbot has been tested and can handle a variety of different questions, while staying consistent with the pre-existing dashboard outputs.

One of the most important features of the chatbot is the read-only design, which ensures security for the database, because within the chatbot the user cannot create, modify, or delete any part of the database. This design approach could be useful for other data tools in which safety, security, consistency, and usability are important requirements. In transportation research, being able to trust the outputs given by the chatbot, and comparing the outputs against the manually toggled dashboard results is an important part of making a reliable tool, especially in a live application.

This research study also shows that by building the chatbot directly into the pre-existing dashboard instead of modifying or creating a completely separate system. Shows the ease of implementation. By reusing the same callback functions, component stores, and filtering logic from the pre-existing dashboard, the chatbot was added without changing the

overall behavior, without needing to use extra tools. This design could be used in other transportation dashboards that are only manually interactive, and are not so user-friendly.

To conclude, this research study shows that by implementing an AI assistant chatbot within a transportation dashboard, it helps by making complex transportation datasets, filtering logic more user-friendly, and can be used in more practical settings. This tool also allows the user to explore and understand the data in an easier format, through the use of natural language rather than manual interaction.

References

[1] National Center for Statistics and Analysis, *Overview of motor vehicle traffic crashes in 2024*, tech. rep. DOT HS 813 791 (National Highway Traffic Safety Administration, Apr. 2026).

[2] K. Tennyson, A. Kivette, E. O. A. Tufuor, A. B. Cottam, and L. R. Rilett, *A low-cost methodology for geospatial exploration of roadway speeds, travel time reliability, and safety performance metrics* (Auburn University Transportation Research Institute, 2026).

[3] C. Bachechi, L. Po, and F. Rollo, "Big data analytics and visualization in traffic monitoring," Big Data Research **27**, 100292 (2022).

[4] National Performance Management Research Data Set Analytics, *Npmrds data quality report 2024 q1*, (2024) https://npmrds.ritis.org/static/help/docs/NPMRDS_Data_Quality_Report_2024_Q1.pdf (visited on 01/05/2025).

[5] L. S. Iyer, "AI-enabled applications towards intelligent transportation," Transportation Engineering **5**, 100083 (2021).

[6] Å. Jevinger, C. Zhao, J. A. Persson, and P. Davidsson, "Artificial intelligence for improving public transport: a mapping study," Public Transport **16**, 99–158 (2024).

[7] S. Saki and M. Soori, "Artificial intelligence, machine learning and deep learning in advanced transportation systems: a review," Multimodal Transportation, 100242 (2025).

[8] R. R. Saxena, *Artificial intelligence in traffic systems*, (2024)

[9] M. F. H. Fahim, M. M. Usman, M. Al Mahedi Hassan, T. H. Sardar, K. P. Bindu Madavi, and N. Venkatachalam, “Revolutionizing road transportation: the role of artificial intelligence in smart and efficient systems,” in *Applications of computational learning and IoT in smart road transportation system* (Springer Nature Switzerland, Cham, 2025), pp. 87–107.

[10] M. J. Tantillo, K. Smith, C. Packard, T. Lomax, and S. Dhuri, *Transportation management center performance dashboards* (2021).

[11] J. Zerafa, R. M. Islam, A. Kabir, and G. Xu, “ExTraVis: exploration of traffic incidents using visual interactive systems,” in 25th international conference on information visualisation (2021), pp. 48–53.

[12] J. Jung, T. Oh, I. Kim, and S. Park, “Open-sourced real-time visualization platform for traffic simulation,” Procedia Computer Science **220**, 243–250 (2023).

[13] Regional Integrated Transportation Information System, *RITIS tool catalog*, https://www.ritis.org/tools (visited on 08/04/2026).

[14] H. Weld, X. Huang, S. Long, J. Poon, and S. C. Han, “A survey of joint intent detection and slot filling models in natural language understanding,” ACM Computing Surveys **55**, 1–38 (2023).

[15] S. Dharmireddi, H. M. Mahdi, M. Rajendran, I. W. Suryasa, and A. Soy, “Artificial intelligence-driven natural language processing for futuristic language processing,” in 2025 international conference on computational innovations and engineering sustainability (ICCIES) (2025), pp. 1–6.

[16] A. Narechania, A. Srinivasan, and J. Stasko, “NL4DV: a toolkit for generating analytic specifications for data visualization from natural language queries,” IEEE Transactions on Visualization and Computer Graphics **27**, 369–379 (2020).

[17] R. Mitra, A. Narechania, A. Endert, and J. Stasko, “Facilitating conversational interaction in natural language interfaces for visualization,” in 2022 ieee visualization and visual analytics (VIS) (2022), pp. 6–10.

[18] Y. Chen, R. Li, A. Mac, T. Xie, T. Yu, and E. Wu, *NL2Interface: interactive visualization interface generation from natural language queries*, (2022)

[19] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, and D. Radev, “Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task,” in Proceedings of the 2018 conference on empirical methods in natural language processing (2018), pp. 3911–3921.

[20] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, “RAT-SQL: relation-aware schema encoding and linking for text-to-SQL parsers,” in Proceedings of the 58th annual meeting of the association for computational linguistics (2020), pp. 7567–7578.

[21] T. Scholak, N. Schucher, and D. Bahdanau, “PICARD: parsing incrementally for constrained auto-regressive decoding from language models,” in Proceedings of the 2021 conference on empirical methods in natural language processing (2021), pp. 9895–9901.

[22] J. Sen, C. Lei, A. Quamar, F. Özcan, V. Efthymiou, A. Dalmia, and K. Sankaranarayanan, “ATHENA++: natural language querying for complex nested SQL queries,” Proceedings of the VLDB Endowment **13**, 2747–2759 (2020).

[23] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, *Text-to-SQL empowered by large language models: a benchmark evaluation*, (2023)

[24] A. Narechania, A. Fourney, B. Lee, and G. Ramos, “DIY: assessing the correctness of natural language to SQL systems,” in Proceedings of the 26th international conference on intelligent user interfaces (2021), pp. 597–607.

[25] R. Zhong, T. Yu, and D. Klein, “Semantic evaluation for text-to-SQL with distilled test suites,” in Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP) (2020), pp. 396–411.

[26] U.S. Department of Transportation, Federal Highway Administration, *Critical analysis reporting environment (CARE)*, (2022) https://highways.dot.gov/safety/data-analysis-tools/rsdp/rsdp-tools/critical-analysis-reporting-environment-care (visited on 08/05/2026).

[27] E. Tufuor, A. Cottam, and L. R. Rilett, “A statewide highway system network analysis: impact of probe vehicle composition,” Transportation Research Record: Journal of the Transportation Research Board **2680**, 523–541 (2025).

[28] Google, *Gemini 2.5 flash*, (2025) https://ai.google.dev/gemini-api/docs/models/gemini-2.5-flash (visited on 08/05/2026).

[29] OpenAI, *Gpt-4.1-mini*, (2025) https://developers.openai.com/api/docs/models/gpt-4.1-mini (visited on 08/05/2026).