An Agentic System for LLM-Driven Public Transportation Analytics: A Practical Application and Case Study in Salvador-Brazil

Lucas T. Borges
Federal University of Bahia
Salvador, Brazil
lucastb@ufba.br

Fei T. Liu
Artificial General Intelligence Pty Ltd.
New South Wales, Australia
feitony.liu@volantech.io

Tatiane Rios
Federal University of Bahia
Salvador, Brazil
tatiane.nogueira@ufba.br

Marcos V. Ferreira
Federal University of Bahia
Salvador, Brazil
marcosvsf@ufba.br

Clovis Carmo
Federal University of Bahia
Salvador, Brazil
cloviscarmo@ufba.br

Danilo B. Coimbra
Federal University of Bahia
Salvador, Brazil
coimbra.danilo@ufba.br

Jorge Nery
Federal University of Bahia
Salvador, Brazil
jorgenery@ufba.br

Matheus Souza
Integra - Association of Transport Companies of Salvador
Salvador, Brazil
matheus.souza@gevan.com.br

Noe O. Garcia
Arcadis
Amsterdam, Netherlands
ecostat.nog@gmail.com

Albert Bifet
University of Waikato
Hamilton, New Zealand
abifet@waikato.ac.nz

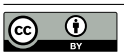
Ricardo Rios
Federal University of Bahia
Salvador, Brazil
rios.fsa@gmail.com

Abstract

Public transportation agencies gather vast and heterogeneous datasets, yet their decisions still depend largely on manual queries and fragmented analyses. To address this gap, we introduce SUNTInsight, an agentic system that combines large language models, machine learning, and visual analytics to strengthen human decision-making. Through a seamless workflow, free-form prompts are automatically converted into SQL queries, relevant data are retrieved and processed, and an LLM, supported by visual interfaces, interprets the results to produce actionable findings. Applied to a case study in Salvador-Brazil, SUNTInsight revealed spatial heterogeneity, identified high-demand segments, and recommended targeted operational strategies, such as short turns and headway control, instead of broad fleet expansions. The implementation follows the principle of least privilege and runs generated code in isolated execution, mitigating risk while keeping humans in control.

CCS Concepts

• **Computing methodologies** → **Information extraction**; • **Software and its engineering** → **Software as a service orchestration system**; • **Applied computing** → **Transportation**.



AGENT '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2399-5/26/04
https://doi.org/10.1145/3786167.3788417

Keywords

Agentic AI, Large language models, urban mobility

ACM Reference Format:
Lucas T. Borges, Fei T. Liu, Tatiane Rios, Marcos V. Ferreira, Clovis Carmo, Danilo B. Coimbra, Jorge Nery, Matheus Souza, Noe O. Garcia, Albert Bifet, and Ricardo Rios. 2026. An Agentic System for LLM-Driven Public Transportation Analytics: A Practical Application and Case Study in Salvador-Brazil. In *International Workshop on Agentic Engineering (AGENT '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3786167.3788417

1 Introduction

Advances in Artificial Intelligence (AI) are reshaping society across domains, including transportation systems [5, 18], which have adopted, for example, Computer Vision for image understanding, Large Language Models (LLM) for natural language interaction, Machine Learning (ML) for knowledge discovery, and Visualization for human understanding.

Beyond individual capabilities, combining AI systems creates a more advanced environment that can execute multistep workflows, plan actions, and support decision-making. In the literature, this paradigm is referred to as Agentic AI [16]. It leverages complementary AI models to integrate information from multiple sources, monitor the operating context, and learn from historical data, enabling systems to perceive context, reason about goals, plan and coordinate actions, and ultimately support decision makers in a human in the loop process.

In public transportation, companies collect large, heterogeneous datasets, yet decision making still relies on static dashboards and

complex manual queries. This creates a significant gap between raw data and actionable operational insights. Agentic AI systems, which combine the reasoning abilities of large language models with external tools and data, offer a promising way to bridge this gap and support decision makers in a human in the loop process.

In this work, we present SUNTInsight, an agentic system that delivers LLM-driven visual analytics for public transportation management. The system offers a natural language interface that lets managers query complex datasets and generate visualizations without specialized knowledge of AI, databases, or programming.

We demonstrate these functionalities through a real world case study in agentic engineering, analyzing the public transportation network of Salvador-Brazil. The primary engineering contributions are:

- The software architecture for a modular agentic system that integrates a Python backend, a database, an LLM server, and a web-based UI (User Interface) to provide managers with an interactive visual analytics workspace to support evidence-based decision-making.
- The implementation of a multi-layered security model to mitigate the risks of executing LLM-generated code, applying the Principle of Least Privilege and a secure execution environment.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work. Section 3 details the system's architecture, design and implementation. Section 4 discusses security and monitoring of the agent performance. Section 5 presents an evaluation of the system's core functionalities as a case study. Finally, Section 6 concludes by summarizing contributions and outlining future work.

2 Related Work

Recent advancements in Large Language Models (LLMs) and foundation models have opened new pathways in transportation research, particularly in advancing intelligent and data-driven mobility systems [18]. Nonetheless, the literature reveals a conceptual divide between AI agentic approaches, in which LLMs operate as embedded functional components, and Agentic AI systems, where autonomy, reasoning, and multi-process coordination emerge as intrinsic system capabilities [1, 16].

Early AI agents lacked self-learning, generative reasoning, and adaptability to unstructured or evolving environments, limiting their scalability in real-world applications. The Agentic AI paradigm, in contrast, builds upon deep learning, reinforcement learning, and foundation models to endow systems with contextual awareness, continuous learning, and emergent autonomy [1]. Recent studies have partially bridged this gap through hybrid designs, for instance, LLMTraveler [19] integrates an LLM with memory and behavioral modeling, and STLLM-DF [17] couples diffusion-based recovery with spatial–temporal reasoning to enhance multimodal predictions, but both remain within the AI agentic scope, as control and goal-setting remain externally defined.

Other frameworks, such as Traffic-IT [6, 7] and LLeCaT [20], further expand multimodal perception and causal reasoning, while LLM-based signal controllers [9] introduce adaptive real-time decision-making. Yet these systems still operate under bounded objectives. A closer step toward autonomy appears in the Traffic Research Agent (TR-Agent) [5], which iteratively refines models through reasoning, evaluation, and self-generated code, approaching the continuous, self-improving capabilities envisioned for Agentic AI in dynamic urban contexts.

In contrast to these works, SUNTInsight represents a complete Agentic AI system explicitly designed for real-world urban mobility management. Rather than embedding LLMs as isolated components, it integrates reasoning, interaction, and visualization into a single, cohesive workflow. By combining natural-language interaction with predictive and visual analytics, SUNTInsight bridges the gap between operational data and strategic decision-making, illustrating how Agentic AI can strengthen the development of intelligent, inclusive, and sustainable urban mobility [19, 21].

3 System Architecture

SUNTInsight is built on a client-server architecture, as shown in Figure 1. The user interface is a lightweight client that runs in the web browser and handles prompt entry and visualization. The server hosts the core services for data access, LLM orchestration, model inference, and visualization pipelines. This separation keeps the client simple and concentrates computation and data governance on the server.

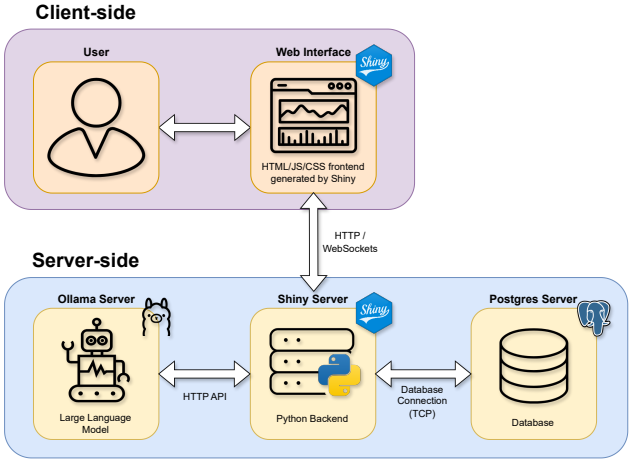


Figure 1: The high-level application architecture diagram of the SUNTInsight application, illustrating the separation between client-side and server-side components.

On the client side, the user interacts with a dynamic web interface. This front-end is generated with the Shiny for Python framework and delivered to the browser as standard HTML, CSS, and JavaScript. The interface includes all interactive elements, such as data visualizations and the chat sidebar used to enter prompts. All charts are rendered as Plotly figures.

The client communicates with the Server-side via a persistent HTTP/WebSocket connection. The central orchestrator on the server is the Shiny Server, which implements the business logic, receives events from the user interface, manages application state, and coordinates tasks across the backend services.

To fulfill user requests, prompted through the chat sidebar, the Shiny server relies on two key dependencies. For data retrieval and manipulation, it establishes a TCP connection with a Postgres Server, where it executes SQL queries against the transit data tables. For natural language understanding and generation, it communicates with an Ollama Server via an HTTP API. This server hosts the LLM that powers the agentic capabilities of SUNTInsight.

Following the system architecture described in Section 3, this section details the core design decisions and implementation of the SUNTInsight system. We first address the foundational data modeling and the model selection process, justifying the design trade-offs involved in selecting a reliable Large Language Model (LLM). Finally, we present the agentic interaction flows, explaining the distinct pipelines for the dashboard and how the agent utilizes tools versus structured outputs.

3.1 Data and Model Selection

A foundational design step was the normalization of the raw SUNT dataset [3] into a structured Postgres relational database. The schema is organized into two logical groups: a core data model (containing the `zone`, `stop`, `route`, `trip`, and `origin_destination` tables) and an application model (containing `chat_session`, `chat_message`, and `od_counter`) to log user interactions and application-specific data. A key normalization decision, in accordance with database normalization principles [2], was to extract trip-related attributes from the main `origin_destination` table into a separate `trip` entity. This resolved significant data redundancy and allowed the `trip` entity to be uniquely identified by a composite primary key of (`trip_date`, `trip_id`), as the `trip_id` alone is not unique across different days.

The selection of a suitable open-weight LLM was an empirical process driven by the need for reliable tool-calling and structured output capabilities. Initial experiments were conducted with smaller models renowned for tool use, including `Lllama-3-Groq-8B-Tool-Use` [4] and `Mistral NeMo` (12B) [8]. While these models could handle simple conversational tasks, they often failed to execute the required tool calls consistently, instead "hallucinating" a response as if the tool had been called. Other models in the 14B–24B parameter range, such as `Qwen2.5-Coder-14B` [12] and `mistral-small` (24B) [11], were also tested but exhibited similar inconsistencies.

A significant improvement in reliability was observed when testing larger, reasoning-focused models with over 30 billion parameters. Both `deepseek-r1:32b` [10] and `qwen3:32b` [13] demonstrated consistent and accurate tool-calling for the tasks required by the dashboard. However, the more lightweight `gpt-oss:20b` [14] model was adopted, as it provided comparable performance to the 32B models for basic tool use, representing a good compromise between reliability and computational cost.

3.2 Interaction and Visualization Layer

The SUNTInsight interface unifies these capabilities in an interactive dashboard that integrates rich visualizations with a conversational AI agent. The dashboard is the primary entry point for data exploration and is composed of diverse visualizations, as shown in Figure 2.

In this figure, panel (b) includes a text box where users type commands for the LLM to select and plot the required information. Initially, the panel shows example prompts and explains how to reset the context to avoid residual influences from previous interactions. In subsequent use, it also keeps a history and shows how prompts are transformed into SQL queries.

The dropdown in panel (a) allows the user to select the primary metric to be analyzed (e.g., loading, n-boarding or n-alighting), which reactively updates the majority of the dashboard's visualizations that are shown in Panel (d). Panel (c) presents a brief summary showing the number of records retrieved, the number of stops or stations analyzed, and the overall mean occupancy.

More advanced interactions are handled by an AI agent via the chat interface, following the request fulfillment pipeline illustrated in Figure 3. The agent can process several types of requests:

(1) For general inquiries that do not require use of agentic tools, the agent provides a direct, text-based answer based on its system prompt, which includes information about the core data structure, such as the tables and columns it has access to, as well as the dataset context.
(2) When the user asks to filter or sort the data (e.g. "Show data from March 4th onwards"), the agent invokes the *update_dashboard* tool. Then, LLM generates a SQL query that filters and/or sorts the *origin_destination* table accordingly. The query is used alongside a pre-made template that joins the resulting table with data from other tables that is relevant for multiple widgets to create a temporary cached table named *base_table*. What follows is the execution of the premade SQL queries for each widget, which are run against the *base_table*. Once the data is retrieved by the application, the executed query is displayed in the chat and Shiny reactively updates the widgets on the dashboard based on the results and the primary metric selected in the dropdown menu. The widgets are rendered in the frontend using the Plotly library.
(3) For questions that require specific data analysis (e.g. "What is the standard deviation for the average loading across all stops in the 'Cabula' zone?"), the agent uses the *query_db* tool. In this flow, the LLM generates a complete SQL query that is then executed in the database. After the required data has been retrieved, the backend sends the result back to the LLM for analysis, allowing it to formulate a final, natural-language answer to the user's question based on the retrieved data. The response is then displayed on the chat, along with the executed query and its results.
(4) The user can issue a "Reset" request, which triggers the *reset_chat* tool. This tool works similarly to the *update_dashboard* tool, but instead of generating the *base_table* using a filtered/sorted version of the *origin_destination* table resulting from an LLM generated query, it runs the *base_table* premade SQL template against the full original table.

4 Safety and Operations

An agentic system that executes LLM-generated code in a real-world setting requires rigorous operational monitoring and safety protocols. This section details the two pillars of this implementation: first, the AgentOps framework used to monitor and evaluate agent

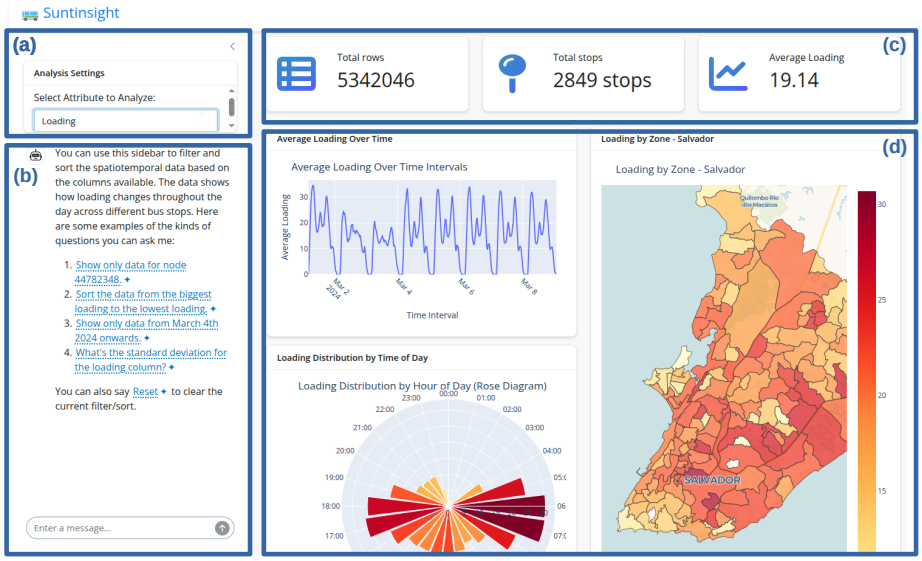


Figure 2: SUNTInsight interface panels: (a) filters, (b) prompt input, (c) summary, and (d) visualizations.

performance, and second, the specific cybersecurity measures taken to mitigate the risks of executing generated code.

4.1 AgentOps and Monitoring

A key component of agentic engineering is "AgentOps", which involves the continuous monitoring, evaluation, and oversight of the agent's performance in deployment [22]. To this end, SUNTInsight implements a comprehensive logging system to evaluate user demands and overall agent performance. The system logs every message exchanged in the `chat_session` and `chat_message` tables. The `chat_session` table tracks which dashboard is being used (via the `session_type` column), allowing for the complete reconstruction of each chat.

In addition to the user's prompt and the model's final reply, the system captures a comprehensive set of metadata for each interaction. These metrics include the SQL query and/or Python code executed, the specific tool called/DynamicResponse type generated, an error flag, the LLM response time, the total end-to-end request time, and the input/output token counts. This token-level data is particularly valuable for estimating the potential operational costs of migrating the system to a commercial cloud-hosted LLM service.

Furthermore, the `od_counter` table tracks the number of times each record in `origin_destination` was included in a filter query from the dashboard. This provides a direct measure of user interest in specific routes, stops, or time periods. This information, along with the message logs, allows the system to evolve based on user demands. For instance, prompts that consistently cause failures can be used to improve the system's prompt engineering.

4.2 Cybersecurity and Safety

This project presents a significant challenge due to the inherent risk associated with executing LLM-generated code in response to arbitrary user prompts. This presents a dual threat: the LLM may generate code that is unintentionally flawed, and a malicious user may attempt to prompt the system to generate and execute harmful code. To mitigate these risks, the application was designed following the Principle of Least Privilege [15], which dictates that each component must have only the minimum permissions necessary to perform its respective tasks.

To ensure data are not tampered with by LLM-generated SQL, a dedicated, non-privileged database role (`app_user`) was created. This role has read-only (`SELECT`) access to the core data tables (`origin_destination`, `stop`, etc.). Write permissions are strictly limited: the application has `INSERT` privileges on `chat_session` and `chat_message` to record conversational history, and a column-specific `UPDATE` privilege only on the `hit_count` column of the `od_counter` table. Moreover, to ensure user privacy, broad `SELECT` permissions on the chat tables are withheld; the application is only granted `SELECT` on the `session_id` column to associate new messages with the current session without exposing other conversations.

5 Evaluation: A Real-World Case Study

To evaluate SUNTInsight's practical utility, we applied the agentic system to the SUNT dataset [3], a real-world collection of public transportation data from Salvador, Brazil. SUNT dataset was collected in Salvador from March 2024 to March 2025 at subminute resolution, covering about 700,000 passengers and approximately 2,000 vehicles operating across nearly 400 lines, and connecting almost 3,000 stops and stations.

The evaluation setup examines the end-to-end workflow of SUNTInsight, encompassing the entire process from interpreting user commands to retrieving and analyzing information for decision-making. In this context, decisions combine evidence derived from historical data with expectations about future system dynamics.

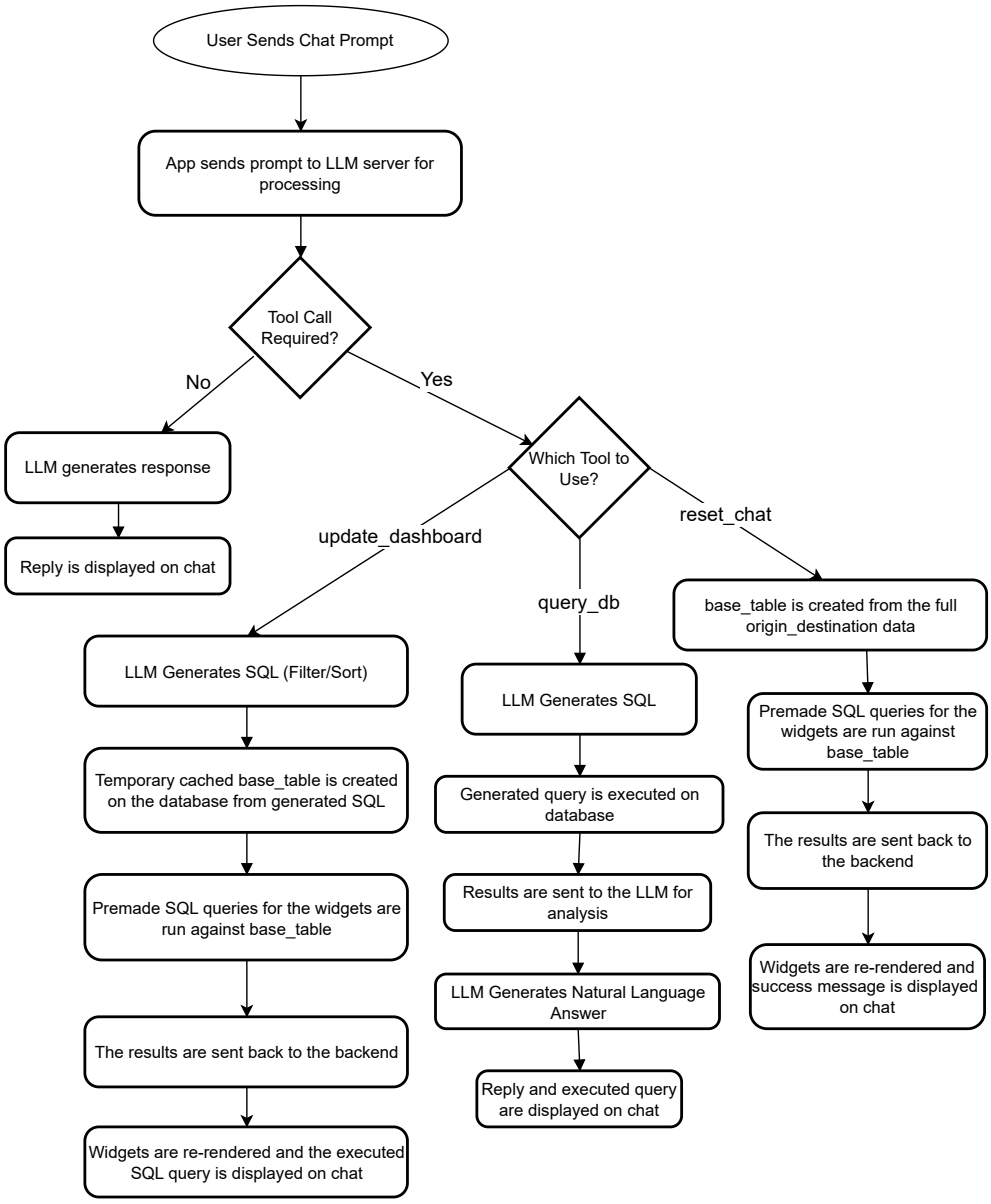


Figure 3: A flowchart illustrating the request fulfillment pipeline for the dashboard.

The first plot in Figure 4 shows how the average load evolves over time. As expected in public transportation, cycles and seasonality are evident. First, daily cycles are clear, since there is a short overnight period when passenger and vehicle counts drop markedly. Lower amplitude cycles are also apparent during the weekend period of 2–3 March. Finally, spikes occur during rush hours as passenger volumes increase.

In Figure 5, the spikes seen in the previous line plot are summarized by hourly intervals. Hotter colors between 6:00 and 7:00 indicate higher load. In the late afternoon, a similar pattern appears between 17:00 and 18:00. A smaller, yet relevant, peak is also visible around 12:00.

Another informative view appears in Figure 6a, where passenger load is mapped to routes with a color scale that reveals the spatial distribution of demand. Beyond directional flows, this visualization highlights corridors with consistently higher passenger volumes. Figure 6b shows a table that summarizes the data selected from the user's prompt.

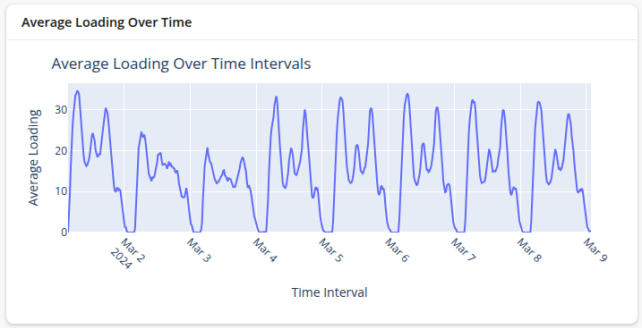


Figure 4: Average load over time, highlighting daily seasonality and rush hour spikes.

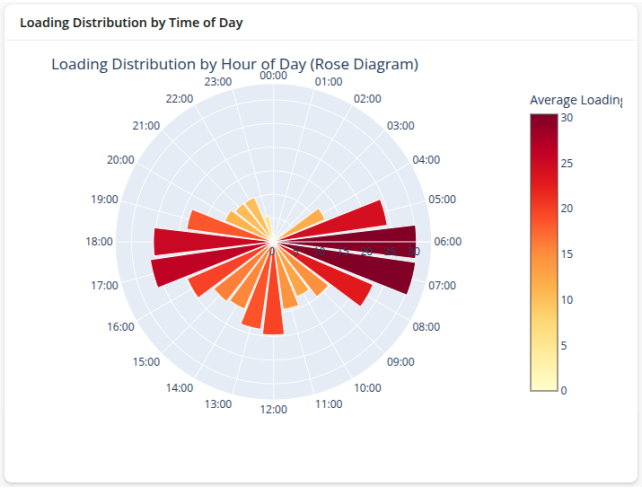


Figure 5: Rose diagram of hourly load distribution, highlighting morning and evening peaks and a smaller midday rise.

An important visualization appears in Figure 7, which summarizes information by neighborhood and highlights regions of the city with higher concentrations of passengers, strong spatial heterogeneity, and clear high-demand coastal-central hotspots.

To explore SUNTInsight in depth, the following examples demonstrate an end to end, decision-centric workflow. We illustrate how public transportation managers mine and visualize relevant information using the following prompt:

Prompt

Data collected on March 5th between 07:00 and 08:00 for the bus line 1637

The LLM agent transforms the natural language text into SQL code, as shown in Figure 8. The resulting query is executed against the database and displayed on the page for inspection, allowing users to verify that it requests the correct information.

After retrieving the required data, SUNTInsight updates the visualization panels as shown in Figure 9. Panel (a) indicates that the selected interval contains 428 records collected from 103 stops served by Line 1637, shown in Panel (c), with a very high occupancy rate, also confirmed by Panel (b), as expected during rush hour.

In Panel (d), SUNTInsight details how occupancy is distributed by neighborhood. Lighter colors indicate initial stops where occupancy remains low. This view highlights that a single line exhibits distinct behaviors along its route. To reduce occupancy in busy neighborhoods, adding buses across the entire line is unlikely to fix localized peaks and may increase traffic, targeted measures are preferable.

6 Final Remarks

This paper presented SUNTInsight, an agentic system that integrates large language models, machine learning, and visual analytics to support human decision making in public transportation. The system converts free form prompts into well formed SQL, retrieves and prepares data that feed BI board visualizations in real time.

In a case study of Salvador-Brazil, examples point out how the application has the potential to improve decision-making workflows. An example showed how line level averages can hide localized stress points. Rather than increasing fleet size across the entire route, the analysis enabled the identification of more resource-efficient actions, such as short turns or segment specific headway control on the high demand corridor.

Beyond analytics quality, the system was engineered for safe operation. It applies the Principle of Least Privilege across data access and code execution, isolates the runtime for generated code, and records auditable traces of prompts, queries, and outputs. These controls help mitigate the dual risk of unintentional model errors and malicious prompt attempts as well as tracking the Agent's performance, which is essential for real world deployments.

Finally, SUNTInsight demonstrates that Agentic AI can serve as a practical co-pilot for transit management, joining natural language interaction, data analysis and visualization into a single workflow that turns heterogeneous data into timely, actionable insights while keeping humans in control.

Acknowledgments

This work was supported by CNPq (Brazilian National Council for Scientific and Technological Development) grants [404771/2024-6, 406354/2023-5, 312755/2023-6, 313053/2023-5], UFBA/CNPq 68/2022 - MAI/DAI, Maria Emilia Foundation grant to 01/2023, INCITE FAPESB (Bahia Research Foundation) grant TO PIE0002/2022, CAPES (Coordination for the Improvement of Higher Education Personnel – Brazil), and FAPESB grant [1589/2021].

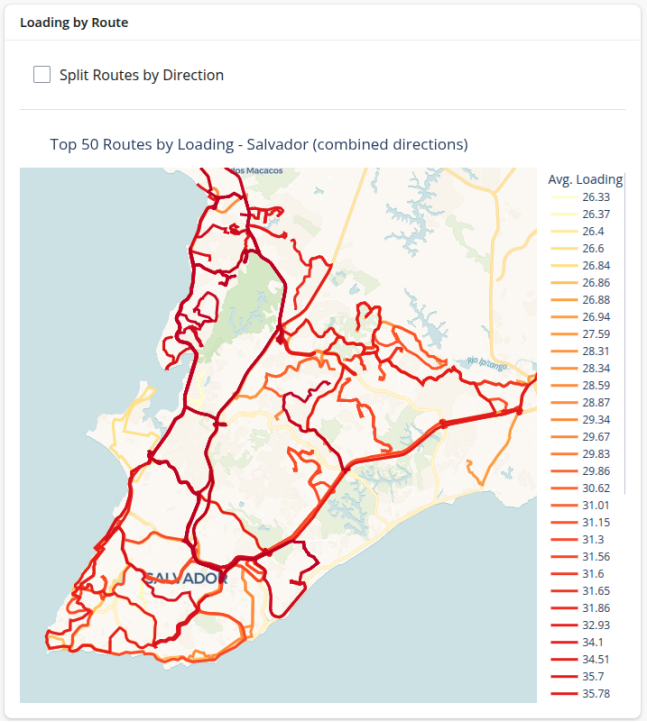
References

[1] Deepak Bhaskar Acharya, Karthigeyan Kuppan, and B Divya. 2025. Agentic ai: Autonomous intelligence for complex goals–a comprehensive survey. *IEEe Access* (2025).

[2] E. F. Codd. 1970. A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* 13, 6 (1970), 377–387.

[3] Marcos V Ferreira, Matheus Souza, Tatiane N Rios, Islame FC Fernandes, Jorge Nery, João Gama, Albert Bifet, and Ricardo A Rios. 2025. Salvador Urban Network Transportation (SUNT): A Landmark Spatiotemporal Dataset for Public Transportation. *Scientific Data* 12, 1 (2025), 1320.

[4] Groq. 2024. Introducing Llama-3-Groq-Tool-Use Models. https://groq.com/blog/introducing-llama-3-groq-tool-use-models Accessed: October 23, 2025.



(a)

Origin-Destination Data (First 100)

Route	Direction	Stop Location	Stop Time	Loading	Boardings	Alightings
164903	I	Av. Suburbana S/N Bahia Brasil	2024-03-07 05:16:23	45.944443	0	0
164903	I	Av. Afrânio Peixoto 9307-9731 - Pepiperi Salvador - BA 40720-270 Brasil	2024-03-07 05:16:48	45.92346	1	1.0209876
164903	I	Av. Afrânio Peixoto 8889-8979 - Praia Grande Salvador - BA 40720-690 Brasil	2024-03-07 05:17:54	45.92346	0	0
164903	I	Av. Afrânio Peixoto 85e - Escada Salvador - BA 40710-250 Brasil	2024-03-07 05:18:17	45.92346	0	0
164903	I	Av. Afrânio Peixoto 43 - Escada Salvador - BA 40710-250 Brasil	2024-03-07 05:19:05	45.92346	0	0

Viewing rows 1 through 5 of 100

(b)

Figure 6: (a) Passenger load by route, revealing the spatial distribution of demand and highlighting high-flow corridors. (b) Table summarizing data selected by the user and used to create all visualizations.

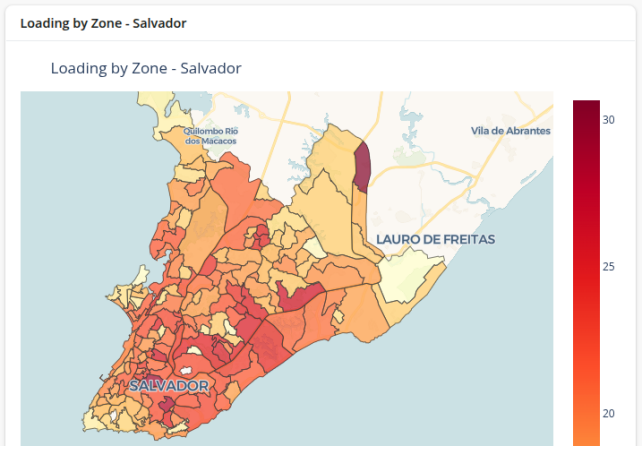


Figure 7: Passenger load aggregated by neighborhood, highlighting areas with higher concentrations of demand.

SQL

```
SELECT origin_destination.*
FROM origin_destination
JOIN trip ON
  origin_destination.trip_date =
  trip.trip_date AND
  origin_destination.trip_id =
  trip.trip_id
WHERE trip.route_short_name = '1637' AND
  origin_destination.trip_date =
  '2024-03-05' AND
  origin_destination.stop_time BETWEEN
  '2024-03-05 07:00:00' AND '2024-03-05
  08:00:00'
```

Figure 8: SQL command to produce the visualization required by the user.

[5] Xusen Guo, Xinxi Yang, Mingxing Peng, Hongliang Lu, Meixin Zhu, and Hai Yang. 2025. Automating traffic model enhancement with AI research agent. *Transportation Research Part C: Emerging Technologies* 178 (2025), 105187.
[6] Senyun Kuang, Yang Liu, Xiaobo Qu, and Yintao Wei. 2025. Traffic-IT: Enhancing traffic scene understanding for multimodal large language models. *Transportation Research Part C: Emerging Technologies* 180 (2025), 105325.
[7] Doaa Mahmud, Hadeel Hajmohamed, Shamma Almentheri, Shamma Alqaydi, Lameya Aldhaheri, Ruhul Amin Khalil, and Nasir Saeed. 2025. Integrating LLMs With ITS: Recent Advances, Potentials, Challenges, and Future Directions. *IEEE Transactions on Intelligent Transportation Systems* 26, 5 (2025), 5674–5709. doi:10.1109/TITS.2025.3528116
[8] Mistral AI team. 2024. Mistral NeMo. https://mistral.ai/news/mistral-nemo Accessed: October 23, 2025.
[9] Mohammad Movahedi and Juyeong Choi. 2024. The crossroads of llm and traffic control: A study on large language models in adaptive traffic signal control. *IEEE Transactions on Intelligent Transportation Systems* (2024).

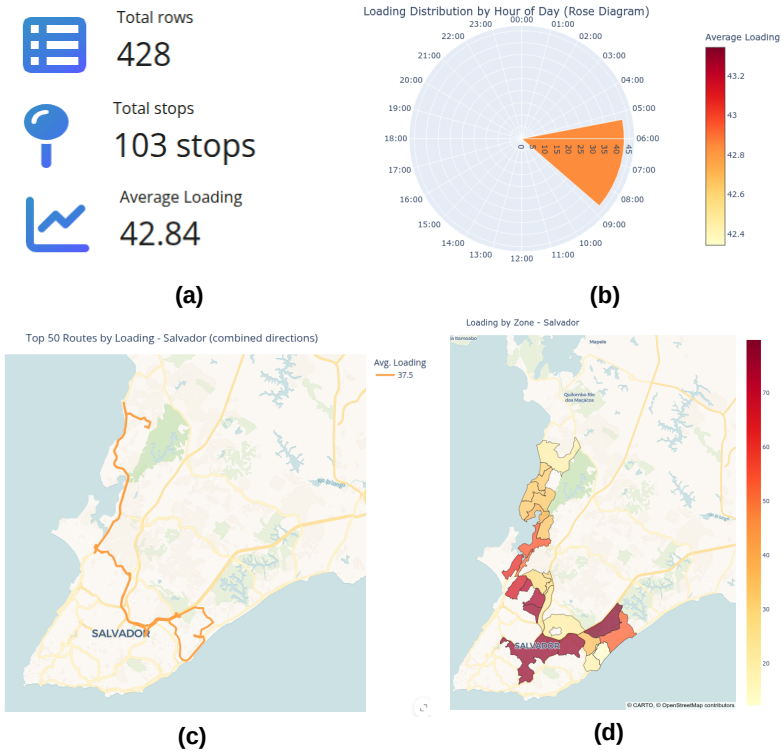


Figure 9: Analyzing data from Line 1637 on March 5th between 5-7 am.

[10] Ollama. 2025. deepseek-r1:32b. https://ollama.com/library/deepseek-r1:32b Accessed: October 23, 2025.

[11] Ollama. 2025. mistral-small. https://ollama.com/library/mistral-small Accessed: October 23, 2025.

[12] Ollama. 2025. qwen2.5-coder:14b. https://ollama.com/library/qwen2.5-coder:14b Accessed: October 23, 2025.

[13] Ollama. 2025. qwen3:32b. https://ollama.com/library/qwen3:32b Accessed: October 23, 2025.

[14] OpenAI. 2025. Introducing gpt-oss. https://openai.com/index/introducing-gpt-oss/ Accessed: October 23, 2025.

[15] J.H. Saltzer and M.D. Schroeder. 1975. The protection of information in computer systems. *Proc. IEEE* 63, 9 (1975), 1278–1308. doi:10.1109/PROC.1975.9939

[16] Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. 2026. AI Agents vs. Agentic AI: A Conceptual taxonomy, applications and challenges. *Information Fusion* 126 (2026), 103599.

[17] Zhiqi Shao, Haoning Xi, Haohui Lu, Ze Wang, Michael GH Bell, and Junbin Gao. 2025. A spatial–Temporal Large Language Model with Denoising Diffusion Implicit for predictions in centralized multimodal transport systems. *Transportation Research Part C: Emerging Technologies* 179 (2025), 105249.

[18] Mohamed R Shoaib, Heba M Emara, and Jun Zhao. 2023. A survey on the applications of frontier ai, foundation models, and large language models to intelligent transportation systems. In *2023 International Conference on Computer and Applications (ICCA)*. IEEE, 1–7.

[19] Leizhen Wang, Peibo Duan, Zhengbing He, Cheng Lyu, Xin Chen, Nan Zheng, Li Yao, and Zhenliang Ma. 2025. Agentic Large Language Models for day-to-day route choices. *Transportation Research Part C: Emerging Technologies* 180 (2025), 105307.

[20] Xiaojie Yang, Yicheng Tao, Hangli Ge, Zipei Fan, Rajendra Akerkar, and Noboru Koshizuka. 2025. LLeCaT: LLM Enhanced Causality-Aware Traffic Accidents Post-Effects Prediction. *IEEE Transactions on Intelligent Transportation Systems* (2025).

[21] Jiangbo Yu. 2025. Preparing for an agentic era of human-machine transportation systems: Opportunities, challenges, and policy recommendations. *Transport Policy* 171 (2025), 78–97. doi:10.1016/j.tranpol.2025.05.030

[22] David Zax. 2025. What is AgentOps? IBM Think. https://www.ibm.com/think/topics/agentops Accessed: November 4, 2025.