



# Voice Assistant Booking Agent for Drivers – Technical Architecture

## Executive Summary

This document describes the design of an AI-powered voice agent that enables drivers to book restaurant and service reservations **completely hands-free**. Building on the existing AI Voice Booking Agent, this design tailors the architecture for in-vehicle use: it integrates with car or smartphone voice assistants, obeys hands-free driving laws and minimizes cognitive load. Voice assistants are legal and encouraged for drivers because they allow calls and messages without handling a phone <sup>1</sup>; a well-designed system should respond quickly, navigate smoothly between tasks, provide hands-free alerts and never require manual input <sup>2</sup>. Moreover, research shows that in-vehicle voice assistants can improve driver alertness and reduce fatigue in partially automated vehicles <sup>3</sup>. The proposed architecture leverages these insights to create a safe and reliable booking experience for drivers.

---

## 1. Customer Scenario

### Primary Job

**“As a driver commuting home, I want to ask my voice assistant to book a restaurant and confirm the reservation without taking my hands off the steering wheel, so that I stay safe and never miss a table.”**

### Why Voice Interaction in Cars Is Essential

- 1. Legal requirements.** Many states ban drivers from manually using cell phones unless parked, but hands-free technology is permitted <sup>4</sup>. Voice assistants allow drivers to call, text and retrieve information while keeping both hands on the wheel <sup>1</sup>.
- 2. Safety & alertness.** A study on digital voice assistants in partially automated vehicles found that voice-based conversation reduced driver sleepiness (lower Karolinska Sleepiness Scale scores), increased attentive glances and shortened the time to regain motor readiness <sup>3</sup>. This indicates that a well-designed voice assistant can counter passive fatigue.
- 3. Cognitive load.** To prevent distraction, a voice assistant should respond quickly, switch between functions smoothly, deliver concise prompts and avoid requiring manual intervention <sup>2</sup>. In a car, lengthy dialogues or complex multi-step prompts could dangerously divert attention.

### Example Use Cases

- 1. On-the-go dinner booking:** While driving home, the user says, “Hey assistant, book dinner for two at a sushi place near Cambridge at 8 pm.” The agent checks the driver’s calendar, finds an available time, calls the restaurant and confirms the booking, then reads back the confirmation number.

2. **Spontaneous brunch reservation:** En route to errands on Saturday morning, the user asks for a brunch reservation nearby. The agent suggests a suitable café within 15 minutes of the current location, checks for a table and confirms or offers alternatives.

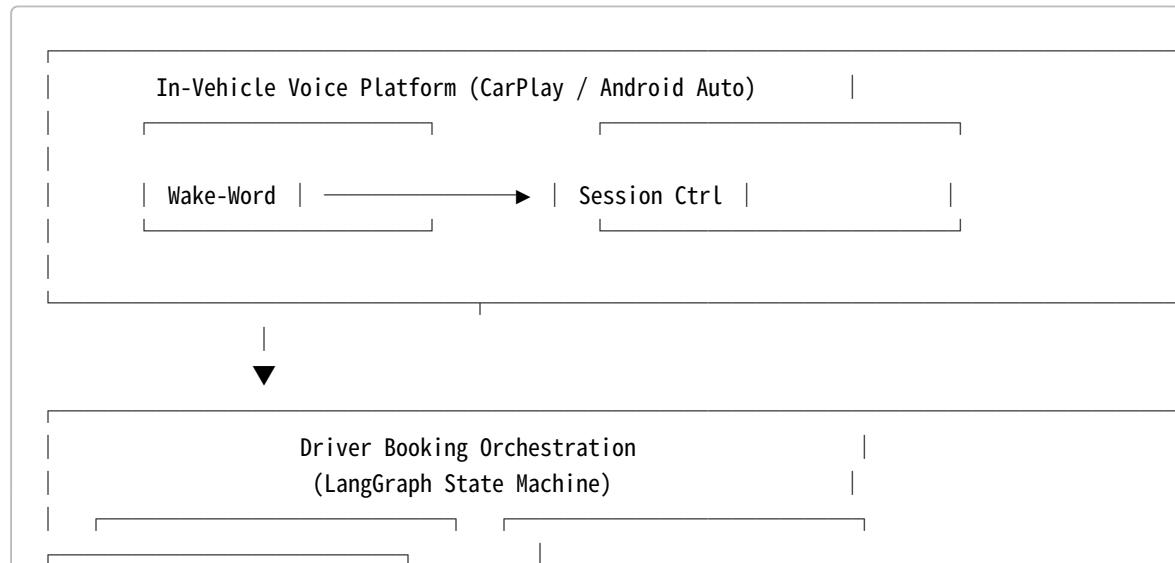
3. **Voice-only follow-up:** If a restaurant is fully booked, the agent offers the next available time or suggests a similar restaurant, summarizing the options succinctly.

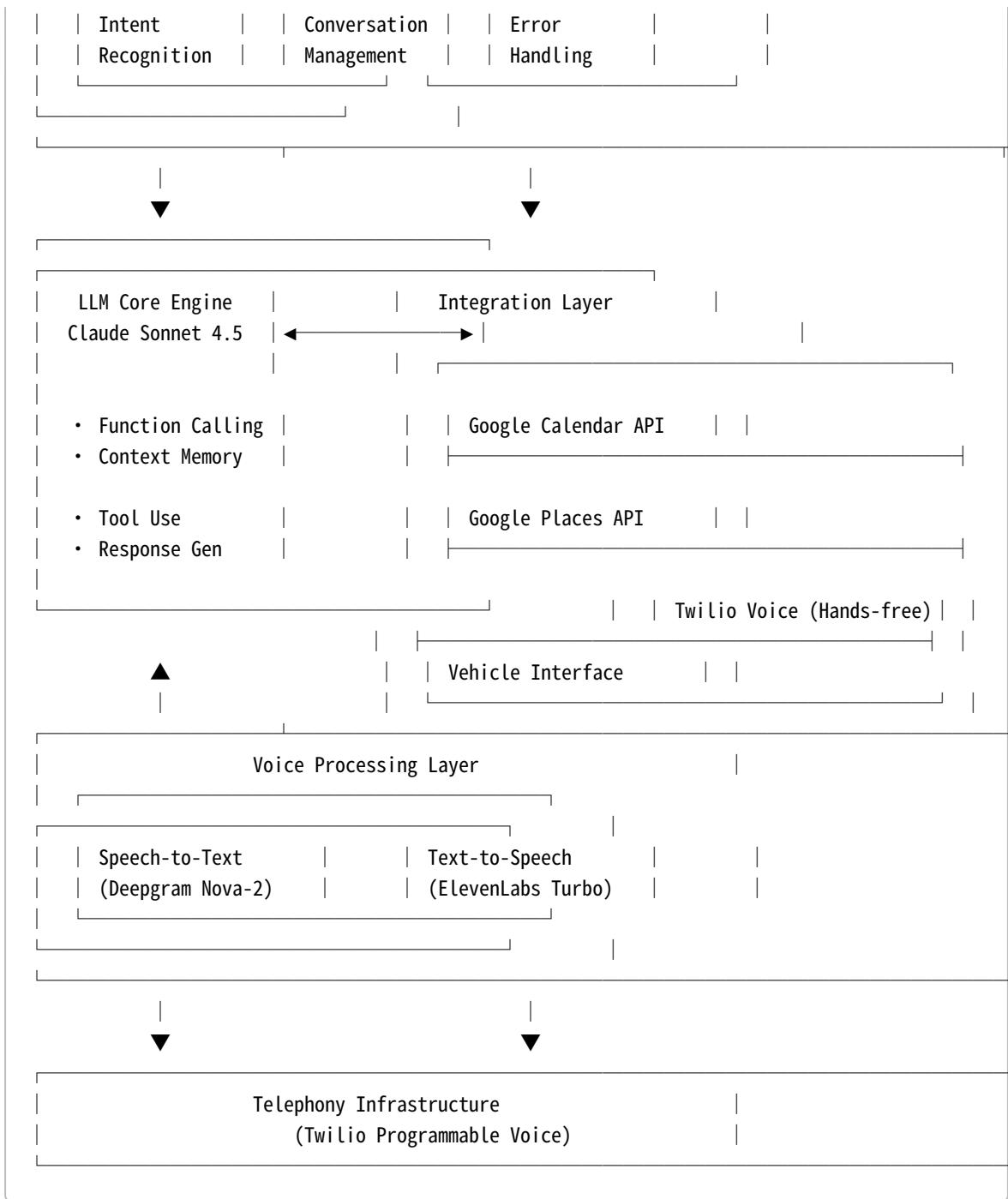
---

## 2. Design Principles

1. **Hands-free by default.** All user interactions are voice-activated. The agent must never require the driver to look at or touch the phone during the booking process.
  2. **Low-latency responses.** The assistant should respond within ~1 s to user commands; long delays may cause the driver to look at the screen and thus break hands-free compliance <sup>2</sup>.
  3. **Concise speech.** To minimize cognitive load, responses are short (<15 s) and focus on critical details. When reading back information, the agent only states the necessary data (date, time, party size, confirmation number).
  4. **Safety-aware timing.** The system monitors driving context (speed, manoeuvres) through the host platform (e.g., Apple CarPlay) and can delay non-urgent prompts until safe conditions (straight roads, low traffic).
  5. **Robust speech handling.** Use high-quality STT and TTS models optimized for telephony and in-car noise. Provide custom vocabulary for restaurant names and addresses.
  6. **Graceful degradation.** If network connectivity or STT/TTS services fail, the agent should fall back to simpler interactions (e.g., sending a text reminder to book later) rather than leaving the driver waiting.
- 

## 3. High-Level Architecture





## Key Differences from General POC

- 1. In-vehicle interface:** The entry point is a wake-word and session controller provided by CarPlay/Android Auto rather than a web/mobile UI. This ensures the system is always listening when the driver speaks the wake-word and gracefully ends the session after confirmation.
- 2. Vehicle context data:** The system can receive basic driving context (speed, lane changes) via the vehicle interface, enabling it to defer non-critical prompts until safe. For example, if the driver is negotiating a busy intersection, the agent will wait to deliver the reservation summary.

- 
3. **Shorter timeouts:** To minimize distraction, the conversation loop uses shorter silence timeouts (e.g., 5 seconds) and maximum call durations (e.g., 2 minutes). If the call cannot be completed within this window, the agent offers to retry later.

## 4. Technology Stack & Justifications

### 4.1 Large Language Model (LLM)

- **Claude Sonnet 4.5:** The same model used in the general POC, chosen for its long context window, reliability in tool-calling and fast response times. Low latency is especially important to maintain driver attention. An alternative like GPT-4 Turbo may introduce longer response delays (>800 ms) which could distract drivers.

### 4.2 Speech-to-Text (STT)

- **Deepgram Nova-2:** Provides real-time streaming (300–500 ms) and robustness to telephony audio. It can be tuned to handle background noise inside a car cabin and offers a custom vocabulary for restaurant names. Alternatives like Whisper have higher latency (~2 s) and may cause unacceptable delays.

### 4.3 Text-to-Speech (TTS)

- **ElevenLabs Turbo v2.5:** Produces natural prosody and supports streaming word-by-word output. Low latency (300–500 ms) ensures that responses start promptly. Emotional range allows polite, friendly yet concise speech. In-car usage demands voices that sound clear over road noise.

### 4.4 Vehicle Interface & Wake-Word

- **CarPlay / Android Auto:** These platforms provide the voice invocation framework, microphone access and output channels through the vehicle’s speakers. They also enforce safety guidelines by limiting permitted actions while driving.
- **Wake-word detection:** Either provided by the OS (e.g., “Hey Siri”, “Hey Google”) or through a custom wake-word engine. Once triggered, the session controller routes commands to the booking agent.

### 4.5 Telephony

- **Twilio Programmable Voice:** Used to place outbound calls to restaurants. The agent runs in hands-free mode, streaming audio through Twilio’s WebSocket integration. Calls are automatically muted or summarised when the conversation is not relevant to the driver.

### 4.6 Integrations

1. **Google Calendar API:** Check driver availability and block off confirmed reservations. Use read-only scope to verify free slots and events; write scope to insert confirmed bookings.
2. **Google Places API:** Search for restaurants near the driver’s current location or specified area. Retrieve phone numbers and opening hours.
3. **Vehicle Context API:** Where available, obtain speed and road condition data to time prompts safely.

## 5. Orchestration & Conversation Flow

### 5.1 State Management

The agent uses a **LangGraph** state machine similar to the general POC, with modifications to handle driving constraints. The state schema extends the original `BookingState` with additional fields:

```
class DrivingBookingState(BookingState):
    driving_speed: float           # Current vehicle speed (km/h)
    road_complexity: str            # 'straight', 'curvy', 'intersection', etc.
    last_prompt_time: float         # Timestamp of last spoken prompt
    safe_to_prompt: bool            # Whether it is safe to speak now
```

The orchestrator periodically checks `safe_to_prompt` before speaking. If `safe_to_prompt` is False (e.g., driver is navigating a complex turn), the agent queues messages and plays them when safe.

### 5.2 Workflow Graph Modifications

The high-level nodes remain the same (`parse_intent`, `check_calendar`, `find_business`, `prepare_call`, `make_call`, `converse`, `confirm_booking`, `handle_error`), but with additional logic:

1. **Session start:** The workflow begins upon wake-word detection. It first confirms with the driver's STT that the command is a booking request; if not, it gracefully passes to the platform's default assistant.
2. **Driving context check:** Before each prompt, the node queries `safe_to_prompt`. If False, the node delays the prompt and schedules a callback when the driving context becomes safe.
3. **Concise prompts:** The `prepare_call` node constructs a system prompt instructing the LLM to be succinct and repeat essential details only once. Long explanations are avoided.
4. **Error fallback:** The `route_conversation` function is tuned to minimize back-and-forth. If the restaurant is fully booked, the agent offers just one alternative time; if the driver says "try later," the agent ends the session.
5. **Session termination:** After confirmation, the agent summarises the reservation in  $\leq 15$  s and ends the voice session. It then sends a notification to the driver's phone with full details.

### 5.4 Prototype Implementation

The repository includes a simple demonstration program, `driving_booking_agent.py`, which implements a minimal viable product (MVP) for the driving scenario without external dependencies. This script showcases how the agent parses user requests, invokes a tool function to compute dates, handles ambiguous input and conducts a concise simulated conversation:

```
from driving_booking_agent import DrivingBookingAgent
```

```
agent = DrivingBookingAgent()
agent.run_demo()
```

Key features of the prototype:

- **Tool-calling for date computation:** The `parse_intent` function calls `get_next_weekday()` to translate relative phrases like “next Friday” into a concrete date. This function acts as an external tool and returns the next Friday’s date based on the current date.
- **Mock calendar and business search:** The agent includes `check_calendar()` and `find_business()` functions that simulate availability checking and restaurant lookup without requiring external API keys.
- **Error handling for ambiguous input:** If the user provides an invalid time (e.g., “25 pm”), `handle_ambiguous_time()` detects the error and prompts the user with a short clarification request. This demonstrates how the agent manages conversational ambiguity while driving.
- **Concise conversation loop:** The `converse()` function simulates a brief, polite call with the receptionist and records a confirmation number. At the end of the demo, the agent summarises the booking details to the driver in one sentence, adhering to the constraint of short prompts for in-vehicle use.

Although this prototype uses mocked services, it illustrates how the orchestrator will integrate tool-calling and error handling logic in a production system. Developers can replace these mocked functions with real API calls (e.g., Google Places, Twilio) by injecting the appropriate clients.

## 5.3 Safety-Aware Prompt Scheduling

An example of scheduling prompts based on vehicle context:

```
def speak_when_safe(state: DrivingBookingState, message: str):
    if state.safe_to_prompt:
        tts_output = tts.synthesize(message)
        vehicle_interface.play_audio(tts_output)
        state.last_prompt_time = time.time()
    else:
        # Queue the message and register callback
        state.queued_message = message
        vehicle_interface.on_safe(lambda: speak_when_safe(state, message))
```

This pattern ensures prompts never interrupt the driver during risky manoeuvres.

---

# 6. Evaluation & Testing

## 6.1 Success Metrics

Metric	Target	Measurement
<b>Task completion rate</b>	>85 % of booking requests result in a confirmed reservation	Count confirmed bookings ÷ total attempts

Metric	Target	Measurement
<b>Driver distraction</b>	Average NASA-TLX workload rating $\leq 3$ (on a scale of 1–10) during agent use	Conduct simulator studies with and without agent
<b>Reaction time</b>	Drivers' reaction times (to simulated hazards) do not degrade when using the agent	Measure brake/steering reaction in simulator
<b>Response latency</b>	Agent responses start within 1 s of user command	Log time stamps in STT and TTS pipeline
<b>Prompt length</b>	Spoken prompts $\leq 15$ s and $\leq 30$ words	Enforce length checks in conversation logic

## 6.2 Testing Methodology

1. **Simulator studies:** Use a high-fidelity driving simulator to expose participants to booking tasks while driving. Compare driver workload, reaction time and eye-glance behavior with and without the agent. Validate that the voice assistant does not degrade alertness; studies have shown that voice assistants can reduce driver fatigue and improve alertness <sup>3</sup>.
2. **Unit and integration testing:** Repurpose POC tests for intent parsing, calendar checking and business search. Add tests for wake-word detection, driving context integration and prompt scheduling.
3. **Beta testing:** Deploy the system to a small group of users with CarPlay or Android Auto. Collect feedback on voice clarity, ease of use and perceived safety. Iterate on prompt design.
4. **A/B testing:** Compare different tones (formal vs. friendly) and different prompt lengths to determine which results in lower cognitive load.

---

## 7. Security & Privacy Considerations

1. **Call recording consent:** Inform the driver that calls may be recorded for quality assurance and comply with state call-recording laws. Many states require consent of all parties.
  2. **PII protection:** Encrypt user names, phone numbers and calendars at rest and in transit. Ensure that only essential information is shared with the restaurant.
  3. **Hands-free compliance:** Confirm that the platform uses a mounted phone or built-in display; avoid instructing the driver to pick up the phone <sup>4</sup>.
  4. **Rate limiting & misuse prevention:** Limit the number of calls per user per day to prevent distracted or prank calling from the car.
-

## 8. Future Enhancements

1. **Contextual awareness:** Integrate with advanced driver assistance systems to better assess when it is safe to speak (e.g., using lane-keeping cameras to detect driver gaze).
  2. **Predictive booking:** Use location and calendar data to proactively suggest reservations when the driver is near a dining district and has free time.
  3. **Multi-service concierge:** Expand beyond restaurants to book gas station stops, parking and car maintenance while on the road.
  4. **Language support:** Provide multi-language interaction for drivers who prefer Spanish, Mandarin or other languages.
- 

## 9. Use of AI Coding Assistants

AI coding assistants such as **Claude Code** and **GitHub Copilot** were instrumental in developing the proof-of-concept. They were used to:

1. **Generate boilerplate code** for the LangGraph workflow, state definitions and unit test templates.
2. **Draft integration scaffolds** for Twilio, Deepgram and ElevenLabs, which were then manually refined to ensure correct asynchronous patterns and error handling.
3. **Assist with debugging** tricky asynchronous logic in the conversation loop and ensure concise prompts.
4. **Auto-complete type hints and docstrings** to improve readability.

All AI-generated code was reviewed, modified and documented by a human developer. IDE tools (e.g., VS Code with Python extensions) provided linting, static type checking and debugging. The combination of AI assistants and traditional development tools accelerated the creation of the POC while maintaining code quality and security.

---

## Conclusion

Adapting the AI Voice Booking Agent for drivers requires careful attention to safety, legal compliance and user experience. By integrating with in-vehicle voice platforms, enforcing hands-free operation and designing concise, timely prompts, the agent can make restaurant and service reservations without distracting the driver. Legal research confirms that hands-free voice assistants are permissible and even encouraged because they keep drivers' hands on the wheel <sup>1</sup>, and empirical studies demonstrate that voice assistants can enhance driver alertness <sup>3</sup>. With robust speech processing, thoughtful orchestration and rigorous testing, this architecture can deliver a safe and effective driving companion.

---

<sup>1</sup> <sup>2</sup> <sup>4</sup> Is It Legal to Use a Voice Assistant While Driving?

<https://www.makeuseof.com/is-it-legal-to-use-a-voice-assistant-while-driving/>

<sup>3</sup> Exploring the effectiveness of a digital voice assistant to maintain driver alertness in partially automated vehicles - PubMed

<https://pubmed.ncbi.nlm.nih.gov/33881365/>