

Sunday Technical Whitepaper

Version 2.0 · September 2026

Table Of Contents

[Executive Summary](#)

[1. Terminology](#)

[1.1 Control Plane](#)

[1.2 Agent Runtimes](#)

[1.3 Edge Workers](#)

[1.4 Event Queue](#)

[1.5 KV Store](#)

[1.6 Vector Database](#)

[1.7 Booking Provider Layer](#)

[1.8 Speech Terms \(STT, TTS, VAD, Turn Detection\)](#)

[2. System Architecture Overview](#)

[2.1 High-Level System Architecture](#)

[2.2 Architectural Principles](#)

[3. Voice Agent and Speech Stack](#)

[3.1 How a call reaches the agent](#)

[3.2 How the voice loop works](#)

[3.3 Guided workflows](#)

[3.4 Call transfer](#)

[3.5 Call recording](#)

[4. Email Agent](#)

[4.1 How email reaches the agent](#)

[4.2 Filtering before any model runs](#)

[4.3 Draft-only by design](#)

[4.4 Email workflows](#)

[5. Knowledge Base Ingestion and Retrieval](#)

[5.1 How ingestion and retrieval works](#)

[6. Integrations Architecture](#)

[6.1 Booking Provider Integrations](#)

[6.2 Edge Tool Execution and Isolation Diagram](#)

[6.3 Tool Definition and Registration](#)

[6.4 Tool Execution Flow](#)

[6.5 Common Integration Categories](#)

[6.6 Integration Guarantees](#)

[7. Member Intelligence](#)

[7.1 Member Sync](#)

[7.2 Member Data Model](#)

[7.3 Members Workspace](#)

[7.4 Atlas: Conversational Analytics](#)

[8. Post-Conversation Analysis Pipeline](#)

[8.1 Analysis Flow Overview](#)

[8.2 Analysis Responsibilities](#)

[8.3 Staff Action Notifications](#)

[8.4 Quality Feedback Loop](#)

[9. Data Handling and Retention](#)

[9.1 Conversation Lifecycle Diagram](#)

[9.2 Data Flow Overview](#)

[9.3 Data Retention Policy](#)

[10. Customer Data Flow: Voice Modality Example](#)

[10.1 End-to-End Voice Call Data Flow](#)

[10.2 Step-by-Step Data Journey](#)

[10.3 External Providers: Data Touch Points](#)

[10.4 Final Data Destinations](#)

[11. External Service Providers](#)

[11.1 AI and Speech Providers](#)

[11.2 Infrastructure Providers](#)

[11.3 Application Services](#)

[12. Regional Deployment Model](#)

[13. Security Architecture](#)

[13.1 Data Protection](#)

[13.2 Tool Execution Security](#)

[13.3 Access Controls and Auditability](#)

[14. Compliance and Operational Maturity](#)

[14.1 What SOC 2 means](#)

[14.2 Agent Evaluation and Testing](#)

[14.3 Observability](#)

[15. Founders](#)

[Ryan Junejo \(CEO\)](#)

[Mathusan Selvarajah \(CTO\)](#)

[16. Closing Note](#)

Executive Summary

Sunday is an AI operations platform for fitness and wellness businesses. It answers front-line customer communications across phone and email, carries out complete member workflows (booking, cancellations, membership freezes, and late-cancel exceptions) directly inside the customer's booking system, and turns the resulting data into a searchable picture of every member.

This document provides technical architecture details for customer due diligence. It is intended to support third-party technical, security, and systems reviews by outlining system design, data flows, operational boundaries, and implementation choices.

Note: The detailed data flow walkthrough (Section 10) uses the **voice** modality (phone calls) as its worked example. The email agent and member intelligence layer are described in their own sections.

Core Technical Components:

- Real-time voice agent with deterministic, multi-step workflow execution
- Draft-only email agent with human review before anything is sent
- Booking platform integrations (Mariana Tek, Mindbody, InChargeLife) behind a single provider interface
- Member intelligence layer with a conversational analytics agent (Atlas)
- Semantic knowledge retrieval system
- Asynchronous, event-driven conversation analysis and notification pipeline
- Isolated, customer-specific integration workers

What changed since Version 1:

- The voice stack moved from a hosted voice provider (Vapi) to a Sunday-operated agent runtime on LiveKit Cloud, with Telnyx for telephony and Cartesia for speech synthesis.
- A production email agent now drafts replies in customers' Gmail and Outlook inboxes.
- Sunday now syncs member, visit, and plan data from booking platforms and exposes it through a Members workspace and the Atlas analytics agent.
- Web chat has been retired as a channel.

1. Terminology

This section defines key terms used throughout this document to avoid ambiguity during third-party review.

1.1 Control Plane

API Layer

- Stateless HTTP APIs responsible for request validation, authentication, and orchestration.
- Implemented using FastAPI.
- Serves two surfaces: the customer dashboard API (authenticated end users) and an admin API used by Sunday's own agent runtimes.

Background Job Layer

- Durable, event-driven functions for everything that should not block a live conversation: post-conversation analysis, notifications, member sync, knowledge base ingestion, and scheduled maintenance.

1.2 Agent Runtimes

Definition: An agent runtime is the process that holds a live conversation. It loads the agent's configuration from the control plane, talks to the model providers, calls tools, and reports the finished conversation back to the control plane.

Sunday operates two runtimes:

- **Voice runtime:** a LiveKit agent deployed on LiveKit Cloud. One isolated agent job runs per phone call.
- **Email runtime:** an agent built on the Claude Agent SDK and deployed on Modal. One isolated run executes per inbound email thread.

Neither runtime stores durable customer data. Configuration is fetched per conversation, and results are written back through the control plane.

1.3 Edge Workers

Definition: Edge workers are isolated, customer-specific execution units used to run lightweight integration logic (e.g., reading class schedules or membership catalogs from a customer's systems).

Characteristics:

- Deployed on Cloudflare Workers for Platforms.
- Execute in isolated V8 environments.
- No shared runtime state across customers.
- Provider credentials are held in Cloudflare Secrets Store, scoped per worker.

Purpose:

- Reduce blast radius of failures.
- Prevent cross-customer data access.
- Enable customer-specific logic without shared code paths.

1.4 Event Queue

Definition: A durable, managed event and workflow engine (Inngest) used to decouple ingestion of events from downstream processing.

Usage:

- Conversation lifecycle events
- Post-conversation analysis
- Notification dispatch
- Member sync and knowledge base ingestion
- Scheduled jobs (phone routing schedules, website re-scrapes, email draft reconciliation)

Properties:

- Durable, step-based execution: each step's result is persisted, so a retry resumes from the failed step rather than starting over
- Automatic retries with backoff
- Per-key concurrency limits and throttling (e.g., one member sync per booking connection at a time)

1.5 KV Store

Definition: A low-latency key-value store (Cloudflare KV) used for non-relational, high-read data at the edge.

Stored Data Includes:

- Cached class schedules and membership catalogs used by edge workers
- Tool routing metadata

Properties:

- Caches are refreshed on a schedule (hourly), so live calls read from the edge instead of hitting customer systems on every request.
- No customer credentials are stored in KV.

1.6 Vector Database

Definition: A specialized database used to store semantic embeddings for documents and website content.

Usage:

- Knowledge base retrieval
- Semantic similarity search

Properties:

- Cosine similarity search
- One namespace per customer knowledge base, so retrieval is scoped by construction
- Metadata filtering by source (website, document, URL, heading)
- No raw documents are reconstructed from vectors alone

1.7 Booking Provider Layer

Definition: A single internal interface that every booking platform integration implements. Sunday currently supports Mariana Tek, Mindbody, and InChargeLife (ICL), plus Pipeline as a CRM-only source.

Purpose:

- Live actions during a conversation: look up a member, book a class, cancel a reservation, check late-cancel status, cancel or freeze a membership.
- Bulk member sync: roster, visits, plans, and lifecycle status for member intelligence.
- Adding a new booking platform is additive: one new provider implementation, with no changes to the agents or shared code.

1.8 Speech Terms (STT, TTS, VAD, Turn Detection)

STT (Speech-to-Text): The process of converting a caller's audio into text in real time (often producing both partial and final transcripts).

TTS (Text-to-Speech): The process of converting the agent's generated text into audio for playback to the caller (often streamed so playback can start before the full response is complete).

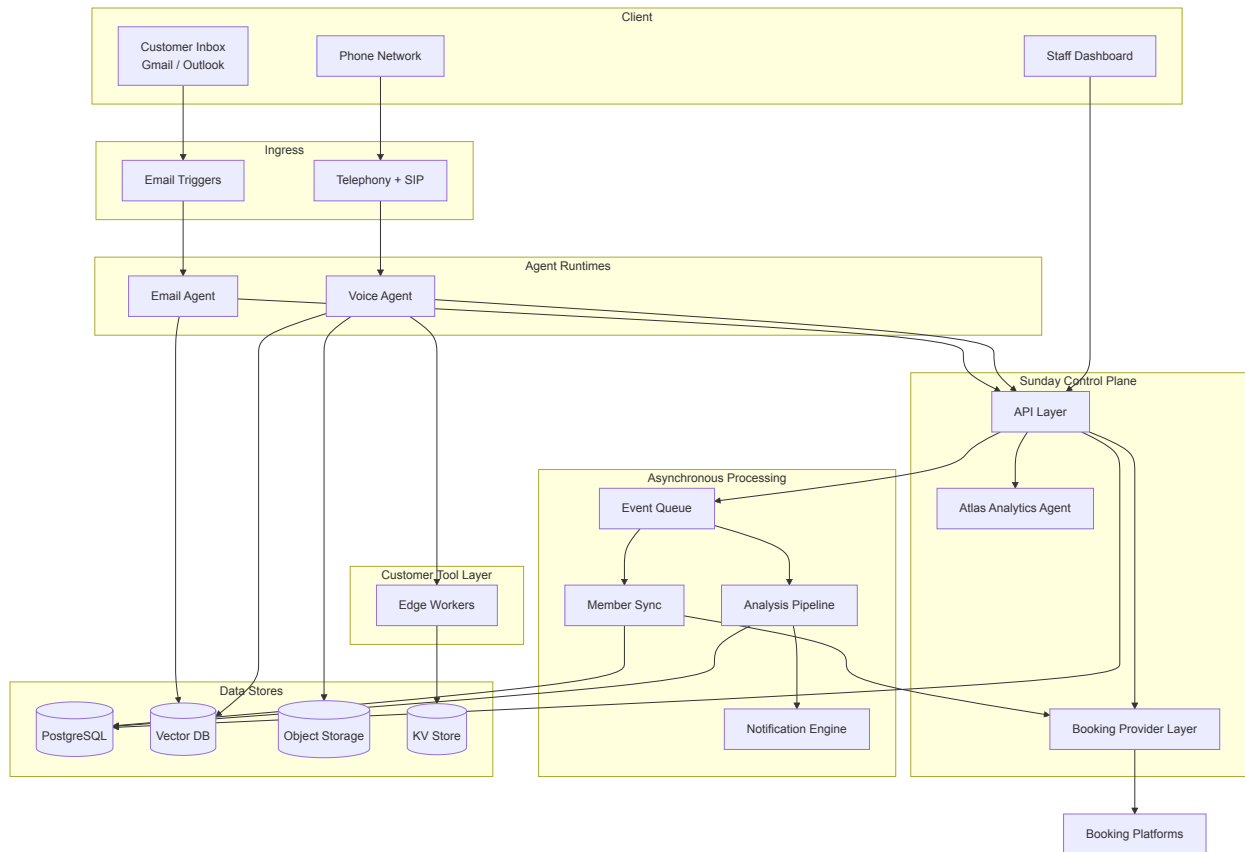
VAD (Voice Activity Detection): The process of detecting when a caller is speaking vs silence/background noise.

Turn Detection: The decision that the caller has finished their turn and the agent should respond. Sunday uses a speech model that predicts end-of-turn directly from the audio and transcript, rather than waiting for a fixed silence timeout. This makes the agent respond faster without talking over callers who pause mid-sentence.

2. System Architecture Overview

Sunday is designed as a modular, event-driven system that separates real-time customer interactions from post-conversation processing. This separation ensures reliability, fault isolation, and operational flexibility as the system scales.

2.1 High-Level System Architecture



2.2 Architectural Principles

Separation of Real-Time and Asynchronous Processing

- Real-time paths handle live phone interactions and are optimized for responsiveness.
- Asynchronous paths handle transcript analysis, structured extraction, notifications, member sync, and email draft tracking.
- Failures in asynchronous systems do not interrupt live conversations.

Stateless Control Plane

- API services are stateless and horizontally scalable.
- All durable state is stored in managed data stores.
- No session affinity is required at the load balancer level.

Deterministic Actions, Conversational Tasks

- The language model handles conversation: understanding the caller, collecting details, and confirming consent.
- Actions that change a member's account (booking, cancelling, freezing) are executed by ordinary code once the required steps have succeeded. The model never decides on its own to call a critical API.

Human in the Loop Where It Matters

- The email agent writes drafts only. A staff member reviews and sends every reply.
- Atlas can read member data freely but must ask for approval before saving anything.

Customer Isolation by Design

- All customer resources (agents, contacts, conversations, knowledge bases, integrations) are scoped to a team.
 - Each customer's edge integrations run in isolated execution environments.
 - Each conversation runs in its own isolated agent job.
-

3. Voice Agent and Speech Stack

The voice agent is the low-latency boundary between telephony and Sunday's real-time orchestration. Since Version 1, Sunday operates this layer directly on LiveKit Cloud rather than through a hosted voice provider. This gives Sunday full control over latency, turn-taking, call transfers, and multi-step workflows.

3.1 How a call reaches the agent

1. **Carrier:** The caller dials a Sunday-provisioned Telnyx number.
2. **SIP bridge:** Telnyx routes the call over SIP to LiveKit.
3. **Dispatch:** LiveKit creates a dedicated room for the call and dispatches a Sunday agent job, tagged with the customer's agent id.
4. **Configuration:** The agent fetches its configuration from the control plane: system prompt, voice, knowledge bases, tools, transfer destinations, and which booking workflows this customer has enabled.

5. **Caller context:** In parallel, the agent looks up the caller's phone number against the customer's member records, so it can greet a known member with the right context before the first word.

Phone routing modes: Each customer number can run in Sunday mode (answered by the agent) or forward mode (sent straight to the front desk). Customers can switch modes on a schedule (e.g., Sunday answers after hours) or with a manual override.

3.2 How the voice loop works

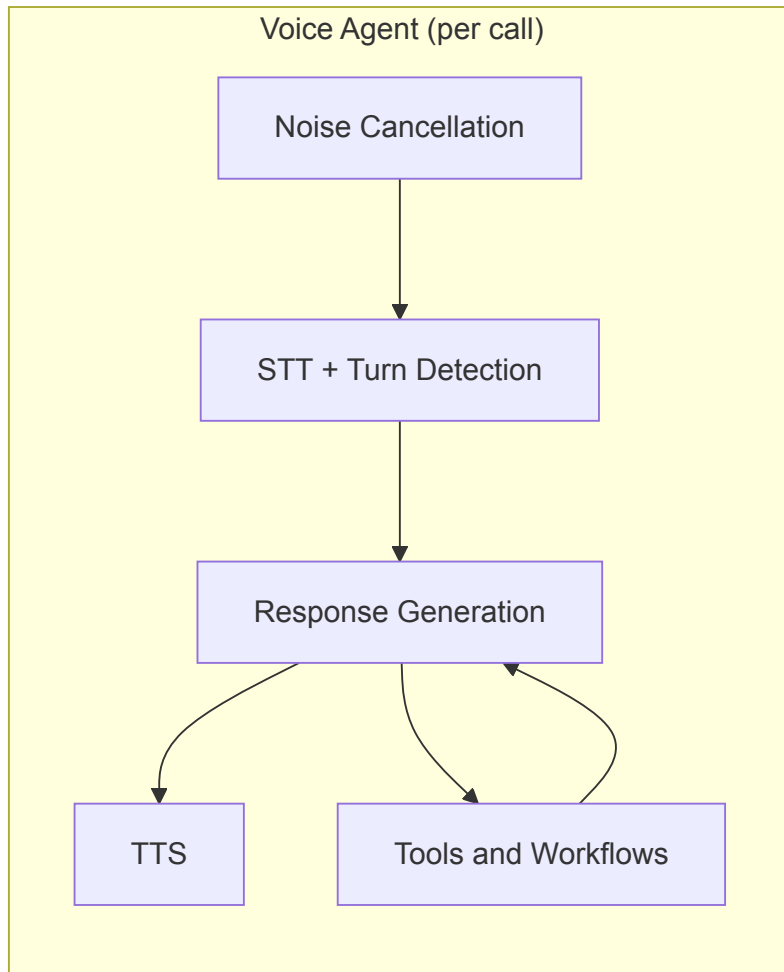
At runtime, the voice agent operates as a continuous loop that converts audio into incremental text, generates a response, and streams synthesized audio back to the caller.

Core loop:

1. **Noise cancellation:** Incoming audio is cleaned with a telephony-tuned noise cancellation model.
2. **STT + turn detection (streaming):** Speech is transcribed into incremental text. The same speech model predicts when the caller has finished their turn.
3. **Response generation:** The transcript, conversation history, and caller context are sent to the language model, which either replies or calls a tool.
4. **TTS (streaming synthesis):** The response text is converted into audio frames as it arrives. Common phrases (greetings, transfer messages) are pre-generated when the call starts so they play instantly.
5. **Audio playback:** The synthesized audio stream is sent back over SIP for real-time playback to the caller.

While a tool is running (for example, checking class availability), the agent speaks short natural filler phrases so the caller never hears dead air.

Voice agent components:



3.3 Guided workflows

Account-changing requests run as guided, multi-step workflows. Each workflow is a sequence of small, focused conversational tasks, with deterministic code in between.

Supported workflows:

- Account lookup and identity verification
- Class booking (including payment option checks)
- Reservation cancellation, with late-cancel policy checks and exception requests
- Membership cancellation
- Membership freeze

How a workflow runs (example: membership freeze):

1. A task confirms the caller's identity.
2. Code asks the booking platform for a freeze preview (dates, fees, eligibility).
3. A task explains the terms and collects the caller's explicit consent.
4. Only after consent does code call the booking platform to apply the freeze.
5. The outcome is reported back to the main agent, which tells the caller what happened.

If the caller changes topic at any point, the workflow exits cleanly and control returns to the main agent. No action is taken on a half-finished workflow.

3.4 Call transfer

- Callers can be transferred to the front desk or to a specific location.
- Transfers use SIP REFER by default. For customers who need guaranteed handoff, Sunday supports an attended transfer that dials the destination, confirms it answered, and retries on common carrier failures before connecting the caller.

3.5 Call recording

- Calls are recorded as audio-only files by LiveKit and stored in Sunday's object storage (Cloudflare R2).
- Staff access recordings from the dashboard through short-lived signed links that expire after one hour.

4. Email Agent

The email agent answers customer email in the customer's own inbox. It is designed around one rule: **Sunday never sends email on its own**. It writes a draft, and a staff member decides whether to send it.

4.1 How email reaches the agent

1. **Connection:** The customer connects a Gmail or Outlook inbox through an OAuth-based integration (Composio).
2. **Trigger:** New messages trigger a webhook to the email runtime.

3. **Configuration:** The agent fetches its configuration from the control plane: instructions, knowledge bases, workflows, and read-only booking access.
4. **Run:** One isolated agent run executes per email thread.

4.2 Filtering before any model runs

Most inbound email should not get a reply (newsletters, receipts, internal staff mail). Sunday filters in three layers, cheapest first:

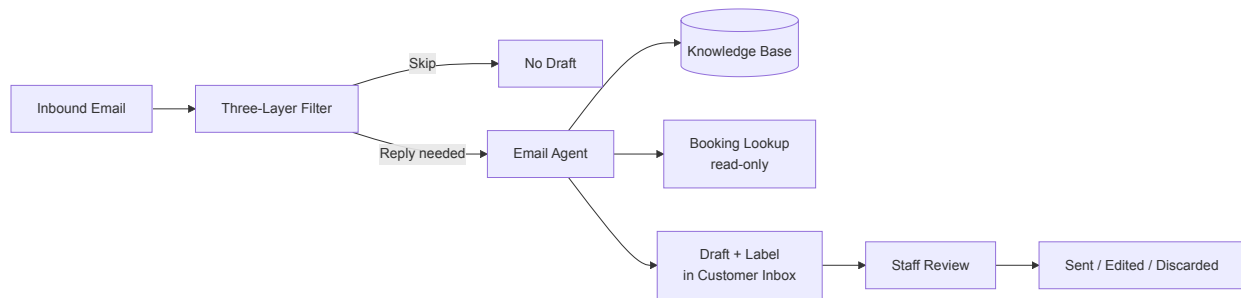
1. **Heuristic pre-filter:** drops obvious non-customer mail at the webhook.
2. **Control plane gate:** skips duplicates, the customer's own staff, and senders on the customer's ignore list.
3. **Model decision:** the agent itself decides whether the thread needs a reply at all.

4.3 Draft-only by design

- The agent can read the thread, search the knowledge base, and look up (but not change) booking information.
- The only write path is through the control plane, which creates or updates one draft and applies labels in the customer's inbox.
- There is no code path that sends email.
- A reconciliation job checks every few minutes whether each draft was sent as-is, sent with edits, or discarded.

4.4 Email workflows

Customers can define **workflows**: step-by-step playbooks for common email types (e.g., freeze requests, new member inquiries). Workflows are edited on a visual canvas in the dashboard, and can be generated from a plain-language description or imported from existing email templates. The agent treats a workflow as guidance for how to handle a thread, and keeps notes across replies within the same thread.



5. Knowledge Base Ingestion and Retrieval

Sunday's agents are only as good as the business knowledge they can reliably use at runtime. A dedicated knowledge base matters because it:

- **Improves answer quality and consistency:** Responses are grounded in customer-owned documents and website content instead of model "best guesses."
- **Reduces operational load:** Fewer escalations and shorter interactions by answering common questions correctly on the first try.
- **Makes updates fast and low-risk:** Teams can change policies/pricing/processes by updating source content, without redeploying application code.
- **Supports compliance and auditability:** Knowledge injection provides a clearer story for "why did the agent say that?" compared to purely generative behavior.

The same knowledge bases serve both the voice agent and the email agent.

5.1 How ingestion and retrieval works

Sunday's knowledge base is a retrieval layer that turns customer documents/websites into searchable embeddings, then injects the most relevant chunks into the agent's context at runtime.

Ingestion pipeline:

1. **Source capture:** Customer uploads documents (PDF, markdown, text) and/or provides website URLs. Websites can be re-scraped on a schedule so answers stay current.
2. **Parse + clean:** Web pages are converted to clean markdown. PDFs are parsed with a layout-aware parser that preserves headings and tables. Original files are kept in object

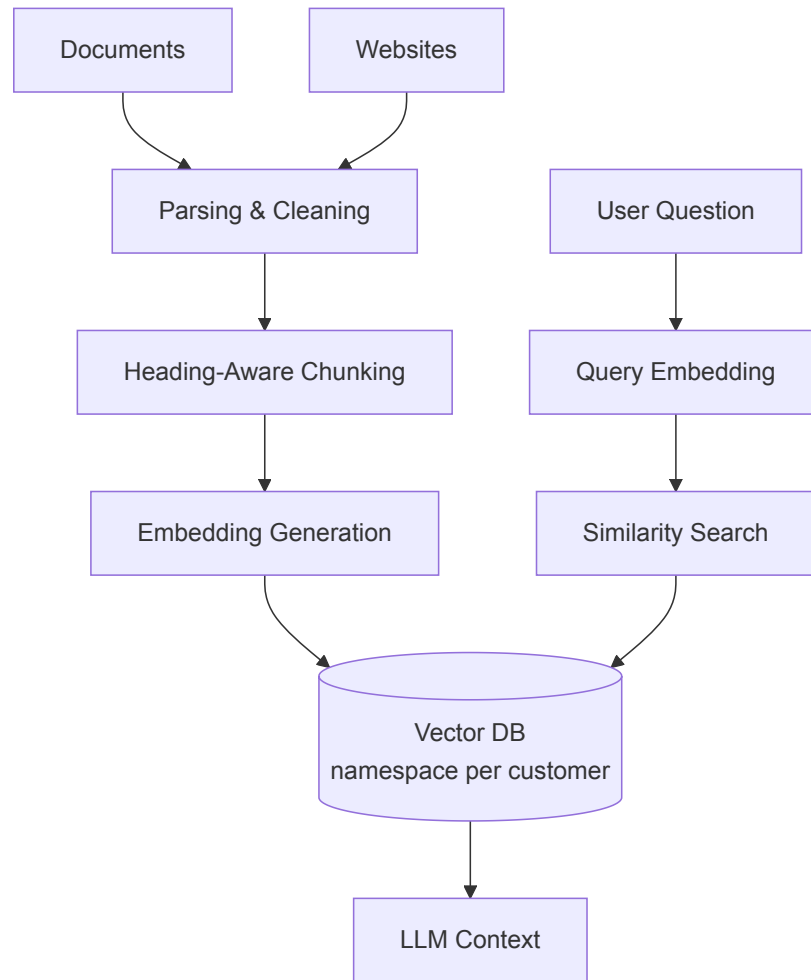
storage.

3. **Chunking:** Content is split first by headings, then into overlapping chunks sized for retrieval. Each chunk carries its heading path so it stays meaningful on its own.
4. **Embedding:** An embedding vector is generated per chunk. Embeddings are computed on Sunday's own infrastructure with an open-source model; document text is not sent to a third-party embedding API.
5. **Index + metadata:** Vectors are stored in a namespace per customer knowledge base, with metadata (source, URL/document id, title, headings).

Retrieval at runtime:

1. **Query embedding:** Create an embedding from the user's question, using the same model.
2. **Similarity search:** Retrieve the top 5 chunks from that customer's namespace.
3. **Context injection:** Return the selected chunks to the agent so the model can answer using grounded business knowledge.

Each knowledge base is exposed to the agent as its own tool, so the agent chooses which source to consult. Namespaces are pre-warmed when a call starts to keep first-query latency low.



This diagram shows how documents and websites are ingested, embedded, and retrieved during agent responses.

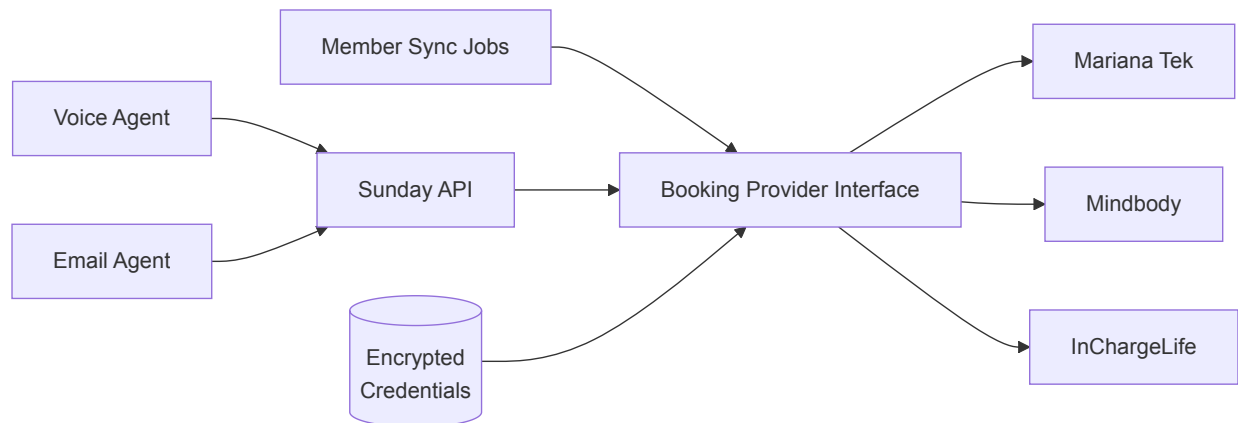
6. Integrations Architecture

Sunday connects to customer systems through two paths, chosen by what the integration needs to do:

- **Booking provider layer (control plane):** for anything that reads or changes a member's account. These actions run through Sunday's API, using encrypted per-customer credentials, so they are logged, validated, and never exposed to the agent runtimes.

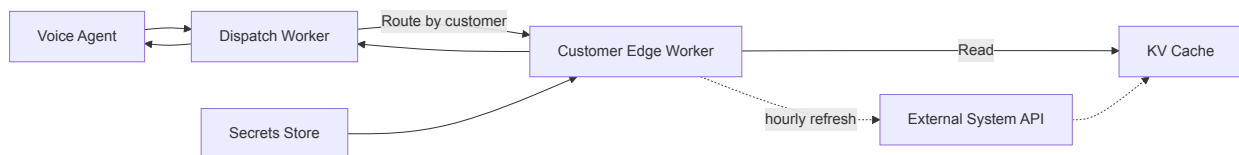
- **Edge workers (tool layer):** for fast, read-only lookups during a call, such as class schedules and membership catalogs, served from an edge cache.

6.1 Booking Provider Integrations



- Every platform implements the same interface: member lookup, reservations, booking, late-cancel checks, membership cancel/freeze previews and actions, and bulk member sync.
- Platform-specific behavior (e.g., whether a confirmation email follows a cancellation) is returned by the provider as facts, so the agents behave correctly without platform-specific logic.
- Studio policies such as late-cancel windows and fees are configured per customer, not hardcoded.
- Where a platform does not offer a suitable public API, Sunday operates a dedicated integration service that maintains authenticated sessions on the customer's behalf.

6.2 Edge Tool Execution and Isolation Diagram



This diagram illustrates the execution path and isolation guarantees for customer-specific edge tools. Each customer gets its own subdomain and its own workers; the dispatcher routes each request only to that customer's workers.

6.3 Tool Definition and Registration

Each tool is defined using a strict JSON schema that describes its purpose and inputs. Tool schemas are attached to an agent configuration and provided to the LLM at runtime.

Example tool schema:

```
{
  "name": "check_availability",
  "description": "Check available appointment slots",
  "parameters": {
    "type": "object",
    "properties": {
      "date": { "type": "string", "format": "date" },
      "service": { "type": "string" }
    },
    "required": ["date"]
  }
}
```

Any bearer tokens attached to a tool are encrypted at rest (AES-256-GCM) and are not included in the configuration sent to agent runtimes.

6.4 Tool Execution Flow

1. The LLM determines that external data or an action is required.
2. A structured tool call is returned instead of natural language output.
3. **Read-only lookups** are routed to the customer's edge worker, which answers from its cache.
4. **Account actions** are sent to the Sunday API, which authorizes the request against the agent's customer, loads that customer's encrypted booking credentials, and calls the booking platform.
5. Results are normalized and returned to the agent as plain guidance on what to tell the caller, never as raw API responses or internal errors.

6.5 Common Integration Categories

- Scheduling and booking systems (Mariana Tek, Mindbody, InChargeLife)

- CRM platforms
- Email inboxes (Gmail, Outlook)
- Staff notification channels (email, Slack, spreadsheets)
- Customer phone systems (for importing calls handled by staff)

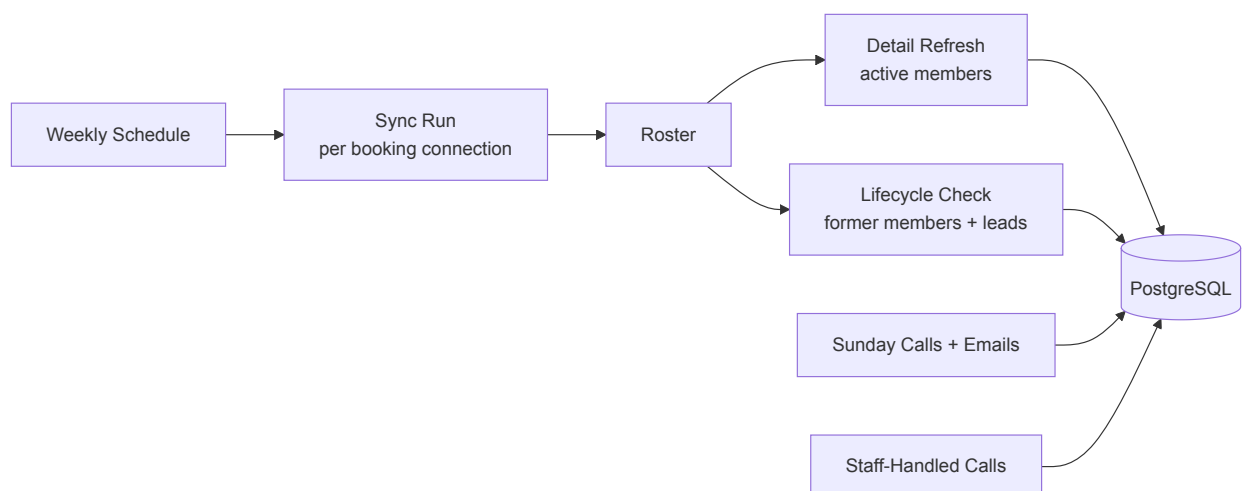
6.6 Integration Guarantees

- Customer isolation at the worker level
- No shared credentials across customers
- Customer credentials encrypted at rest and decrypted only inside the control plane
- Deterministic execution of account actions outside the LLM
- Adding a booking platform does not change shared agent code

7. Member Intelligence

Sunday started as a way to answer calls. Every conversation, though, is about a member: their plan, their attendance, their reason for calling. Member intelligence brings that context together in one place, so staff (and Sunday's agents) can see who a member is and how they are trending.

7.1 Member Sync



- Member data is synced from each connected booking platform on a weekly schedule, with a one-time backfill during onboarding.
- Only one sync runs per booking connection at a time, and requests are throttled to respect each platform's rate limits.
- Sunday's own calls and emails are linked to the same member records automatically.
- Customers can optionally import calls handled by their own staff (from RingCentral or recording archives). These are transcribed; the audio itself is not retained.

7.2 Member Data Model

For each member, Sunday stores:

- **Profile:** name, contact details, plan, member-since date, referral source, tags
- **Lifecycle status:** member, lead, former member, or unknown
- **Visits:** class, instructor, location, time, and status (booked, attended, no-show, late cancel, cancelled, waitlist)
- **Engagement events:** every Sunday call or email, with its intent, outcome, and summary
- **Summary fields:** lifetime visits, last visit, and lifetime value

7.3 Members Workspace

- A filterable member list with saved views (e.g., "members who haven't visited in 3 weeks").
- A member detail page showing 90-day attendance, visits-per-week trend, lifetime value and tenure, days since last visit, a weekly schedule heatmap, and a full timeline of visits, calls, and emails.
- All metrics are computed directly from the synced data with standard database queries, so every number traces back to a source record.

7.4 Atlas: Conversational Analytics

Atlas is the chat interface on the Sunday dashboard. Staff ask questions in plain language ("Which members on the unlimited plan stopped coming this month?") and Atlas answers from the customer's data.

- **Structured queries, not data dumps:** Atlas writes typed queries against a fixed catalog of member fields and metrics. The control plane compiles and runs them, and enforces

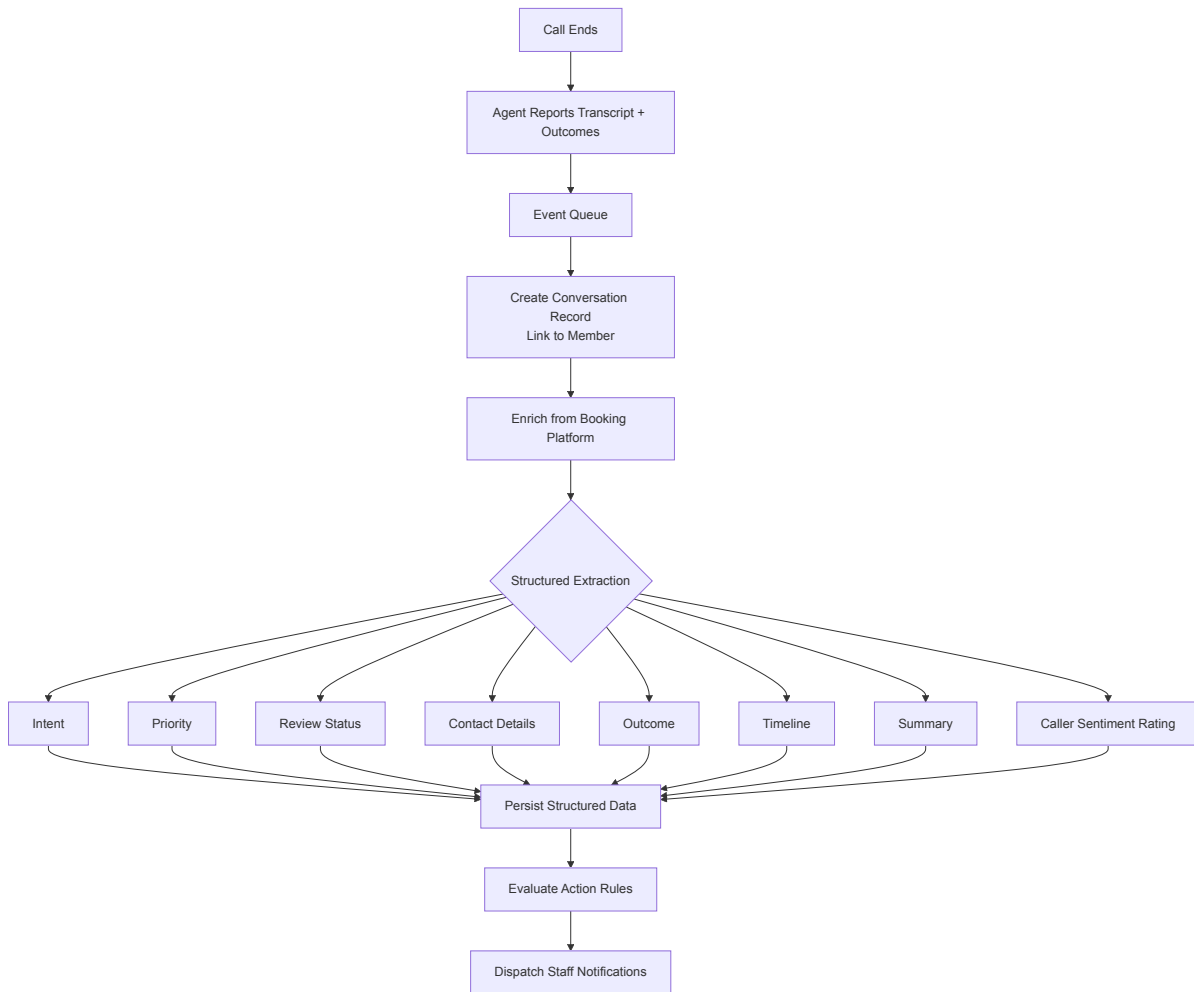
that every query is scoped to the asking customer.

- **Read tools:** member lists and profiles, visits and attendance, call and email history, and knowledge base search.
 - **Approval-gated writes:** the only write action (saving a member view) requires the user's explicit approval.
 - **Operator-defined segments:** Atlas does not apply a hidden "at-risk" score. It asks the operator what they mean (e.g., "no visit in 21 days") and builds the segment transparently.
-

8. Post-Conversation Analysis Pipeline

Sunday performs structured analysis after each conversation using an asynchronous pipeline. This design ensures that analysis, notifications, and downstream updates never block live customer interactions.

8.1 Analysis Flow Overview



8.2 Analysis Responsibilities

The post-conversation pipeline is responsible for:

- Classifying the primary intent of the interaction from the customer's own list of intents
- Determining urgency and whether staff review or escalation is needed
- Extracting structured contact information
- Recording the outcome and a timeline of what happened on the call
- Generating concise conversation summaries
- Linking the conversation to the member record

Each extraction task runs as its own durable step, so a failure in one is retried independently without redoing the others.

8.3 Staff Action Notifications

When the agent completes an account action, staff are notified so nothing happens silently:

- Membership cancelled
- Membership frozen
- Late-cancel fee exception requested

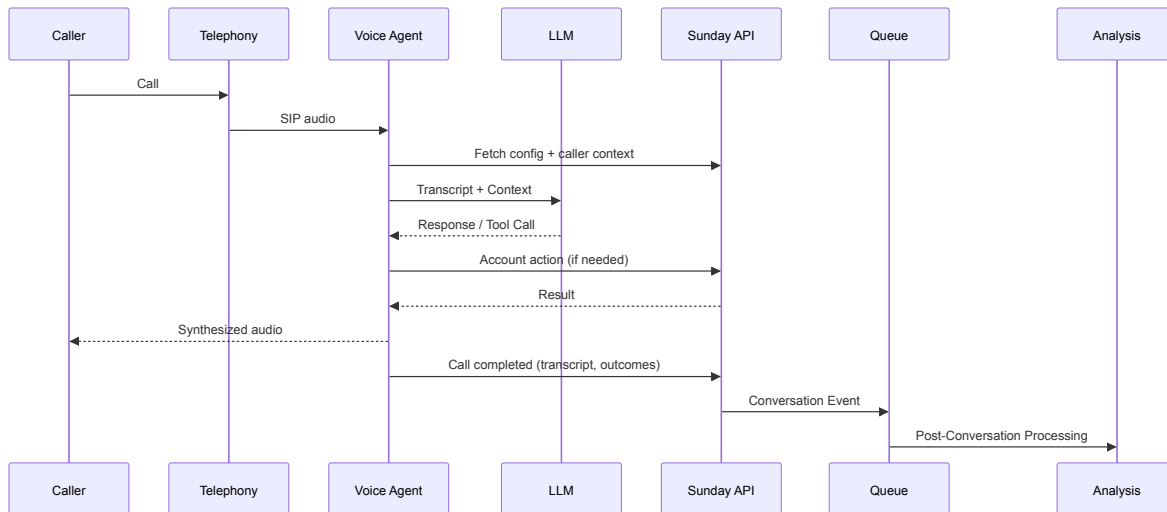
Recipients are configured per customer in the dashboard. Customer-specific destinations (such as a retention team's tracking sheet) are supported.

8.4 Quality Feedback Loop

- **Call review:** Every call is marked as handled by Sunday, needing review, or escalated. Staff resolve calls from the dashboard and can leave feedback on any call.
 - **Email edit analysis:** When staff edit a draft before sending, Sunday classifies the change (minor polish, tone shift, content change) so the team can see where the agent needs improvement.
 - **Shift summaries:** Customers can opt into a periodic summary of the day's calls posted to their team chat.
-

9. Data Handling and Retention

9.1 Conversation Lifecycle Diagram



9.2 Data Flow Overview

- Live audio is streamed to speech providers for transcription and synthesis during the call.
- Call recordings (audio only) are stored in Sunday's object storage and made available to the customer's staff through the dashboard.
- Conversation transcripts are stored in structured form for analysis and auditability.
- Extracted metadata (intent, outcome, contact details) is stored separately from raw transcripts.
- Member data synced from booking platforms is stored in Sunday's database and scoped to the customer.
- For imported staff-handled calls, only the transcript is kept; the source audio is not stored.

For a concrete end-to-end walkthrough of how this applies to a **voice call**, see **Section 10**.

9.3 Data Retention Policy

Retention policies are designed to support customer compliance needs.

Defaults:

- Conversation transcripts, recordings, and structured metadata: retained for the life of the customer account to support call review, reporting, and member history
- Member data: refreshed on every sync and retained for the life of the customer account
- Knowledge base content: retained until removed by the customer; deleting a source removes its vectors, stored file, and records
- Email agent run records: automatically deleted 60 days after completion
- Imported call audio: not stored

Customers may request custom retention or deletion policies.

10. Customer Data Flow: Voice Modality Example

This section provides a detailed walkthrough of customer data flow using voice calls as an example. It addresses the critical question: **"Where does my data travel, and where does it ultimately reside?"**

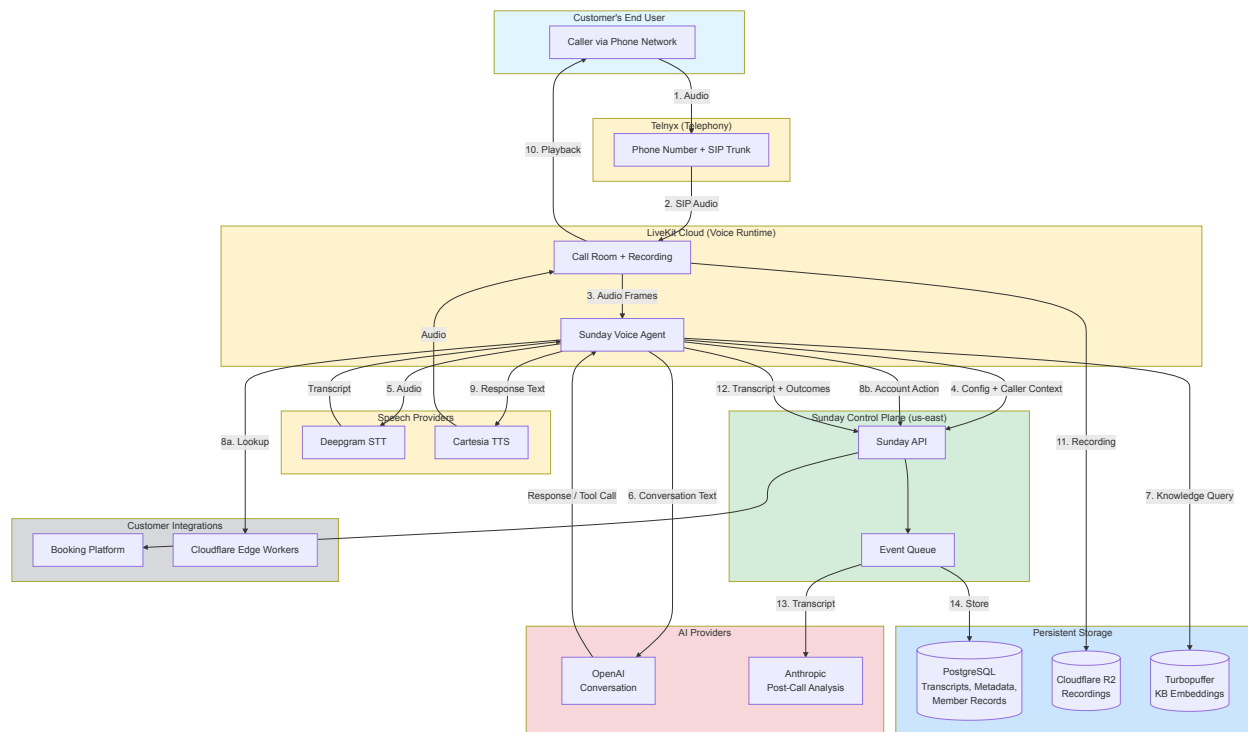
Understanding the complete data journey is essential for assessing data leak risk, regulatory compliance, and security posture.

TLDR:

- **In transit (real-time processing):** audio is carried by the telephony provider (**Telnyx**) and the voice runtime (**LiveKit Cloud**), transcribed by **Deepgram**, answered by **OpenAI**, and synthesized by **Cartesia** to execute the call.
- **Persisted by Sunday:** conversation **transcripts + metadata** and **member records** in **PostgreSQL (Railway)**; **call recordings** in **Cloudflare R2**; knowledge base **embeddings** in **Turbopuffer (vector DB)**.
- **Analyzed after the call:** transcripts are sent to **Anthropic** for structured extraction.
- **Controls:** encryption in transit/at rest, tenant isolation, and least-privilege access controls are described in **Section 13**.

10.1 End-to-End Voice Call Data Flow

When a customer's member makes a voice call to Sunday, their data passes through multiple stages and touches different providers before being stored. The following walkthrough traces this journey step-by-step.



10.2 Step-by-Step Data Journey

Phase 1: Call Setup

1. Caller → Telnyx (Telephony Provider)

- Caller dials the customer's Sunday phone number
- Telnyx handles the phone number, carrier routing, and SIP trunking
- **Data: Real-time audio packets, caller phone number**
- **Provider: Telnyx**

2. Telnyx → LiveKit Cloud (Voice Runtime)

- The call is bridged over SIP into a dedicated LiveKit room
- LiveKit dispatches an isolated Sunday agent job for this call

- **Data: Real-time audio**
- **Provider: LiveKit**

3. Voice Agent → Sunday API

- The agent loads the customer's agent configuration
- The agent looks up the caller's phone number in the customer's member records
- **Data: Agent configuration, caller phone number, member profile summary**

Phase 2: Conversation Processing

4. Voice Agent → Deepgram (Speech-to-Text)

- Caller audio is streamed for transcription and end-of-turn detection
- **Data: Audio stream → Text transcripts**
- **Provider: Deepgram**

5. Voice Agent → OpenAI (Response Generation)

- The agent sends the system prompt, conversation history, caller context, and tool schemas
- The model returns a response or a tool call
- **Data: Conversation text, prompts, member context**
- **Provider: OpenAI (GPT-5.4 mini)**
- **Data use: Configured so API data is not used for provider model training (per provider terms / enterprise settings)**

Phase 3: Knowledge Retrieval (if needed)

6. Vector Database Query

- The question is embedded on Sunday's infrastructure
- Similarity search runs against the customer's knowledge base namespace
- **Data: Question embedding, retrieved knowledge chunks**
- **Storage: Turbopuffer (vector DB)**

Phase 4: Customer Tool Execution (if needed)

7. Read-only lookups → Cloudflare Workers

- Class schedules and membership catalogs are served from the customer's edge worker cache
- **Data: Structured tool parameters (e.g., date, class type)**
- **Provider: Cloudflare edge network**

8. Account actions → Sunday API → Booking Platform

- Booking, cancellation, and freeze actions go through the Sunday API
- The API decrypts the customer's booking credentials and calls the booking platform
- **Data: Member identifiers, reservation and membership details**
- **Provider: Customer's own booking platform**

Phase 5: Response Generation & Audio Synthesis

9. Voice Agent → Cartesia (Text-to-Speech)

- Response text is streamed to the TTS service
- Returns an audio stream
- **Data: Text → Audio stream**
- **Provider: Cartesia**

10. LiveKit → Telnyx → Caller

- Audio streamed back over SIP
- Played to caller in real time
- **Data: Audio stream**

Phase 6: Persistent Storage and Analysis

11. Call Recording → Cloudflare R2

- Audio-only recording of the call
- **Location: Cloudflare R2 (Sunday-owned bucket)**
- **Access: Dashboard only, via signed links that expire after one hour**

12. Transcript + Outcomes → Sunday API → Event Queue

- The agent reports the transcript, duration, and outcome flags (e.g., membership frozen)
- **Data: Full conversation text, outcome metadata**

13. Post-Call Analysis → Anthropic

- Transcript sent for structured extraction (intent, priority, summary, contact details)
- **Data: Conversation text, analysis prompts**
- **Provider: Anthropic (Claude Sonnet 4.6)**

14. PostgreSQL (Railway)

- Conversation transcript and structured metadata stored
- Linked to the member record
- **Location: Railway-managed Postgres (us-east)**
- **Encryption: Encrypted at rest (provider-managed)**

10.3 External Providers: Data Touch Points

All third-party providers listed below are selected to meet enterprise security expectations and maintain SOC 2 compliance (typically SOC 2 Type II, or an equivalent independent assurance report). Vendor reports can be provided under NDA as part of due diligence. See **Section 14.1** for what SOC 2 means.

The following table summarizes all external providers that process customer data during a voice call:

Provider	Purpose	Data Processed	Data Retention by Provider	Location	Compliance
Telnyx	Phone numbers, carrier routing, SIP	Real-time audio, phone numbers	Call metadata per carrier requirements	US	SOC 2
LiveKit	Voice runtime, media routing, recording	Real-time audio	Not retained beyond call (recordings written to Sunday's R2)	US	SOC 2
Deepgram	Speech-to-text transcription	Audio → Text transcripts	Not used for training; short operational retention	US	SOC 2

Provider	Purpose	Data Processed	Data Retention by Provider	Location	Compliance
OpenAI	Conversational AI generation	Conversation text, prompts	Not used for training (API terms / enterprise settings)	US	SOC 2
Cartesia	Text-to-speech synthesis	Response text → Audio	Not retained beyond synthesis	US	SOC 2
Anthropic	Post-call analysis	Conversation text, analysis prompts	Not used for training (API terms / enterprise settings)	US	SOC 2
Cloudflare	Edge workers, recording storage	Tool parameters, cached schedules, recordings	Recordings persistent; cache short-lived	Global edge / R2	SOC 2
Railway	Application + database hosting	Transcripts, metadata, member records	Persistent (configurable retention)	us-east	SOC 2
Turbopuffer	Vector database	Embeddings + retrieval metadata	Persistent until explicitly deleted	us-east	SOC 2

10.4 Final Data Destinations

Where Customer Data Ultimately Resides:

1. PostgreSQL (Railway)

- Conversation transcripts (text)
- Structured metadata (intent, priority, outcome, contact details)
- Member records, visits, and engagement history
- Encrypted integration credentials (AES-256-GCM)
- **Retention: Life of the customer account unless deletion is requested**
- **Encryption: Encrypted at rest (cloud-provider managed) and encrypted in transit (TLS)**

2. Object Storage (Cloudflare R2)

- Call recordings (audio only)
- Uploaded knowledge base documents
- **Access: Signed links that expire after one hour**

- **Encryption: Encrypted at rest (cloud-provider managed)**

3. Vector Database (Turbopuffer)

- Customer knowledge base embeddings
- Document/website content chunks (vectorized)
- **Retention: Until explicitly deleted by customer**
- **Encryption: Encrypted at rest (cloud-provider managed)**

4. KV Store (Cloudflare)

- Cached class schedules and membership catalogs
- **Retention: Short-lived, refreshed hourly**

Important: Live audio is only persisted as the call recording

- Audio streamed to speech providers is processed in real time and not retained by Sunday outside the recording
- Recordings are stored only in Sunday's own storage, never in a provider's
- Imported staff-handled calls are stored as transcripts only

This section is a worked example; security controls and residency posture are consolidated in **Section 12** and **Section 13**.

11. External Service Providers

Sunday integrates with a set of third-party providers to deliver voice, language, and infrastructure capabilities. These providers are used in clearly scoped roles. All providers that process customer data are selected to meet enterprise security expectations and maintain SOC 2 compliance. See **Section 14.1**.

11.1 AI and Speech Providers

- **OpenAI** – Large language model for real-time voice conversations
- **Anthropic** – Large language models for the email agent, post-call analysis, and the Atlas analytics agent
- **Deepgram** – Speech-to-text transcription and end-of-turn detection

- **Cartesia** – Voice synthesis
- **LiveKit** – Real-time voice runtime, media routing, and call recording
- **Telnyx** – Phone numbers, carrier connectivity, and SIP trunking

Customer data sent to these providers is limited to what is required for request execution. Provider data use controls are described in **Section 13**.

11.2 Infrastructure Providers

- **Railway** – Application and database hosting (including managed Postgres)
- **Cloudflare** – Edge workers, KV cache, object storage (R2), and DNS
- **Turbopuffer** – Vector database for knowledge retrieval
- **Modal** – Serverless hosting for the email agent
- **Inngest** – Durable event queue and background job execution

11.3 Application Services

- **Clerk** – Dashboard authentication and user management
- **Composio** – OAuth connections to Gmail and Outlook for the email agent
- **Knock** – Staff notification delivery
- **Firecrawl** – Website content capture for the knowledge base
- **LlamaParse** – Document parsing for the knowledge base
- **Sentry** – Error monitoring

For a voice-call-specific view of provider touch points (what data is processed where), see **Section 10.3**.

12. Regional Deployment Model

Sunday operates with primary infrastructure deployed in a single region by default (to minimize operational complexity). Today, Sunday's primary deployment runs in us-east for Railway (application + Postgres) and Turbopuffer (vector database). The voice runtime runs on LiveKit Cloud, and edge workers run on Cloudflare's global network.

Regional Strategy:

- All components are designed to be region-replicable.
- New regions can be provisioned with replicated control-plane services and data stores.
- Production and staging run in fully separate environments for the voice runtime, event queue, and vector database.

Multi-region active-active deployments are not enabled by default but can be supported based on customer requirements.

13. Security Architecture

13.1 Data Protection

This section describes the controls that reduce the risk of unauthorized access or cross-tenant data exposure.

- **Encryption in transit:** TLS is used for network communication between browsers, agent runtimes, Sunday APIs, and third-party providers.
- **Encryption at rest:** Persistent data stores (e.g., PostgreSQL, object storage, vector stores) use cloud-provider-managed encryption at rest.
- **Application-level encryption:** Customer integration secrets (booking platform credentials, tool tokens, call-import credentials) are additionally encrypted with AES-256-GCM before being written to the database, and are decrypted only inside the control plane at the moment of use.
- **Secret management:** Platform secrets are stored in encrypted secret managers; edge worker credentials are held in Cloudflare Secrets Store.
- **Data minimization:** Only the data required to execute a request is sent to providers (see **Section 10.3**). Knowledge base embeddings are computed on Sunday's infrastructure. Agent runtimes never receive customer credentials.

13.2 Tool Execution Security

- **Per-customer credentials:** Each customer has unique credentials for each integration; credentials are never shared across customers.

- **Credentials stay in the control plane:** Account actions are executed by the Sunday API, not by the agent runtimes, so booking credentials are never exposed to a model or a conversation.
- **Tenant boundary enforcement:** Every agent action is checked against the agent's customer before it runs. Edge workers are routed per customer subdomain.
- **Isolated execution:** Each call runs in its own agent job. Customer edge integrations run in isolated execution environments (Cloudflare Workers for Platforms).
- **Deterministic tool execution:** Account-changing actions run in ordinary code after the required conversational steps succeed, not at the model's discretion (see **Section 3.3**).
- **Draft-only email:** The email agent has no ability to send email (see **Section 4.3**).

13.3 Access Controls and Auditability

- **Authentication:** Dashboard users authenticate through Clerk. Requests are verified server-side and accepted only from approved origins.
 - **Role-based access:** Users belong to a team and hold a role (front desk, management). Settings, integrations, and knowledge base management require the management role.
 - **Least privilege:** Production access is limited to required roles and is reviewed periodically.
 - **MFA:** Multi-factor authentication is required for production access.
 - **Audit trail:** Email draft lifecycle events and agent actions are logged with correlation identifiers for review.
-

14. Compliance and Operational Maturity

Sunday is in the process of completing SOC 2 Type I and II compliance using Vanta.

14.1 What SOC 2 means

SOC 2 is an independent audit report that evaluates whether an organization's controls are designed and operating effectively to protect customer data. It's commonly used by enterprises to assess vendor risk.

- **Trust Service Criteria (TSC):** SOC 2 evaluates controls across one or more categories such as **Security, Availability, Confidentiality, Processing Integrity, and Privacy.**
- **Type I vs Type II:** **Type I** attests that controls are designed appropriately at a point in time. **Type II** attests that controls operated effectively over a period of time.
- **How this applies to Sunday:** Sunday is pursuing SOC 2, and Sunday also prefers vendors with SOC 2 (or equivalent) assurance when they process customer data. Reports can typically be shared under NDA during due diligence.

Operational Controls Include:

- Access reviews and least-privilege enforcement
- Change management and deployment tracking
- Incident response documentation
- Vendor security assessments

14.2 Agent Evaluation and Testing

Changes to agent behavior are tested before they reach customers.

- **Voice flow evaluations:** A simulated caller runs full conversations against the real production prompts for each customer, covering booking, cancellations, freezes, knowledge questions, and safety cases (emergencies, off-topic requests, adversarial callers). An independent model judges each conversation for accuracy, relevance, safety, and task completion. Only state-changing APIs are mocked.
- **Email agent evaluations:** The email agent runs inside a sealed, read-only sandbox against a simulated inbox. A deterministic verifier checks the draft and labels, confirms nothing was sent, and confirms the agent never used a tool it was not given.
- **Continuous integration:** Unit tests and database migration checks run automatically on every change to the backend, agents, and dashboard.
- **Staged rollout:** Every change deploys to a staging environment, with separate voice runtime, event queue, and vector database, before production.

14.3 Observability

- **Error monitoring:** Errors across the dashboard, API, and both agent runtimes are reported to Sentry.

- **Tracing and metrics:** The API and voice agent emit OpenTelemetry traces and resource metrics, including spans for speech synthesis and call transfers.
 - **Structured logging:** Log lines carry customer, conversation, and run identifiers so any incident can be traced end to end.
 - **On-call playbook:** A written runbook maps symptoms to dashboards, provider status pages, and debugging steps.
-

15. Founders

Ryan Junejo (CEO)

Ryan Junejo is the CEO of Sunday, leading the company's strategy and execution, with a background spanning applied ML, model evaluation, and data quality for production LLM systems. At **Cohere**, he worked on data quality and evaluation processes used to refine large language models, including benchmarking against peer systems to improve reliability on computational and coding-based tasks. At the **Vector Institute**, he developed Retrieval-Augmented Generation (RAG) approaches and evaluation pipelines for taxonomy and species classification in collaboration with domain experts. His work centers on building and evaluating AI/LLM systems with an emphasis on reliability, safety, and measurable performance.

Mathusan Selvarajah (CTO)

Mathusan Selvarajah is the CTO of Sunday, leading the company's core architecture, infrastructure, and reliability. At **Google**, he helped launch **Firebase App Hosting**, a serverless platform for deploying full-stack web applications, and supported its path from public preview (Google I/O 2024) to general availability (Google Cloud Next 2025). Previously, he worked on the **Traffic** team at **Shopify**, responsible for the edge systems that route, secure, and accelerate essentially all incoming traffic to Shopify, including performance under large spikes and protections against attacks such as DDoS; earlier, he contributed to **Kubernetes**, working on core components of the open-source platform widely used to run production systems at scale.

16. Closing Note

This architecture is designed to balance reliability, security, and speed of iteration while providing clear operational boundaries suitable for enterprise technical review. Since Version 1, Sunday has taken direct ownership of its voice runtime, added a draft-only email agent, and built a member intelligence layer on top of customers' booking data, while keeping the same core principles: real-time paths stay fast, account actions stay deterministic, and customer data stays isolated.