IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 June 2025, accepted 13 July 2025, date of publication 17 July 2025, date of current version 23 July 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3590264

APPLIED RESEARCH

Agentic AI Systems: Architecture and Evaluation Using a Frictionless Parking Scenario

ALAA KHAMIS, (Senior Member, IEEE)
AI for Smart Mobility Laboratory, Interdisciplinary Research Center for Smart Mobility and Logistics, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

e-mail: alaa.rashwan@kfupm.edu.sa

This work was supported by the Deanship of Research (DOR) at King Fahd University of Petroleum and Minerals (KFUPM) through the Agentic AI-Based Framework for Seamless Integrated Mobility under Grant ECR241-ISE-301.

ABSTRACT An AI agent is a goal-oriented autonomous computational entity that connects reasoning and action using large language models (LLMs) or Large Reasoning Models (LRMs), memory systems, and external tools to achieve contextually intelligent outcomes. An agentic AI system comprises and coordinates multiple specialized AI agents to achieve complex goals with minimal or no human supervision. In this paper, agentic AI systems are elucidated through a frictionless-parking scenario that illustrates core components of the system, namely the design of individual AI agents, their interaction mechanisms (handoff and cueing), and potential cooperation patterns (augmentative, integrative, and debative). This scenario provides an experimental test-bed for demonstrating how agentic AI can deliver context-aware, personalized services. A six-factor factorial experiment evaluates performance of the implemented AI agents across five user profiles, four GPT backbones, three entropy levels, three verbosity settings, three query complexities, and three prompt specificity levels. To guarantee that each agent's recommendation is both plausible and constraint-compliant, the system combines guardrails, which reject or rewrite answers that violate user requirements, with Chain-of-Thought prompting that exposes intermediate reasoning steps for internal self-checks. Key metrics (agent's response time or latency and lexical consistency) show that a lightweight gpt-4o-mini backbone and concise verbosity minimize latency, while medium prompt specificity and moderate query complexity optimize consistency. Decoding entropy influences stylistic diversity without significant latency costs but reduces consistency at high settings. User intent, particularly for creative or ambiguous profiles, drives variability. A SHAP analysis ranks model size, verbosity, and prompt specificity as top performance drivers.

INDEX TERMS Agentic AI, AI agents, large language models (LLMs), large reasoning models (LRMs), multi-agent system, parking, urban mobility.

I. INTRODUCTION

Agentic AI represents a new generation of intelligent systems characterized by autonomy, goal orientation, and adaptive planning capabilities. According to IBM, an agentic AI system is one that aims to solve long-horizon tasks through sophisticated reasoning with minimal human supervision. It consists of a collection of large language models (LLMs) or Large Reasoning Models (LRMs)-based agents that interact with humans and the environment to fulfill specified goals [1]. Similarly, NVIDIA defines agentic AI as a class of systems that use sophisticated reasoning and iterative planning to autonomously solve complex, multi-step problems [2]. These systems go beyond static task completion, exhibiting capabilities such as memory, tool use, self-reflection, and coordination with other agents. Acharya et al. [3] define agentic AI as autonomous systems designed to pursue complex goals in dynamic environments with minimal human intervention. These systems are capable of independently pursuing long-term objectives, making

The associate editor coordinating the review of this manuscript and approving it for publication was Sawyer Duane Campbell.

decisions, executing complex multi-step tasks, and proactively orchestrating processes–demonstrating sophisticated reasoning and adaptability.

The emergence of agentic AI represents a paradigm shift in how intelligent systems perceive, reason, and act in dynamic, real-world environments. Unlike traditional AI models that operate under narrow, task-specific parameters, agentic AI systems are designed to exhibit autonomy, proactivity, memory, and adaptive tool use. These capabilities allow the system to coordinate, plan, and respond to evolving user needs and environmental constraints. Powered by LLMs or LRMs, structured memory, and integrated toolkits, these agents are increasingly capable of engaging in multi-step reasoning and collaborative behaviors, unlocking new possibilities in fields such as data operations [4], networking [5], retail systems [6], marketing research, strategy and actions [7], and urban mobility.

Urban mobility, in particular, presents a compelling domain for deploying agentic AI systems. As urbanization accelerates globally, the demand for smarter, more personalized mobility solutions continues to grow. According to the United Nations' World Urbanization Prospects report [8], two-thirds of the global population is projected to reside in urban areas, reversing the rural-urban distribution observed in the mid-20th century. One of the most persistent challenges in dense urban environments is parking—an activity that affects not only drivers but also pedestrians, businesses, and city planners. In a survey conducted by EasyPark Group [9], 32% of European respondents reported difficulty finding parking at least once a month, with that figure rising to 50% in southern countries like Spain, Italy, and France. Additionally, over one-third of respondents spent between 5 to 20 minutes searching for a parking spot, highlighting the inefficiency and frustration that characterize the current state of urban parking. The parking innovation that most respondents look for in the future is the ability to easily be directed to a free parking spot (39%), followed by automatic payment of parking sessions (30%).

In total, 21% of the European respondents use parking apps to park in a parking garage or gated parking area. In the Nordics, the corresponding number is 40% [9]. Parking apps like EasyPark, Parkopedia, Parkster, SharePm, Parkamo, ParkMe, and ParkingPay often focus on specific parts of the parking journey, such as finding or paying for parking, but they fail to address the process comprehensively. While they may offer features like real-time availability or booking, they typically overlook other critical aspects, such as persona-specific or context-specific hard and soft constraints to be considered during the search, seamless navigation to the spot with micro-routing, or a smooth transition from a parking to departure. As a result, these apps leave gaps in the user experience, particularly in enabling natural interaction and in integrating the entire journey from search and guidance to entry, parking, and exit.

To address these gaps, the paper presents an agentic AI architecture tailored to enable frictionless urban parking. Frictionless urban parking seamlessly guides drivers from intelligent search and on-route guidance, through micro-routing and access facilitation, to on-spot monitoring and seamless departure. It integrates adaptive navigation, real-time stall and charger availability, on-site wayfinding, and fully automated payment and exit, eliminating manual steps and delays throughout the entire journey. The envisioned architecture orchestrates a multi-agent framework that spans all stages of the parking process—from informative search and on-route guidance to micro-routing, access facilitation, on-spot monitoring, and seamless departure. Each stage is managed by a specialized AI agent operating through well-defined interaction mechanisms and cooperation patterns, ensuring a cohesive and user-centered experience. Although the framework envisions multiple cooperative AI agents, this paper details only the implementation and evaluation of two of these agents—the Interaction Agent and the Informative Search Agent. The remaining agents are included at a conceptual level to illustrate the broader potential of agentic AI across the complete parking workflow and will be implemented and evaluated in future work.

The scope of this paper is threefold:

1) to investigate the broader potential of agentic AI for delivering context -aware, personalized services within smart mobility ecosystems;
2) to conceptualize a modular AI-agent framework that demonstrates the potential of specialized AI agents, their interaction mechanisms, and cooperation patterns in enabling frictionless urban parking; and
3) to conduct comprehensive evaluation experiments that systematically vary user profiles, LLM backbones, decoding entropy, verbosity caps, query complexity, and prompt specificity, thereby quantifying the agent's performance under diverse real-world conditions.

The remainder of this paper is organized as follows. Section II provides an overview of software agents, tracing the evolution from classical, rule-based systems to modern LLM-based agents. Section III introduces the general structure of AI agents, detailing key components such as prompt template, foundational model, short-term memory, long-term memory, and tool use. Section IV presents the envisioned agentic AI architecture for frictionless parking, describing the specialized agents and the patterns of cooperation and interaction mechanisms among them. Section V outlines a use case implementation to evaluate the agent's performance and ability to adapt with real-world parking constraints. Finally, Section VI concludes the paper and discusses directions for future work.

II. FROM CLASSICAL TO LLM-BASED AGENTS

The notion of an agent is not new and has been widely explored across disciplines such as philosophy, artificial intelligence, distributed systems, and robotics. Semantically, an agent is understood as an autonomous entity that possesses an ontological commitment and operates with an agenda of its own. Crucially, autonomy is a defining characteristic: an entity that merely follows instructions cannot be regarded as an agent in the true sense. The following subsections present an overview of classical artificial agents and LLM-based agents, highlighting the key differences between them.

A. CLASSICAL ARTIFICIAL AGENTS

Within computer science and AI, the concept of artificial agents as computational entities varies depending on the domain of application. Accordingly, numerous definitions have been proposed over the years to capture the essential nature of agency. One of the most widely cited definitions comes from Russell and Norvig [10], who describe an agent simply as an entity that can perceive its environment and act upon it. This foundational definition has been extended in various ways to account for more complex behavioral properties.

Wooldridge and Jennings [11], for instance, emphasized four core attributes that elevate computational systems to agent status: autonomy, reactivity, proactiveness, and social ability. Nareyek [12] adds that agents are proactive autonomous units capable of goal-directed behavior and interaction. Zambonelli et al. view agents as software entities capable of achieving goals through high-level interaction protocols [13]. Bellifemine et al. place agents within the paradigm of distributed object technologies enhanced with artificial intelligence and speech act theory, highlighting the agent as a software abstraction that is autonomous, proactive, and communicative [14]. Singh and Huhns further extend the definition by adding the dimension of persistent identity, defining an agent as an active computational entity that maintains an identity, perceives and reasons about its environment, and communicates effectively with both human and artificial agents [15]. Popularized descriptions have also been used to make the concept accessible. For instance, Wayner illustrates agents through a hypothetical narrative where software agents assume the role of a secret agent, capable of intelligent decision-making, autonomy, reactivity, mobility, and communication — encapsulating the aspirational capabilities of futuristic agent-based systems [16].

Synthesizing across these perspectives, artificial agents are generally recognized by a set of key characteristics. They are autonomous, meaning they possess control over their own actions and decision-making processes. They are reactive, capable of perceiving and responding to environmental stimuli. They are proactive, pursuing goals through self-initiated actions. Agents are also social, engaging in interaction with other agents or humans to achieve system-level goals. Finally, agents are often learnable, enabling them to improve performance and adapt over time through communication and experience.

B. LLM-BASED AGENTS

In the field of agentic AI, researchers are increasingly focused on developing systems that not only process information but also exhibit autonomy by making decisions and taking actions based on context and environmental factors. This level of intelligence, where systems act with agency, has the potential to significantly expand the scope of AI applications, enabling more dynamic and complex problem-solving. A major advancement in this direction has been the emergence of AI agents powered by Large Language Models (LLMs) or Large Reasoning Models (LRMs) [17], which extend the traditional agent paradigm. These agents not only demonstrate core traits such as autonomy, reactivity, proactiveness, and social ability, but also possess the capacity for rich natural language understanding, contextual reasoning, multi-step planning, and the use of external tools to take appropriate actions aligned with user goals and environmental conditions.

An AI agent is defined as an autonomous application that observes its environment, reasons about its goals, and proactively takes actions using available tools to achieve those goals with minimal human intervention [18]. A modern AI agent augmented with an LLM can dynamically interpret user intent, manage short- and long-term memory, perform complex decision-making processes in real time, and query external tools. These agents are also capable of handling complex, multimodal, and multilingual queries, allowing them to process diverse inputs such as text, speech, images, or mixed-language interactions. Unlike earlier rule-based or logic-driven agents, LLM-powered agents operate with a high degree of flexibility, language fluency, and contextual awareness, enabling more generalizable and adaptive behavior across diverse domains.

Furthermore, the integration of structured long-term memory, including semantic memory (factual knowledge, such as user preferences or environment features), episodic memory (records of past interactions and experiences), and procedural memory (task execution sequences), enables these agents to simulate human-like cognition. When combined with sensory input (e.g., GPS, sensors) in working memory, external toolkits (e.g., search or data APIs), and learning mechanisms, LLM-based agents represent a significant step toward more robust, general-purpose artificial intelligence systems capable of sustained interaction and goal fulfillment in dynamic environments. According to a survey conducted by LangChain involving 1,300 professionals [19], there is widespread appreciation for the diverse capabilities of AI agents. These include managing multi-step tasks, automating repetitive tasks, task routing and collaboration, and demonstrating human-like reasoning. Table 1 highlights the main differences between classical agents and LLM-based AI agents.

LAM is a closely related term to LLM-based AI agent, referring to Large Action Models that aim to execute complex, multi-step tasks by translating human intent into action. However, everything LAMs are proposed to do, such as reasoning, planning, and acting, LLM-based agents are already capable of, often with greater flexibility and maturity. With high-quality prompting and fine-tuned

TABLE 1. Classical agents versus LLM-based AI agents.

Aspect	Classical Agents	LLM-based AI Agents
Architecture	Typically rule-based or logic-driven	Neural network-based, leveraging large-scale pretrained language models
Perception and Sensing	Limited to predefined inputs and formats	Capable of handling multimodal inputs (e.g., text, images, speech)
Language Under-standing	Rigid, based on predefined templates or grammars	Rich natural language understanding with contextual sensitivity
Planning and Reasoning	Hand-coded rules or symbolic planners	Emergent reasoning and multi-step planning through language-based prompting
Learning Capability	Often static or requires manual retraining	Can learn through fine-tuning, reinforcement, or in-context (few-shot or zero-shot) learning
Tool Use for Actions	Limited to preprogrammed actions or APIs	Dynamically uses external tools and APIs based on interpreted user intent
Memory Systems	Minimal or predefined context handling	Supports short-term (working) and structured long-term memory (semantic, episodic, procedural)
Multilingual and Multimodal Support	Generally monolingual and unimodal	Capable of handling multilingual queries and multimodal inputs
Adaptability	Task-specific and brittle to change	Highly flexible and adaptable across domains and tasks
Social and Commu-nicative Abilities	Basic message passing or command-response	Rich dialog management and natural interaction with humans and other agents

orchestration, LLM agents can manage sophisticated workflows, invoke tools, and adapt dynamically to user goals. In contrast, the concept of LAMs remains relatively vague, lacking a clear architectural framework or standard use cases. As a result, LLM-based agents currently offer a more concrete and deployable foundation for agentic AI systems.

The following section provides a detailed description of the architecture of LLM or LRM-based agents—referred to hereafter as AI agents—highlighting their core components, memory systems, reasoning capabilities, and tool integration.

III. AI AGENT ARCHITECTURE

An AI agent is an autonomous, goal-oriented, and context-aware computational system that perceives its environment, maintains and utilizes memory, reasons about its objectives, and takes actions—often through the use of external tools—to fulfill user intentions. It leverages a LLM or LRM to interpret complex, multimodal user queries, and relies on both short-term (working) and long-term (semantic, episodic, and procedural) memory to support contextual understanding, reasoning, multi-step planning, and actions. Reasoning capabilities of the agent is enabled through prompt engineering frameworks such as ReAct [20], Chain-of-Thought (CoT) [21] or Tree-of-Thoughts (ToT) [22]. Figure 1 illustrates the different components of an AI agent.

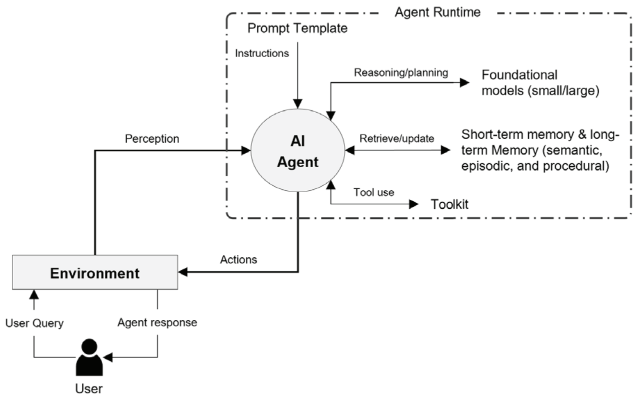

FIGURE 1. AI agent architecture.

To better understand these components, consider an example within the context of a parking search scenario. Suppose a driver is navigating through downtown Toronto and asks their in-vehicle AI assistant: "Can you help me find a parking spot?". This assistant is an AI agent equipped with access to real-time GPS data from the vehicle, a prompt template for instructions, a foundational model for reasoning and planning, a working memory for short-term contextual awareness, and a long-term memory for storing persistent user preferences and an access to toolkit.

- **Prompt template:** defines the reasoning steps the AI agent should follow, such as checking the vehicle's current location via GPS and filtering parking options based on user preferences; for example, it instructs the agent to prioritize on-street spots with EV charging if the user has indicated such a preference.
- **Foundational model:** the model, whether small or large, general-purpose or fine-tuned, unimodal or multimodal, interprets the user's request (e.g., "Can you help me find a parking spot?"), determines that a parking search should be performed based on the current context, selects an appropriate tool (either predefined or chosen dynamically by the agent), and generates a natural language response such as, "There is an available spot 200 meters ahead on Wellington Street with EV charging at $2 per half-hour."

- **Short-term memory:** helps the agent manage the immediate interaction and stores temporary information such as current GPS location (e.g., Front Street West), the parking request just made and the search results, like a nearby available parking spot with the rate and time limit.
- **Long-term memory:** stores information learned or configured over time as part of the user profile such as driver's parking preferences (e.g., on-street parking vs. underground parking), whether the driver needs accessible parking, whether the vehicle requires charging stations and past interactions or frequently visited locations. This memory can include semantic, episodic, and procedural components.
  - Semantic memory stores general facts and structured knowledge, such as the user's preference for on-street parking and the requirement for EV charging.
  - Episodic memory captures past experiences, such as the locations where the driver has successfully parked before or past interactions where certain recommendations were accepted or rejected.
  - Procedural memory encodes the agent's learned sequences of actions, including how to query real-time parking data, filter results based on constraints, and initiate navigation upon user confirmation.

  For example, long-term memory allows the AI agent to remember a driver's preferred garages, typical arrival and departure times, past satisfaction ratings or payment histories, and even seasonal or weekday/weekend availability trends. When a new search is initiated, the agent consults this memory to bias its candidate list toward locations that the user has previously favored (or to avoid ones they rejected), to anticipate likely occupancy or pricing fluctuations based on historical data, and to tailor walk-distance trade-offs according to the driver's habitual tolerance. In this way, Long-term Memory transforms each recommendation from a one-off lookup into an informed suggestion that evolves with the user's behavior and the city's parking dynamics.
- **Toolkit:** enables the agent to interact with external systems and services to support various stages of the parking journey. This may include querying real-time parking availability APIs, interacting with booking systems to reserve spots, accessing mapping and routing services for navigation, retrieving contextual data such as pricing, accessibility, EV charging availability, and user reviews or invoking payment systems for seamless billing. These tools extend the agent's capabilities beyond reasoning and decision-making, allowing it to take meaningful action within its environment.

Figure 2 presents a sequence diagram illustrating the interaction between the driver, the AI agent, and its internal components during the parking spot search scenario, including preference resolution, memory access, and real-time data retrieval using tools. The sequence begins when the Driver issues a request to the Agent, such as "Can you help me find a parking spot?" The Agent receives the query and forwards it to the Model for interpretation and reasoning.

Upon receiving the query, the Model determines that it requires both the current location of the vehicle and the driver's parking preferences to proceed with an informed response. The Model communicates this requirement to the Agent.

The Agent then sends a request to the Short-term Memory to retrieve the vehicle's current location. In parallel, it sends a request to the Long-term Memory to retrieve the user's stored parking preferences. The Short-term Memory responds with the current GPS location, while the Long-term Memory replies that no preferences are available.

Based on the absence of preference data, the Agent contacts the Model to request a clarification prompt. The Model generates a natural language question such as, "Do you have any parking preferences?" and returns it to the Agent, which relays it to the Driver.

The Driver responds, for example, "Accessible parking lots with charging facility." The Agent updates user preferences in the Long-term Memory and asks the Model to generate a structured query with the current location and user's preferences. The Model retruns this structured query to the Agent that sends it to the Tool (e.g., a real-time parking API) to retrieve filtered parking information. The Model formulates a user-friendly natural language response and returns it to the Agent, such as: "There is an available accessible parking spot with EV charging 200 meters ahead on Wellington Street at $2 per half-hour." The Agent delivers this response to the Driver and updates the Long-term Memory with the newly acquired user preferences for use in future interactions.

Chain-of-Thought (CoT) reasoning is critical because it exposes the intermediate steps the LLM-based agent takes when decomposing a parking request into sub-tasks such as geo-coding the destination, filtering candidate garages, estimating walking distance and ranking by user preferences. By prompting the backbone model to "think step-by-step," the agent can explicitly verify each partial result before committing to the next API call, thereby reducing error propagation and improving factual reliability. The CoT reasoning of the agent can be summarized in the following steps:

1) Interpret the user query as a parking search request.
2) Identify the need for current location and user preferences.
3) Retrieve current location from short-term memory.
4) Check long-term memory for stored preferences.
5) Detect missing preferences and generate a clarifying prompt.
6) Ask the user for their parking preferences.
7) Receive and interpret the user's response.

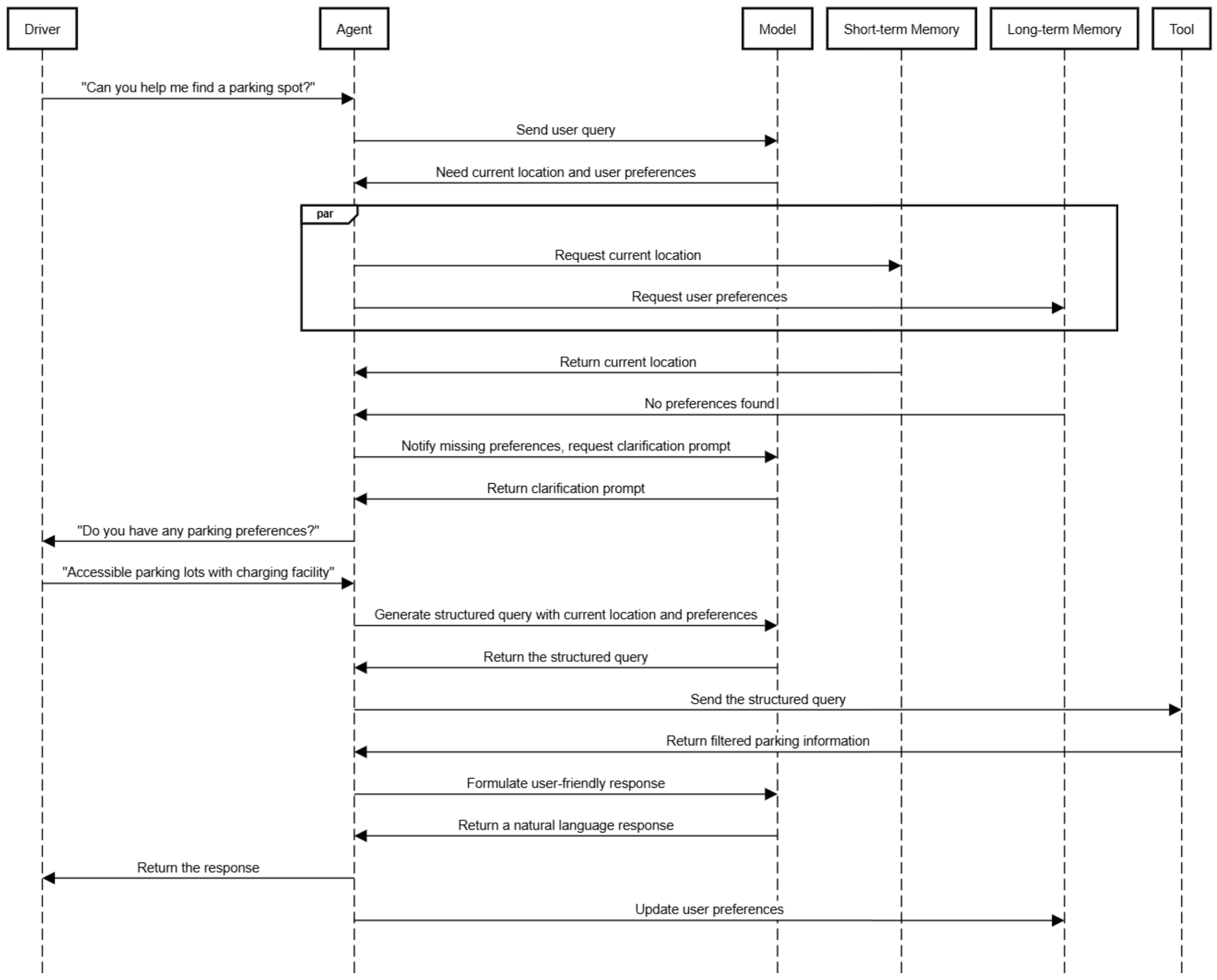
Driver
Agent
Model
Short-term Memory
Long-term Memory
Tool
"Can you help me find a parking spot?"
Send user query
Need current location and user preferences
par
Request current location
Request user preferences
Return current location
No preferences found
Notify missing preferences, request clarification prompt
Return clarification prompt
"Do you have any parking preferences?"
"Accessible parking lots with charging facility"
Generate structured query with current location and preferences
Return the structured query
Send the structured query
Return filtered parking information
Formulate user-friendly response
Return a natural language response
Return the response
Update user preferences

FIGURE 2. **Sequence diagram of a parking request.**

8) Query the parking toolkit using location and preferences.
9) Receive and filter parking results.
10) Generate a natural language response.
11) Deliver the response to the user.
12) Update long-term memory with new preferences.

This scenario demonstrates how the agent's perception, orchestration, memory systems, language model, and tool integration operate in coordination to deliver context-aware and personalized assistance.

IV. PROPOSED AGENTIC AI ARCHITECTURE

The envisioned architecture encompasses different AI agents with distinct capabilities and functionalities, as illustrated in Figure 3, and is designed to enable seamless coordination through structured cooperation patterns and interaction mechanisms to support a frictionless parking experience. The following subsections describe the role of each agent, the patterns of cooperation through which they collaborate to fulfill complex parking tasks and their interaction mechanisms.

A. AI AGENTS

The frictionless parking journey is handled using the following agents:

1) INTERACTION AGENT

This agent serves as the primary interface between the user and the system. It is responsible for receiving and interpreting user queries, managing dialogue, and orchestrating the activities of other specialized agents. It retrieves and updates user preferences from a personalized cloud-based profile and supports adaptive learning by collecting feedback through post-trip scoring. The role and responsibilities of the Interaction Agent within the frictionless parking system are defined as follows:

- **Name:** Interaction Agent

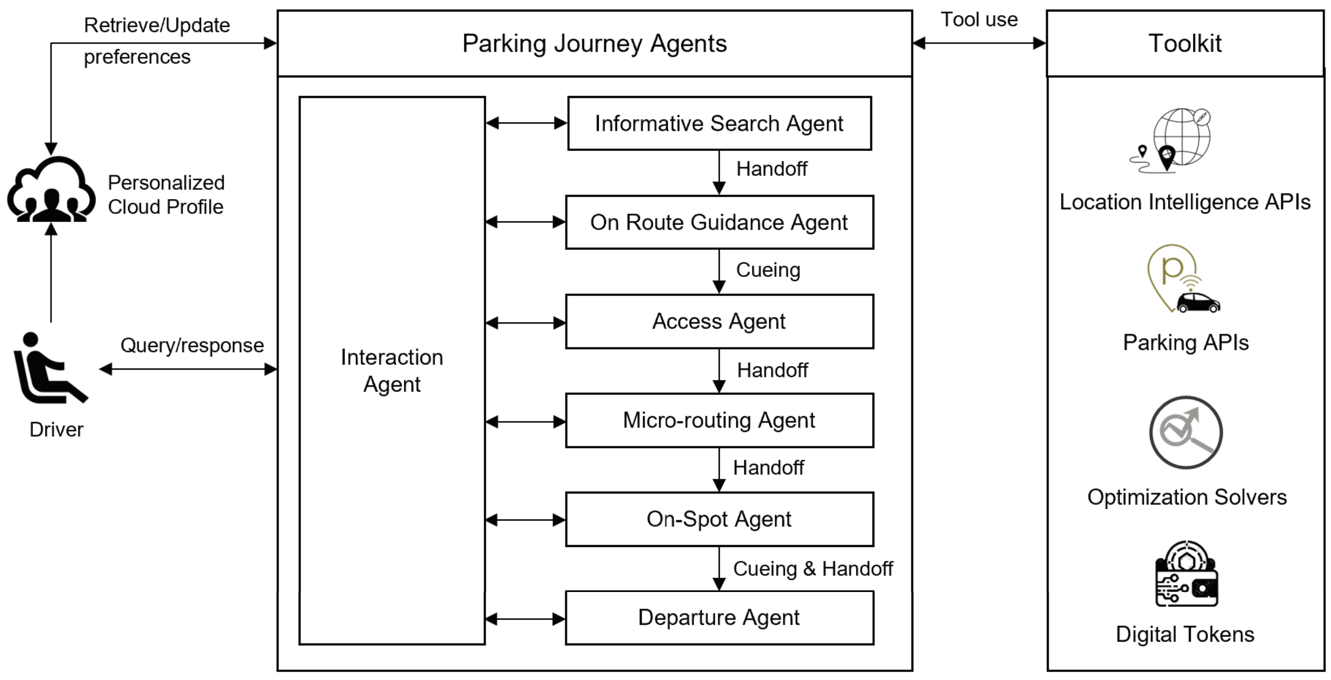

FIGURE 3. **Proposed Agentic AI Architecture.**

- **Instructions:** You are a conversational and orchestration AI agent responsible for managing user interactions and coordinating the behavior of specialized agents in the frictionless parking system. When a user issues a parking-related query, interpret the request, retrieve or update their personalized preferences from the cloud profile, and delegate tasks to the appropriate agents. At the end of the trip, collect user feedback to support post-trip learning and preference refinement.
- **Functions:**
  - Handle user queries via natural language
  - Retrieve and update user preferences
  - Coordinate and delegate to other agents (e.g., Search, Routing, Access)
  - Trigger post-trip scoring to improve personalization

Interaction Agent ensures personalized and context-aware experiences by coordinating reasoning, planning, and decision-making across the system.

2) INFORMATIVE SEARCH AGENT

This agent is tasked with identifying optimal or near-optimal parking spots that align with the user's needs and preferences. It formulates the parking search as a multi-criteria constrained optimization problem, incorporating objective functions such as cost (free, flat, or dynamic rates) and proximity to the destination. It considers hard constraints like required duration, accessibility, vehicle fit (e.g., spot size, height clearance), and time windows (e.g., immediate or future parking needs). Soft constraints, such as preferred parking types (e.g., on-street, underground), are also accounted for. This agent relies on tools such as web search, Parking APIs and optimization solvers. The role and responsibilities of the Informative Search Agent are defined as follows:

- **Name:** Informative Search Agent
- **Instructions:** You are an intelligent search and optimization AI agent responsible for finding an optimal or near-optimal parking spot that aligns with the user's preferences and current context. Treat the problem as a multi-criteria constrained optimization task, considering both hard and soft constraints. Use appropriate tools such as web search, parking APIs or optimization solvers to evaluate available options.
- **Functions:**
  - Formulate the parking search as a multi-objective optimization problem
  - Apply hard constraints (e.g., duration, accessibility, vehicle fit, time window)
  - Incorporate soft constraints (e.g., preferred type of parking)
  - Query real-time parking availability using external APIs
  - Find optimal or near-optimal parking options
  - Return parking options to the Interaction Agent

3) ON-ROUTE GUIDANCE AGENT

This agent manages the user's journey to the selected parking location. It supports functions such as route selection, reservation management, real-time navigation, estimated time of arrival (ETA) prediction, and route adaptation based on dynamic conditions. These include road closures, traffic incidents, crowd levels, and weather conditions near the destination. It integrates with external tools such as Google Maps and Places APIs to provide timely and accurate routing information. The role and responsibilities of the On-Route Guidance Agent are defined as follows:

- **Name:** On-Route Guidance Agent
- **Instructions:** You are a navigation and trip coordination agent responsible for guiding the user to the selected parking location. Your role includes providing real-time navigation, predicting arrival time, and adapting routes based on dynamic conditions such as traffic, road events, crowd levels, and weather. Utilize external tools like Google Maps and Places APIs to enhance navigation accuracy and relevance.
- **Functions:**
  - Initiate navigation to the selected parking spot
  - Predict ETA based on live traffic data and historical data
  - Adapt routes dynamically in response to road events or changing conditions
  - Provide additional contextual information, such as crowd level and weather at the destination
  - Assist in reservation handling and support booking confirmation if applicable

4) ACCESS AGENT

This agent facilitates seamless entry to parking facilities, supporting non-stop touchless access mechanisms such as automatic gate opening, license plate recognition, or valet coordination. It also supports advanced entry methods like digital handshaking using secure tokens, when such infrastructure is available at the parking facility. The role and responsibilities of the Access Agent are defined as follows:

- **Name:** Access Agent
- **Instructions:** You are responsible for enabling seamless and contactless entry into parking facilities. Facilitate non-stop access through mechanisms such as license plate recognition, QR code validation, or digital tokens. Where supported, coordinate valet services or advanced digital handshaking with the parking infrastructure to minimize user friction during the entry process.
- **Functions:**
  - Initiate seamless or contactless entry into parking areas
  - Validate access using digital credentials (e.g., token, license plate)
  - Coordinate with valet services if available
  - Interface with parking infrastructure for gate control and entry logging
  - Ensure compatibility with the user's selected access preferences

Access Agent reduces the friction of physical access and supports the transition to automated and contactless entry systems.

5) MICRO-ROUTING AGENT

This agent handles fine-grained navigation within parking facilities. It localizes the user's vehicle and guides it to a designated or pre-booked spot using real-time occupancy data and indoor navigation capabilities. The role and responsibilities of the Micro-Routing Agent are defined as follows:

- **Name:** Micro-Routing Agent
- **Instructions:** You are responsible for navigating the vehicle within the parking facility to the designated or pre-booked parking spot. Utilize occupancy data and facility layout information to perform spot localization and provide precise, indoor micro-routing. Support routing adjustments in case of spot reassignment or dynamic availability changes.
- **Functions:**
  - Perform parking spot localization based on facility and occupancy data
  - Provide indoor micro-routing directions to the selected spot
  - Monitor real-time spot availability to update routing if needed
  - Support localization in multi-level or complex parking structures

Micro-Routing Agent enhances the in-facility experience by minimizing search time and enabling efficient spot-level routing.

6) ON-SPOT AGENT

This agent monitors and manages the vehicle's status while parked. It enables access to additional services such as EV charging—either through fixed stations, mobile EV chargers or curbside induction—and integrates with security systems to provide surveillance and advanced alerting. This agent ensures safety and service continuity during the parked phase of the journey. The role and responsibilities of the On-Spot Agent are defined as follows:

- **Name:** On-Spot Agent
- **Instructions:** You are responsible for managing the vehicle's status and services while it is parked. Enable on-site functionalities such as electric vehicle (EV) charging via charging stations, mobile EV chargers or curbside induction, and support safety features including security monitoring and advanced alarming if supported by the facility.
- **Functions:**
  - Activate and monitor EV charging services where applicable
  - Detect and report charging status or failures
  - Interface with surveillance systems for parking security

- Enable advanced alerting (e.g., intrusion, movement, or proximity alarms)

7) DEPARTURE AGENT

This agent oversees the final phase of the parking experience. It issues timely notifications before the end of the allowed parking duration, facilitates duration extension when permitted, and provides walking directions back to the vehicle. It also handles seamless payment processing through subscriptions, credit cards, or digital tokens, and provides violation risk alerts based on regulatory data or historical patterns. The role and responsibilities of the Departure Agent are defined as follows:

- **Name:** Departure Agent
- **Instructions:** You are responsible for managing the end-of-parking experience. Monitor time limits and issue timely notifications to the user. Enable session extensions when permitted, and support walking directions to the vehicle. Ensure seamless payment and billing via supported methods, and provide violation risk alerts based on local regulations or historical patterns.
- **Functions:**
  - Track parking session duration and notify before time expiry
  - Facilitate extension requests through the app or interface
  - Provide walking directions to the vehicle
  - Handle payment and billing via subscriptions, credit cards, or digital tokens
  - Warn users of potential parking violations based on time, zone, or historical data

B. COOPERATION PATTERNS

In agentic systems, cooperation is essential for distributing responsibilities, enhancing robustness, and improving solution quality. Within the frictionless parking architecture, three primary forms of cooperation can be manifested among agents: augmentative, integrative, and debative [23]. Each pattern reflects a different mode of knowledge distribution and task allocation among agents.

1) AUGMENTATIVE COOPERATION

This pattern of cooperation occurs when multiple agents possess similar or identical capabilities or know-how and work together to accomplish a task that is too complex, large-scale, or time-sensitive for a single agent. The task is partitioned into smaller, parallelizable sub-tasks, allowing agents to collaboratively achieve the overall objective more efficiently. Formally speaking, augmentative cooperation occurs when a set of agents $\{A_1, A_2, \ldots, A_n\}$ with similar capabilities jointly execute a large task τ by dividing it into smaller sub-tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$:

$$\bigcup_{i=1}^{n} A_i(\tau_i) \rightarrow \text{Action}(\tau) \tag{1}$$

where $\tau = \sum \tau_i$, and each agent A_i is responsible for a sub-task τ_i. This cooperation pattern is used when a single agent cannot handle the task due to scale or time constraints.

For example, multiple Informative Search Agents may be deployed simultaneously to search for optimal parking spots across different geographic zones in a high-demand area as illustrated in Figure 4.

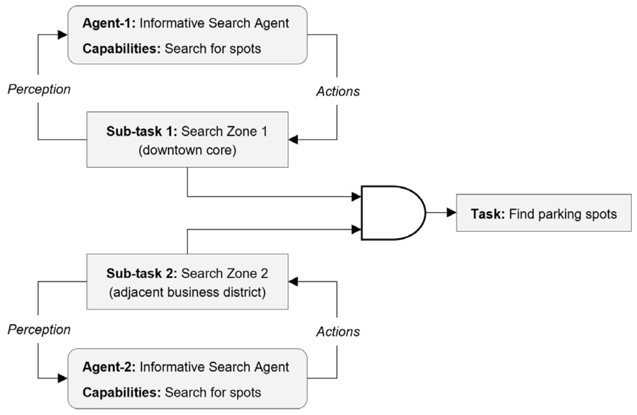
Agent-1: Informative Search Agent
Capabilities: Search for spots
Perception
Actions
Sub-task 1: Search Zone 1 (downtown core)
Task: Find parking spots
Sub-task 2: Search Zone 2 (adjacent business district)
Perception
Actions
Agent-2: Informative Search Agent
Capabilities: Search for spots

FIGURE 4. Augmentative cooperation between the agents.

Each agent performs the same type of search and optimization logic but focuses on a distinct spatial partition or time window. Their results are then combined by the Interaction Agent to present the user with a unified set of options.

2) INTEGRATIVE COOPERATION

This pattern of cooperation involves agents with different, complementary expertise or know-how working together to achieve a complex task that cannot be fulfilled by any individual agent alone. This form of cooperation resembles a team of domain-specific experts contributing distinct but necessary knowledge components. From a formal perspective, integrative cooperation involves agents with complementary capabilities, each solving a domain-specific sub-task τ_i that contributes to a broader task τ. The partial outputs are then combined to form a unified action:

$$\mathcal{F}(A_1(\tau_1), A_2(\tau_2), \ldots, A_n(\tau_n)) \rightarrow \text{Action}(\tau) \tag{2}$$

where $\mathcal{F}$ is a fusion function that integrates the outputs from specialized agents into a coherent, goal-directed action.

In the context of frictionless parking, find-and-navigate complex task can be accomplished as a result of integrating Informative Search Agent and On-Route Guidance Agent as illustrated in Figure 5. The Informative Search Agent selects an optimal or near-optimal parking spot based on optimization criteria, while the On-Route Guidance Agent evaluates the travel time, traffic conditions, route complexity and provides guidance to the selected parking spot.

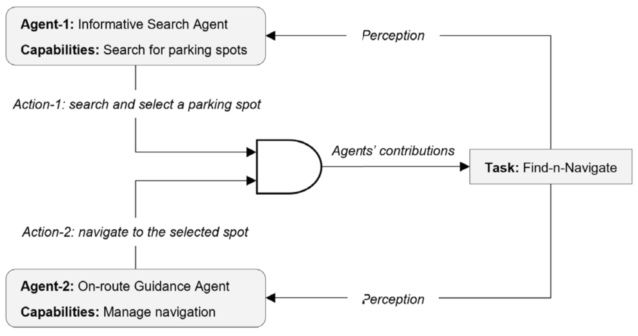
Agent-1: Informative Search Agent
Capabilities: Search for parking spots
Perception
Action-1: search and select a parking spot
Agents' contributions
Task: Find-n-Navigate
Action-2: navigate to the selected spot
Agent-2: On-route Guidance Agent
Capabilities: Manage navigation
Perception

FIGURE 5. Integrative cooperation between the agents.

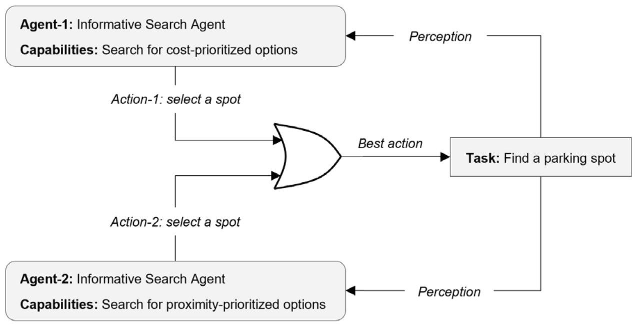
Agent-1: Informative Search Agent
Capabilities: Search for cost-prioritized options
Perception
Action-1: select a spot
Best action
Task: Find a parking spot
Action-2: select a spot
Agent-2: Informative Search Agent
Capabilities: Search for proximity-prioritized options
Perception

FIGURE 6. Debative cooperation between the agents.

3) DEBATIVE COOPERATION

This pattern of cooperation occurs when multiple agents with similar capabilities address the same task independently, then compare and evaluate their respective solutions to select the most suitable one. Theoretically, in debative cooperation, multiple agents $\{A_1, A_2, \ldots, A_n\}$ with similar capabilities independently attempt the same task τ. Their proposed actions are compared, and the best one is selected:

$$\text{Best}(\{A_i(\tau)\}_{i=1}^{n}) \rightarrow \text{Action}(\tau) \tag{3}$$

This cooperation pattern supports diversity of reasoning and redundancy, enabling the system to choose the most suitable outcome through evaluation or consensus.

For example, two or more Informative Search Agents, possibly with different optimization strategies (e.g., cost-prioritized vs. proximity-prioritized), may independently compute the best parking option for the user as illustrated in Figure 6. Their solutions are then compared, either by the Interaction Agent or a voting or weighting mechanism or a Pareto mechanism, and the one that best satisfies the user's preferences and context is selected for presentation.

These cooperation patterns enable the parking agent system to balance efficiency, expertise, and reliability. In practice, a combination of these patterns may emerge dynamically, depending on the user request and system constraints. Their thoughtful application contributes to a robust, personalized, and adaptive parking experience. However, to the best of my knowledge, there is no native implementation of these cooperation patterns supported by existing agentic AI frameworks, especially within no-code or low-code platforms.

C. CUEING AND HANDOFF IN AGENT CHAIN

In agentic systems, cueing and handoff are two key forms of inter-agent interaction that support effective coordination by enabling early context sharing for task preparedness and seamless transfer of control when responsibilities shift between agents across different stages of the task lifecycle.

Cueing is defined as the process in which a cuer agent A_i prepares a cued agent A_j for a future task by transmitting relevant contextual information, without transferring control. Formally, cueing can be expressed as:

$$\text{Cue}(A_i, A_j, d) \rightarrow A_j.\text{prepare}(d) \tag{4}$$

where d is a data payload sufficient to the cued agent A_j for an upcoming event or decision. Cueing improves A_j's response time or performance by enabling early awareness.

Handoff refers to the transfer of control and task responsibility from one agent A_i to another agent A_j, when the task context or scope exceeds A_i's capabilities. Formally:

$$\text{Handoff}(A_i, A_j, \tau) \rightarrow A_j.\text{takeover}(\tau) \tag{5}$$

where τ denotes the task being delegated. Handoff ensures continuity and correctness in multi-agent workflows by routing responsibility to the most appropriate agent.

Table 2 shows different examples of cueing and hand-off in the context of frictionless parking system that can be managed by the Interaction Agents as an orchestration agent. Some agentic AI frameworks such as LangChain and OpenAI Agents SDK support handoff, but there is no native support for cueing.

V. USE CASE IMPLEMENTATION AND EXPERIMENTS

Different workflows are available to facilitate the development of agentic AI systems as summarized in Table 3. In this use case, we focus on implementing the Interaction Agent and Informative Search Agent using OpenAI Agents SDK [24]. This SDK represents a production grade successor to the earlier Swarm experimentation framework and is deliberately structured around a minimal set of primitives.

A. AGENTS

As illustrated in Figure 7, the OpenAI Agents SDK was adopted to implement both an Interaction Agent and an Informative Search Agent, thereby demonstrating that AI agents can engage in natural dialogue while reasoning, planning, and invoking external tools. An agent is defined as an LLM equipped with instructions and tools; handoffs allow one agent to delegate narrowly scoped subtasks to another agent; and guardrails validate every input before it is processed, ensuring correctness and safety throughout the interaction pipeline.

TABLE 2. Examples of cueing and handoff interactions.

From	To	Interaction	Trigger and Purpose
Informative Search Agent	On-Route Guidance Agent	Handoff only	Handoff occurs after the user approves a selected parking spot, transferring control for navigation.
On-Route Guidance Agent	Access Agent	Cueing only	Cueing occurs as the user approaches the parking facility, allowing the Access Agent to prepare for entry authentication.
Access Agent	Micro-Routing Agent	Handoff only	Handoff takes place after successful entry, enabling in-facility routing to the assigned spot.
Micro-Routing Agent	On-Spot Agent	Handoff only	Handoff occurs once the vehicle is parked, transitioning control to the agent managing on-spot services.
On-Spot Agent	Departure Agent	Cueing and Handoff	Cueing occurs during the parked session to share timing and context; handoff happens as the departure time approaches.

TABLE 3. Agentic AI frameworks.

Aspect	No-code/Low-code	Code-first
Development Interface	Visual drag-and-drop interfaces (GUI)	Programmatic scripting in Python or similar languages
User Target	Non-technical users or rapid prototyping by technical teams	Developers and researchers with programming experience
Flexibility and Customization	Limited flexibility, predefined components and integrations	High flexibility with full control over logic, tools, and workflows
Scalability and Complexity	Suitable for simple to moderately complex workflows	Suitable for highly complex, dynamic, and large-scale agent systems
Multi-agent Orchestration	Basic support or limited orchestration logic	Advanced orchestration models (centralized, decentralized, hierarchical and federated [25])
Examples	n8n, Flowise, Rivet, Make and Amazon Bedrock	LangChain, LangGraph, CrewAI, SmolAgents, PydanticAI, Agents SDK, AutoGen, and Magnetic-One

1) HANDOFFS

Handoffs enable one agent to delegate a specific task to another. In the proposed architecture, the Interaction Agent passes responsibility for identifying suitable parking spots to the Informative Search Agent. The latter employs a web-search tool to retrieve candidate facilities that satisfy the driver's stated requirements and preferences—location, price, walking distance, EV charging, accessibility, and review rating—and returns a ranked list to the Interaction Agent, as illustrated in Figure 7. The Interaction Agent then formats and delivers these recommendations to the driver. For example, for a Toronto Town Hall query it recommends three garages with direct URLs to the booking pages:

1) **Green P Carpark 36 (Nathan Phillips Square)**
110 Queen St W, Toronto
Coordinates: 43.6517° N, 79.3844° W
87 m walk; CA$4.00 per half-hour; EV chargers; wheelchair-accessible; 4.2★.
2) **Parking Town Hall Garage**
361 University Ave, Toronto
Coordinates: 43.6532° N, 79.3859° W
200 m walk; CA$2.50 per half-hour; EV chargers; wheelchair-accessible; 4.8★.
3) **Bell Trinity Square – Lot 235**
483 Bay St, Toronto
Coordinates: 43.6544° N, 79.3828° W
240 m walk; CA$1.76 per half-hour; EV chargers; no dedicated accessibility features listed; 4.0★.

2) GUARDRAILS

In the OpenAI Agent SDK, guardrails can be specified so that both the initial user request and the final agent response are rigorously validated. Within a frictionless-parking scenario, input guardrails enforce clarity and consistency before search, whereas output guardrails ensure that the resulting recommendation remains safe, plausible, constraint-compliant, and ethically sound. Examples of potential input and output guardrails are provided below:

- **Input guardrails:** The destination provided in a request is first disambiguated. When an expression such as "the park" is received, additional context is solicited until a single, precise geolocation is obtained; requests that lie outside the system's operational region are declined with an explanatory notice. Declared preferences are then normalized. Statements about acceptable walking distance are converted to numeric values in a standard unit, either meters or minutes, and a default value of 500 m is applied when no explicit value is present. Price sensitivity is converted to an internal tier structure. Terms such as gratis or no cost are assigned to Tier 0, low or affordable to Tier 1, and so forth. Because the mapping is region-sensitive, a description considered "cheap" in a small town is not conflated with the same description in a major downtown core. Conflicting constraints, for example a request for a free space inside a premium underground garage, trigger a clarification cycle instead of silently rejecting it. Vehicle–facility compatibility is verified next. Requests associated with electric vehicles must be matched to connectors of the correct type, and over-sized vehicles are filtered out of compact-only facilities.

- **Output guardrails:** After search, each candidate parking space is subjected to a second validation stage. Legal permissibility, non-duplication, and the absence of temporary bans caused by construction, special events, or snow removal are confirmed. Safety and accessibility flags are honored by prioritizing well-lit areas, curb cuts, and designated disabled-access provisions whenever such metadata are available. Every recommendation can be accompanied by concise explanatory text such as, "The space lies 50 m from the destination, was reported vacant within the last two minutes, and is equipped with Type 2 AC charging." When confidence is limited, one or two alternative locations are appended along with numerical confidence scores so that an informed decision can be made. An ethics module then checks for systematic bias toward particular neighborhoods or price tiers, and individual-level location data are withheld from third parties unless explicit consent has been granted.

B. EXPERIMENTAL CONFIGURATION

The experimental configuration used in the study are summarized as follows.

1) USER PROFILES

- **commuter_saver**: Drives an EV on a tight budget. Prioritizes *lowest half-hour rate* and a *working charger*, even if the walk is ten minutes. Amenities, ratings, and ambiance are irrelevant.
- **efficient_multitasker**: Values time over money. Seeks the *closest, top-reviewed* space—ideally valet or tap-and-go—within a block of a downtown meeting. Will pay a premium; chargers and accessibility are optional.
- **creative_wanderer**: Wants a memorable, off-beat location in an artsy, less-touristy area. Prefers quirky reviews and atmosphere over price or distance; EV charging is *not* required.
- **independent_elder**: Uses a wheelchair and avoids crowds. Requires a ground-level or lift-equipped space with wide bays *directly* next to the venue entrance. Short, obstacle-free access is paramount; price and charging are secondary.
- **green_professional**: Business traveler in an electric company car. Desires a centrally located garage with a *reliable, well-reviewed fast charger* and good lighting. Accepts mid-range pricing; accessibility is optional, charger up-time is critical.

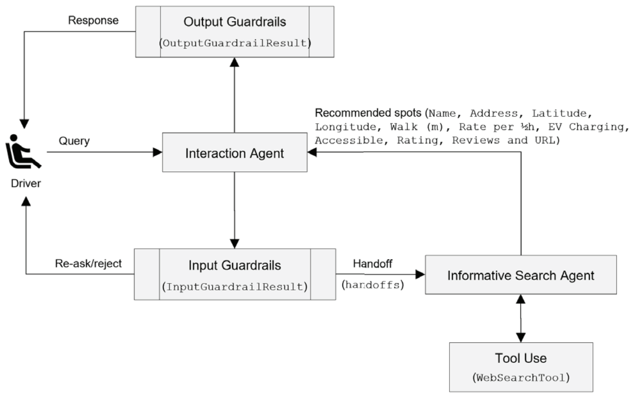

FIGURE 7. Implemented agents, guardrails, handoff and tool use.

2) LLM MODELS

Two categories of large language models are employed:

- **GPT models**: `gpt-4.1` and `gpt-4.1-mini`.
- **Reasoning models**: `gpt-4o` and `gpt-4o-mini`.

3) ENTROPY SETTINGS

Entropy controls the stochasticity of token sampling through the temperature T and nucleus sampling ($top-p$, henceforth p) parameters, while the maximum output length max_tokens is held constant. Low values of T and p produce nearly deterministic text; high values encourage creativity and lexical diversity. Three entropy conditions are evaluated:

$$\text{entropy\_low} : (T, p, max\_tokens) = (0.2,\ 0.8,\ 150)$$
$$\text{entropy\_medium} : (0.7,\ 0.9,\ 150)$$
$$\text{entropy\_high} : (1.0,\ 1.0,\ 150)$$

4) VERBOSITY SETTINGS

Verbosity isolates response length by varying the token cap max_tokens, while holding stochasticity at the balanced setting $T = 0.7, p = 0.9$. This allows us to measure timing and consistency effects due solely to answer length:

$$\text{length\_short} : (T, p, max\_tokens) = (0.7,\ 0.9,\ 100)$$
$$\text{length\_medium} : (0.7,\ 0.9,\ 150)$$
$$\text{length\_long} : (0.7,\ 0.9,\ 200)$$

In summary, the *entropy sets* manipulate randomness at a fixed length limit, whereas the *verbosity sets* manipulate length at a fixed level of randomness.

5) QUERY COMPLEXITY

Query complexity reflects how many explicit constraints the user specifies.

- **simple**: *"Cheap parking near Toronto City Hall."* This *simple* query mentions only one landmark and a single intent (e.g., cost).

- **moderate**: *"Locate a well-rated, budget-friendly garage within 500m of Toronto Eaton Centre."* This *moderate* query introduces additional numeric or qualitative filters such as distance, price, or rating.
- **complex**: *"Identify a garage in Toronto's Financial District that is $\leq$250m walk, offers at least two Level-2 EV chargers, is step-free accessible, costs $\leq$ \$4 per half-hour, and is rated 4.5★ or higher."* This *complex* query combines several hard constraints simultaneously (e.g., distance, EV charging, accessibility, price ceiling, rating threshold), requiring multi-criteria optimization by the assistant.

6) QUERY SPECIFICITY

Query specificity captures the informational detail with which the user articulates the request.

- **high_specificity**: *"Locate an accessible garage with at least two EV chargers, within 300m of Toronto City Hall, costing $\leq$ \$5 per half-hour and rated $\geq$4.0★."* This *high-specificity* prompt provides concrete numeric targets (distance, price) and attribute thresholds (rating, EV chargers).
- **medium_specificity**: *"I'd like reasonably priced parking somewhere in downtown Toronto; a charger would be nice."* This *medium-specificity* prompt conveys general preferences in plain language but omit precise numbers.
- **low_specificity**: *"What's the best place to park down there?"* This *low-specificity* prompt is vague or colloquial, supplying minimal detail beyond a broad objective.

C. KEY PERFORMANCE INDICATORS

To the best of our knowledge, no publicly available parking assistant offers an agentic AI baseline of comparable scope. Consequently, the evaluation in this section relies exclusively on real GPT-based calls and live web search data, tested under multiple backbones, entropy levels, and verbosity settings, rather than comparison against an existing reference system. The system employs two complementary safeguards to ensure that every recommendation is plausible and satisfies all user-specified constraints. First, guardrails automatically reject or rewrite responses that violate constraint or omit required fields (e.g., EV charger, wheelchair access). Second, the agent prompts the backbone model with an explicit Chain-of-Thought (CoT) directive, obliging the agent to enumerate each reasoning step—parsing constraints, filtering candidates, scoring, and final selection—so that intermediate results can be internally checked before the answer is returned. This combination of guardrails and CoT reasoning provides a pragmatic, automated test of factual correctness without the need for external ground truth or manual validation.

Two complementary metrics are tracked throughout the experimental grid:

1) LATENCY

The latency or response time is the wall-clock latency (in seconds) between dispatching the user query to the model endpoint and receiving the final token of the assistant's reply. It captures the perceived wait experienced by an end-user and is therefore a primary determinant of conversational usability. All timings were recorded on a notebook equipped with an Intel Core i7-1355U (1.7 GHz base clock, 10-core, 12 logical processors) and 16 GB RAM; no discrete GPU was present.

2) RESPONSE CONSISTENCY

While large language models are probabilistic, a production system should return semantically stable answers to semantically equivalent prompts. Algorithm 1 quantifies the consistency of a language model by averaging the pair-wise textual similarity of its responses across repeated runs. Similarity is obtained with *SequenceMatcher* [26], which locates the longest common subsequences between two strings and converts the amount of shared content into a ratio that ranges from 0 (no overlap) to 1 (identical text).

Algorithm 1 Compute Consistency Score

```
Require: $O$ list of strings, $r$ number of runs
Ensure: $c$ consistency score in [0, 1]
 1: $L \leftarrow$ nonempty elements of $O$
 2: if $|L| < 2$ or $r < 2$ then
 3:     return 1.0
 4: end if
 5: $S \leftarrow \emptyset$
 6: for $i = 1$ to $|L| - 1$ do
 7:     for $j = i + 1$ to $|L|$ do
 8:         $s \leftarrow$ SequenceMatcher($L_i$, $L_j$).ratio()
 9:         $S \leftarrow S \cup \{s\}$
10:     end for
11: end for
12: $c \leftarrow \frac{1}{|S|} \sum_{s \in S} s$
13: return $c$
```

By evaluating every unordered pair of non-empty outputs, the algorithm yields a single consistency score whose upper bound corresponds to perfectly reproducible answers, while lower values indicate increasing variability in the model's response.

D. EXPERIMENTAL ANALYSIS

We adopted a full-factorial scheme in which every level of each independent variable is crossed with every level of the remaining variables. Six factors were considered:

- **User profile** (5 levels): commuter saver, efficient multitasker, creative wanderer, independent elder, green professional.
- **LLM backbone** (4 levels): `gpt-4.1`, `gpt-4.1-mini`, `gpt-4o`, `gpt-4o-mini`.
- **Entropy setting** (3 levels): low, medium, high.

- **Verbosity cap** (3 levels): short, medium, long.
- **Query complexity** (3 levels): simple, moderate, complex.
- **Query specificity** (3 levels): high, medium, low.

The total number of unique parameter combinations is therefore 5 profiles $\times$ 4 models $\times$ 3 entropy levels $\times$ 3 verbosity levels $\times$ 3 complexities $\times$ 3 specificities $=$ $1,620$ distinct conditions. Each condition is evaluated twice (two independent calls to the model) to permit estimation of response variability, yielding $1,620$ conditions $\times$ 2 runs $=$ $3,240$ model invocations. This exhaustive design enables unbiased main-effect and interaction estimation across user intent, model choice, decoding entropy, verbosity, and prompt formulation.

In our analysis, we first examine the joint distribution of response latency and lexical consistency to identify key performance patterns, then we employ robust GLM fixed-effects models to quantify each factor's additive impact on these metrics, and conclude with a SHAP-based feature-importance analysis to rank the strongest global drivers of latency and consistency.

1) LATENCY AND CONSISTENCY

Figure 8 visualizes the joint distribution of response time (latency) and lexical stability (consistency) across all experimental factors. Three overarching patterns emerge.

All profiles cluster in the four–second region, yet their linguistic stability diverges markedly: *commuter_saver* exhibits the highest consistency ($\approx$ 0.75), while *creative_wanderer* falls below 0.50. The latter profile combines EV preference with a request for non-touristic areas, apparently provoking more variable generation. Among the four backbone models, `gpt-4o-mini` attains the lowest latency (3.7s) at an intermediate consistency of 0.66, outperforming the larger `gpt-4.1` (latency 4.8s, consistency 0.62). Hence, increasing parameter count beyond the mini variant yields diminishing returns for both speed and stability in this task.

Low-entropy and short-verbosity settings jointly minimize latency (3.7–4.2s) and maximize consistency (0.66–0.70). High entropy or long responses increase latency by roughly one second and reduce consistency by 8–12 %. A moderate prompt specificity is optimal: it raises consistency to 0.71 without incurring the latency penalty observed for highly specific requests. Task complexity shows a similar U-shaped profile, with *moderate* queries being both the quickest (3.95s) and the most consistent (0.66).

Overall, Figure 8 indicates that the assistant's speed–consistency frontier is driven more by decoding settings and prompt formulation than by sheer model size. Practitioners can therefore obtain near-optimal performance by employing the `gpt-4o-mini` backbone, constraining entropy and verbosity, and expressing requests at a moderate level of specificity and complexity.

TABLE 4. Results of the non-parametric Kruskal–Wallis *H* test for each experimental factor. Boldface indicates $p < .05$.

Factor	Latency (s)		Consistency	
	H	p	H	p
profile	29.515	0.000	248.775	0.000
model	309.482	0.000	28.692	0.000
entropy	1.630	0.443	13.755	0.001
verbosity	254.753	0.000	69.197	0.000
complexity	185.045	0.000	17.797	0.000
specificity	119.926	0.000	64.388	0.000

TABLE 5. Robust GLM coefficient estimates relative to the reference condition. Negative values indicate faster replies (delta latency less than 0) or more stable wording (delta consistency greater than 0). Significance levels are marked as follows: ***p < 0.001, **p < 0.01, *p < 0.05.

Level	ΔLatency (s)	ΔConsistency
Profiles		
creative_wanderer	0.172**	−0.277***
efficient_multitasker	0.297***	−0.065***
green_professional	0.206**	−0.100***
independent_elder	0.347***	−0.123***
Model		
gpt-4.1	1.139***	−0.035*
gpt-4.1-mini	0.507***	0.018**
gpt-4o	0.492***	−0.060***
Entropy		
entropy_medium	−0.032	−0.032*
entropy_high	0.033	−0.056***
Verbosity		
length_medium	0.521***	−0.078*
length_long	0.918***	−0.105***
Complexity		
simple	0.074***	−0.009
complex	0.689***	−0.055***
Specificity		
low_specificity	−0.081***	−0.106***
high_specificity	0.498***	−0.106***

2) FACTOR ANALYSIS WITH THE KRUSKAL–WALLIS TEST

We applied the non-parametric Kruskal–Wallis H test [27] to determine whether the median latency or response consistency differs across the levels of each experimental factor. Table 4 summarizes the resulting H statistics and p-values.

For latency, the omnibus effect of *model* is by far the strongest ($H = 309.48,\ p = 0.000$), followed by *verbosity*, *complexity*, and *specificity*. Decoding *entropy* alone does not significantly influence latency ($p = 0.443$), indicating that randomness in token sampling is not a time bottleneck.

Conversely, the largest heterogeneity in consistency arises from *profile* ($H = 248.78,\ p = 0.000$), confirming that the linguistic stability of the assistant's answers is highly sensitive to the type of user request, whereas *entropy*, although negligible for timing, still affects lexical reproducibility ($H = 13.76,\ p = 0.001$). These findings reinforce the design guidance that (i) selecting the `gpt-4o-mini` backbone and limiting verbosity yield the greatest reductions in waiting time, while (ii) moderate prompt complexity and specificity offer the best trade-off between speed and consistency, and (iii) entropy can be tuned primarily for stylistic diversity without incurring a latency penalty.

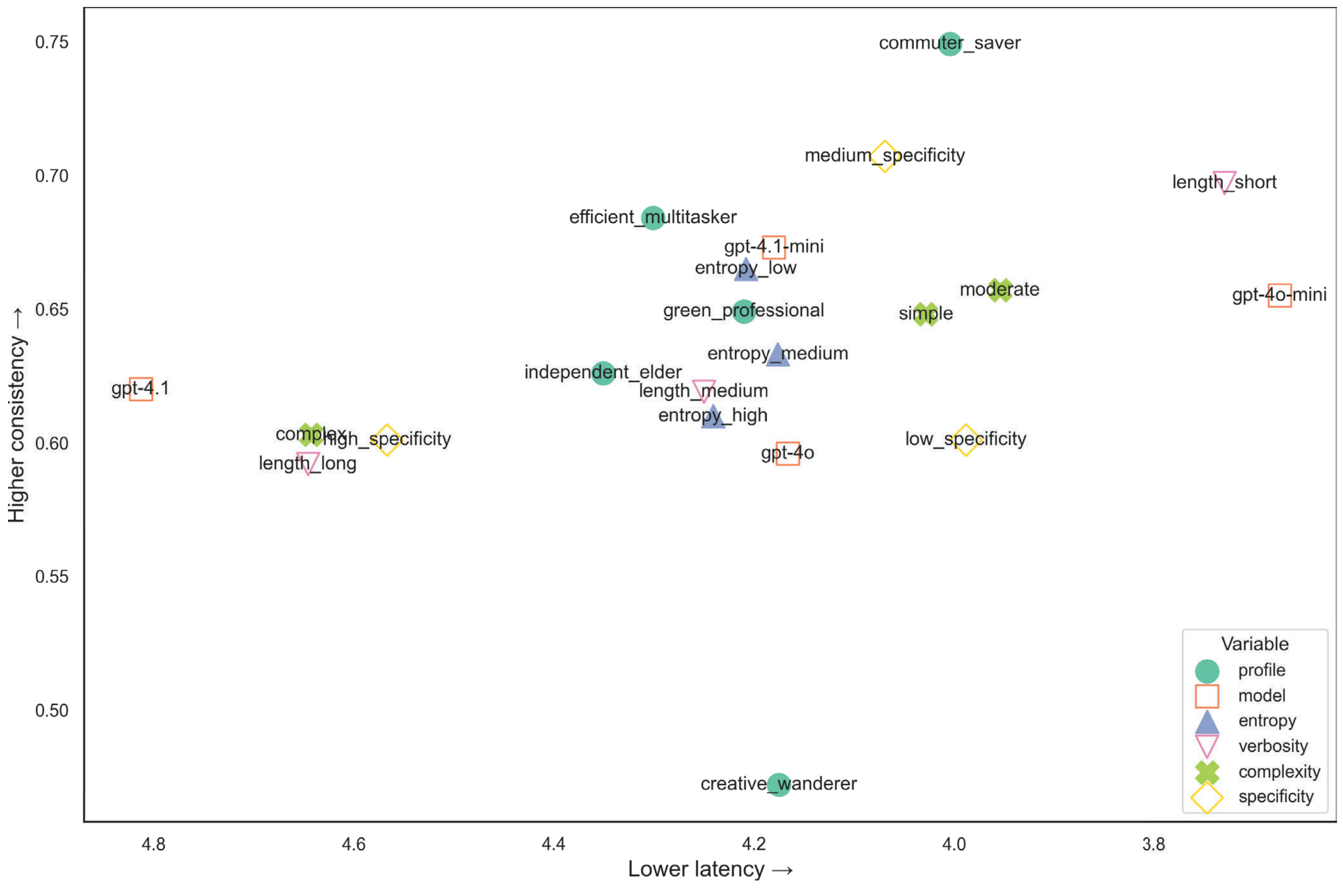

FIGURE 8. Average latency and consistency.

3) ROBUST GLM FIXED EFFECTS

Generalized linear models (GLMs) [28] extend ordinary least-squares regression to any member of the exponential family and relate the mean response to a linear predictor through a link function, providing a flexible framework for modeling non-Gaussian outcomes. In this analysis, a GLM is used to quantify how experimental factors—user profile, model backbone, entropy, verbosity, query complexity, and prompt specificity—jointly influence two key performance metrics: the agent's latency or response time and response consistency. We adopt an identity link and Gaussian error family so that each coefficient represents the expected *additive* change in latency (seconds) or consistency (absolute units) when a factor level is substituted for the reference condition, holding all other factors constant.

In Table 5, coefficients represent the deviation from the reference condition (Commuter saver, gpt-4o-mini, low entropy, short verbosity, moderate complexity, medium specificity).

Moving to the larger `gpt-4.1` backbone adds 1.14s and reduces consistency by 0.06, while even the lighter `gpt-4.1-mini` incurs a 0.51s penalty for only a marginal +0.02 improvement in consistency; hence `gpt-4o-mini` remains the latency–consistency front-runner.

Increasing verbosity from short to medium or long slows replies by 0.52–0.92s and depresses consistency by up to 0.11, confirming that concise output is still optimal. Among profiles, *creative_wanderer* and *independent_elder* remain the most taxing, adding 0.17–0.35s and sharply lowering consistency (−0.28 and −0.12, respectively). Entropy tweaks have negligible cost in speed but high entropy further erodes consistency (−0.06). Complex queries introduce a sizeable 0.69s delay and cut consistency by 0.06, whereas simple queries barely affect either KPI. Finally, lowering specificity saves a modest 0.08s but at the expense of a notable −0.11 drop in consistency, reinforcing medium specificity as the best overall trade-off.

4) SHAP–BASED FEATURE IMPORTANCE

SHAP-based feature importance [29] is a method to evaluate the contribution of each feature to a machine learning model's predictions using SHAP (SHapley Additive exPlanations) values. These values calculates how much each feature contributes to the difference between the actual prediction and the average prediction across the dataset. It considers all possible combinations of features to ensure a fair attribution. Features with higher average SHAP values have a greater impact on the model's output.

Figure 9 visualizes the mean absolute SHAP values obtained from the Gradient-Boosting model ordered by this joint score. These values quantify how strongly each factor influences the model's prediction of latency or response consistency. Higher values indicate a larger contribution to the variance in latency or consistency.

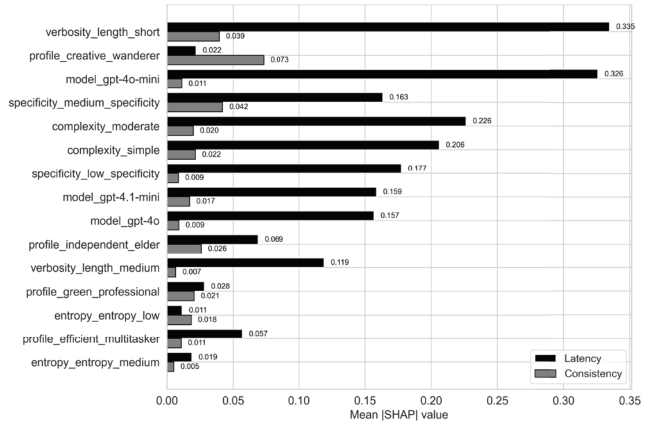

FIGURE 9. SHAP feature importance ordered by joint importance score.

The two most influential determinants of latency are *output length* and *model size*. Switching to `length_short` responses or deploying the lightweight `gpt-4o-mini` backbone each explains roughly one-third of the variation in response time, greatly outweighing any single user profile. Prompt-side factors—specifically moderate or simple query complexity and lower specificity—collectively account for another 40 % of importance mass, indicating that concise, moderately complex requests are answered fastest. Profiles contribute comparatively little: only the *independent_elder* user type enters the top ten drivers, and entropy settings are negligible. Overall, latency is governed primarily by system parameters that designers can control (model, verbosity) rather than by user traits or decoding randomness, suggesting clear optimization levers for real-time parking guidance.

Unlike latency, consistency is driven chiefly by user-profile and prompt formulation. The *creative_wanderer* profile is the single most influential factor, lowering stability by more than twice the effect size of any system parameter. Medium prompt specificity and short outputs both increase lexical similarity, suggesting that moderately detailed instructions and concise responses yield more reproducible answers. Simpler or moderately complex queries improve consistency, whereas entropy—particularly the *entropy_low* setting—has a modest positive effect, aligning with traditional views that lower randomness reduces variability. Model choice still matters, but its influence is secondary to content factors: the lightweight `gpt-4o-mini` improves consistency only marginally compared with `gpt-4.1-mini`.

TABLE 6. Joint SHAP importance (Equation (8)) ranked in descending order.

Feature	Joint score
verbosity_length_short	1.119
profile_creative_wanderer	1.001
model_gpt-4o-mini	0.977
specificity_medium_specificity	0.717
complexity_moderate	0.699
complexity_simple	0.648
specificity_low_specificity	0.516
model_gpt-4.1-mini	0.488
model_gpt-4o	0.453
profile_independent_elder	0.351
verbosity_length_medium	0.333
profile_green_professional	0.231
entropy_entropy_low	0.193
profile_efficient_multitasker	0.163
entropy_entropy_medium	0.023
verbosity_length_long	0.000
model_gpt4.1	0.000
complexity_complex	0.000
specificity_high_specificity	0.000
profile_commuter_saver	0.000
entropy_entropy_high	0.000

To obtain a reference–invariant ranking of how each feature i affects both latency and response consistency, we compute a composite *joint score*. The procedure involves three steps:

$$\tilde{I}_{\text{lat},i} = \frac{I_{\text{lat},i} - \min_j I_{\text{lat},j}}{\max_j I_{\text{lat},j} - \min_j I_{\text{lat},j}} \tag{6}$$

$$\tilde{I}_{\text{con},i} = \frac{I_{\text{con},i} - \min_j I_{\text{con},j}}{\max_j I_{\text{con},j} - \min_j I_{\text{con},j}} \tag{7}$$

$$\text{JointScore}_i = \sqrt{\tilde{I}^2_{\text{lat},i} + \tilde{I}^2_{\text{con},i}} \qquad 0 \le \text{JointScore}_i \le \sqrt{2}\,, \tag{8}$$

where $I_{\text{lat},i}$ and $I_{\text{con},i}$ are the mean absolute SHAP values for latency and consistency, respectively. Equations (6) and (7) rescale each metric to [0, 1]; Equation (8) then combines the two on equal footing. Table 6 shows joint SHAP importance score for each feature.

The top-ranked factor, short-verbosity, cuts latency by roughly one third without harming consistency, recommending concise default answers. The *creative_wanderer* profile is the greatest destabilizer of wording; templated or guided replies should be introduced for this cohort. Employing the lightweight `gpt-4o-mini` backbone delivers a latency benefit comparable to verbosity control with negligible stability loss, making it the preferred production model. Medium prompt specificity and moderate (or simple) complexity jointly boost both KPIs, so interfaces should steer users toward moderately detailed yet straightforward queries. Entropy settings and the remaining profiles have far lower joint scores and can be deprioritized in optimization efforts. Overall, these SHAP results indicate that to enhance answer consistency, the assistant should prioritize controlled user profiles, medium specificity, and brevity, while model architecture and entropy tuning play supporting roles. These findings are fully consonant with the earlier Kruskal–Wallis and GLM findings.

VI. CONCLUSION

This paper presented an envisioned end-to-end agentic AI architecture for frictionless urban parking, demonstrating how AI agents equipped with LLMs, memory systems, and external toolkits can collaboratively solve complex, real-time urban mobility challenges. The architecture comprises a suite of specialized agents, including search, routing, access, on-spot monitoring, and departure agents, coordinated by an Interaction Agent to deliver a seamless user experience along the entire parking journey. By formalizing cooperation patterns such as augmentative, integrative, and debative strategies, along with cueing and handoff inter-agent interaction mechanisms, the system can support flexible and dynamic agent collaboration. Despite growing interest in agentic AI, existing frameworks, particularly no-code and low-code platforms lack native support for these cooperation patterns and inter-agent interaction mechanisms.

The Interaction Agent and the Informative Search Agent with handoff interaction mechanism and input and output guardrails are implemented and evaluated in this paper. To evaluate the design, a six-factor full-factorial experiment crossed five user profiles, four GPT backbones, three entropy levels, three verbosity caps, three query-complexity levels and three specificity levels, yielding a comprehensive view of how architectural, decoding and prompt variables interact. The results converge with non-parametric Kruskal–Wallis test, robust GLM fixed effects, and SHAP analysis: model scale and verbosity exert the greatest influence on response time, with the lightweight "gpt-4o-mini" backbone and concise short-verbosity cap providing the fastest, most stable baseline. Decoding entropy has no measurable latency cost but degrades lexical consistency at higher settings, confirming that entropy primarily influences stylistic expression rather than core performance. User intent is the chief determinant of reproducibility, as reflected by the marked instability of "Creative Wanderer" and "Independent Elder" prompts. Medium prompt specificity and moderate (or simple) complexity strike the best balance between speed and consistency. Collectively, these findings endorse a design strategy that defaults to the mini backbone, limits verbosity, scaffolds prompts toward medium specificity, and adjusts entropy only when diversity is desired, thereby offering a reproducible blueprint for future agentic-AI deployments beyond the parking domain.

Future work will explore several complementary directions. First, a broader set of foundational reasoning models will be benchmarked in place of the current GPT-4 variants, including OpenAI's o1 and o3 series, DeepSeek-R1, Anthropic's Claude 3.7 Sonnet Thinking and Google's Gemini Thinking, to evaluate trade-offs in reasoning depth, latency and cost. Second, diverse memory subsystems—semantic, episodic and procedural—will be integrated to support both short-term context retention and long-term personalization across repeated parking journeys. Third, additional specialist agents (on-route guidance, gate access, micro-routing, on-spot monitoring and departure coordination) will be implemented and inter-agent interaction mechanisms, including cueing and handoff, and various cooperation patterns will be evaluated. Together, these enhancements aim to further improve end-to-end responsiveness, reliability and user-centric adaptability in complex urban mobility scenarios.

REFERENCES

[1] E. Miehling, K. Natesan Ramamurthy, K. R. Varshney, M. Riemer, D. Bouneffouf, J. T. Richards, A. Dhurandhar, E. M. Daly, M. Hind, P. Sattigeri, D. Wei, A. Rawat, J. Gajcin, and W. Geyer, "Agentic AI needs a systems theory," 2025, *arXiv:2503.00237*.

[2] E. Pounds. (2024). *What is Agentic AI?*. Accessed: Jul. 15, 2024. [Online]. Available: https://blogs.nvidia.com/blog/what-is-agentic-ai/

[3] D. B. Acharya, K. Kuppan, and B. Divya, "Agentic AI: Autonomous intelligence for complex Goals—A comprehensive survey," *IEEE Access*, vol. 13, pp. 18912–18936, 2025.

[4] LangChain. (2023). *Vodafone Uses LangChain To Build Production-grade LLM Applications*. Accessed: Jul. 15, 2024. [Online]. Available: https://blog.langchain.dev/customers-vodafone/

[5] Y. Xiao, G. Shi, and P. Zhang, "Towards agentic AI networking in 6G: A generative foundation Model-as-Agent approach," 2025, *arXiv:2503.15764*.

[6] L. N. Mishra and B. Senapati, "Retail resilience engine: An agentic AI framework for building reliable retail systems with test-driven development approach," *IEEE Access*, vol. 13, pp. 50226–50243, 2025.

[7] N. Kshetri, "From predictive and generative to agentic AI: Shaping the future of marketing operations and strategies," *Computer*, vol. 58, no. 4, pp. 121–129, Apr. 2025.

[8] United Nations, Dept. Econ. Social Affairs, Population Division. (2018). *World Urbanization Prospects: The 2018 Revision*. [Online]. Available: https://population.un.org/wup/

[9] *The Future of European Parking: Demand for Electric Cars and Free Parking Spots*, Stockholm, Sweden, 2023.

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[11] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, Jun. 1995.

[12] A. Nareyek, *Constraint-based Agents: An Architecture for Constraint-based Modeling and Local-search-based Reasoning for Planning and Scheduling in Open and Dynamic Worlds*, vol. 2062. Cham, Switzerland: Springer, 2003.

[13] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multiagent systems: The Gaia methodology," *ACM Trans. Softw. Eng. Methodology*, vol. 12, no. 3, pp. 317–370, Jul. 2003.

[14] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, "JADE: A software framework for developing multi-agent applications. Lessons learned," *Inf. Softw. Technol.*, vol. 50, nos. 1–2, pp. 10–21, Jan. 2008.

[15] M. P. Singh and M. N. Huhns, *Service-oriented Computing: Semantics, Processes, Agents*. Hoboken, NJ, USA: Wiley, 2005.

[16] P. Wayner, *Agents Unleashed: A Public Domain Look At Agent Technology*. New York, NY, USA: Academic, 2014.

[17] P. Shojaee, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, and M. Farajtabar, "The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity," 2025, *arXiv:2506.06941*.

[18] J. Wiesinger, P. Marlow, and V. Vuskovic, "Agents," Google, Mountain View, CA, USA, Tech. Rep., 2024. [Online]. Available: https://ppc.land/content/files/2025/01/Newwhitepaper_Agents2.pdf

[19] LangChain. (2024). *State of AI Agents*. [Online]. Available: https://www.langchain.com/stateofaiagents

[20] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, Jan. 2022, pp. 1–7.
[21] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-Thought prompting elicits reasoning in large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 24824–24837.
[22] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2023, pp. 11809–11822.
[23] K. Schmidt, "Cooperative work: A conceptual framework," in *Distributed Decision Making: Cognitive Models for Cooperative Work*, J. Rasmussen, B. Brehmer, and J. Leplat, Eds., Chichester, U.K.: Wiley, Jan. 1991, pp. 75–110.
[24] OpenAI. (2025). *OpenAI Agents SDK*. [Online]. Available: https://openai.github.io/openai-agents-python/
[25] M. Finio and A. Downie. (2024). *What is AI Agent Orchestration?*. [Online]. Available: https://www.ibm.com/think/topics/ai-agent-orchestration
[26] P. S. Found. (2025). *Difflib—Helpers for Computing Deltas*. Accessed: May 26, 2025. [Online]. Available: https://docs.python.org/3/library/difflib.html
[27] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *J. Amer. Stat. Assoc.*, vol. 47, no. 260, pp. 583–621, Dec. 1952.
[28] P. McCullagh and J. A. Nelder, *Generalized Linear Models*. London, U.K.: Chapman & Hall, 1989.
[29] S. Lundberg and S. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2017, pp. 4768–4777.

ALAA KHAMIS (Senior Member, IEEE) is currently an Associate Professor and the Director of the AI for Smart Mobility Laboratory, Department of Industrial and Systems Engineering, and the Interdisciplinary Research Center for Smart Mobility and Logistics, King Fahd University of Petroleum and Minerals (KFUPM). Prior to joining KFUPM, he was the AI and Smart Mobility Technical Leader of General Motors Canada. He is also an Adjunct Faculty Member with the University of Toronto and an Adjunct Professor with Ontario Tech University. He has authored four books and over 190 scientific papers published in refereed journals and international conferences. He holds 72 U.S. patents, trade secrets, and defensive publications. He is the author of *Smart Mobility: Exploring Foundational Technologies and Wider Impacts* and *Optimization Algorithms: AI Techniques for Design, Planning, and Control Problems*. His research interests include the intersection of AI and mobility systems, services, and business models, addressing challenges, such as seamless integrated mobility, observability in software-defined vehicles (SDVs), combining reasoning and action, contextual causal inference, and optimization in people mobility, logistics, and transportation infrastructure. He was a recipient of the 2022–2024 GM Critical Talent Award, the First Place in the 2025 Sustainable Solutions for Pilgrims Challenge, the Best Paper Award in the Smart Mobility Governance Track from the 2023 IEEE International Conference on Smart Mobility, and the 2018 IEEE Member and Geographic Activities (MGA) Achievement Award.