ChurnGuard AI

Product requirements document

Owner	Tabassum Khanum
Status	Draft - v1
Last updated	July 10, 2026
Related	ChurnGuard AI Case Study (companion document)

---

1. Overview

ChurnGuard AI is a retention system that detects at-risk SaaS customers from product usage signals and automatically runs a multi-channel rescue sequence: an AI voice call, then WhatsApp, then email, then a human escalation, without a CS team having to notice or initiate anything manually. Six n8n workflows orchestrate the sequence across Vapi, Supabase, Twilio, Slack, Gmail, and Google Sheets/Calendar, with a Next.js dashboard as the team-facing surface.

In scope for v1: automated detection, voice-led rescue with multichannel fallback, human escalation with full context handoff, and a team-facing dashboard. Out of scope for v1 is anything requiring multi-tenant support, real-time (sub-hour) detection, or a custom telephony layer; see Non-goals below.

2. Problem statement

Retention teams typically operate reactively: churn dashboards surface risk, but turning that risk into a personalized, timely intervention is a manual process that only scales to a handful of high-touch accounts. By the time a human notices and reaches out, the customer has often already mentally checked out. The opportunity is to compress the gap between signal and intervention from days to minutes, and to make the intervention feel like genuine support rather than a retention script.

3. Goals and non-goals

Goals

- Detect churn risk automatically from usage signals, with no manual review step.
- Initiate a personalized voice intervention within one detection cycle (6 hours) of a customer crossing the risk threshold.
- Fall back gracefully across channels: voice, then WhatsApp, then email, then human, so no flagged customer goes untouched.
- Give a CS lead or founder a live view of what's working without reading raw call logs.

- Log every touchpoint for auditability and post-hoc pattern analysis.
- Keep per-cycle spend predictable and bounded, so cost doesn't scale unpredictably with the size of the at-risk pool.

Non-goals

- Not a replacement for a full CRM or ticketing system; the support_tickets table is intentionally minimal.
- Not real-time (sub-minute) detection; the 6-hour cycle is a deliberate cost and noise tradeoff.
- Not building a custom telephony stack; Vapi is the voice layer, not something ChurnGuard reimplements.
- No self-serve multi-tenant support in v1; single company/workspace.
- No automated discounting or contract changes; the agent can offer support, not commercial concessions.

4. Target users

- **CS / retention lead** - needs to know which accounts need a human today, not read every transcript. Primary consumer of the dashboard and the escalation Slack alerts.
- **Founder / exec** - wants a weekly signal on retention health, not a live feed of every event. Primary consumer of the weekly digest email.
- **The customer** - the end recipient of the call or message; experience quality here is the real product, since a call that feels invasive undermines the entire system's purpose.

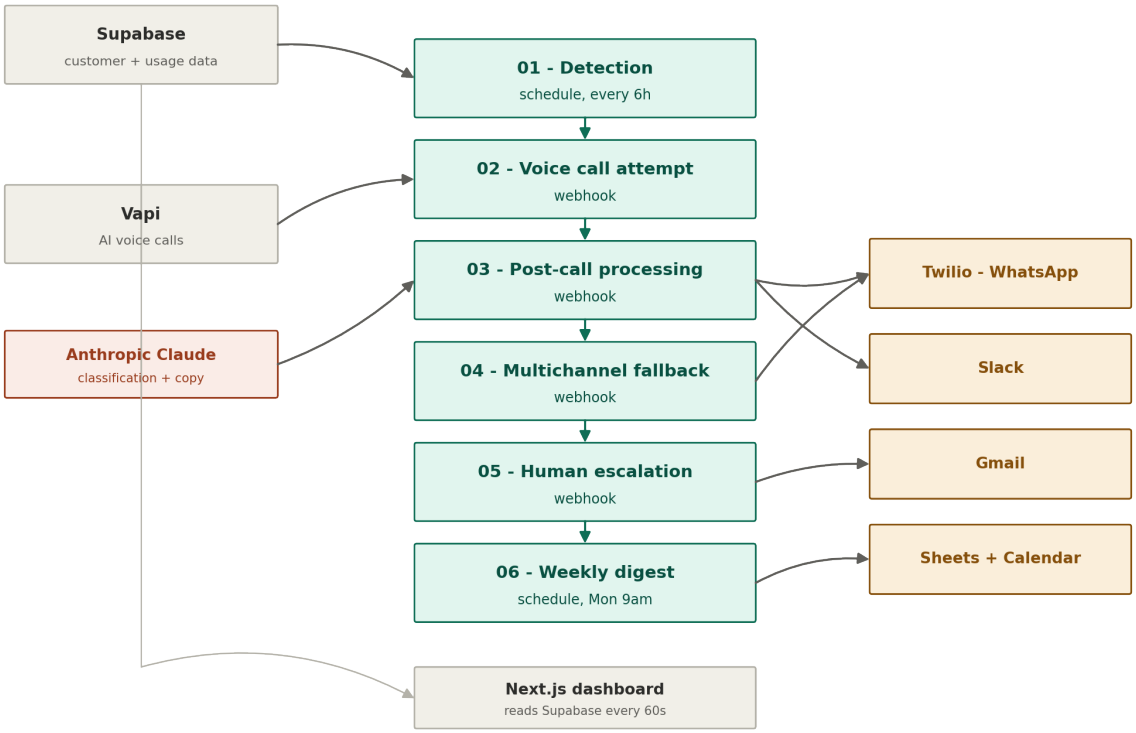
5. User stories

- As a retention lead, I want at-risk customers contacted automatically so I don't have to manually monitor a spreadsheet.
- As a retention lead, I want to be pulled in only when the AI can't resolve something, so my time goes to genuinely hard cases.
- As a retention lead, I want every escalation to arrive with a summary and context, so I'm not starting the conversation cold.
- As a founder, I want a weekly narrative summary of retention performance so I can spot trends without digging into the dashboard.
- As a founder, I want to see which usage signals are actually predictive, so the team doesn't waste calls on false positives.
- As a customer, I want the outreach to feel like real support, not a retention script, so I don't feel tracked or sold to.
- As a customer whose issue isn't resolved on the call, I want a human to follow up within a clear timeframe, so I'm not left waiting indefinitely.

6. System architecture

n8n is the orchestration layer between the Next.js app and every external service. It was chosen over custom backend code because the hard parts of this system, branching by outcome, retries, waits, and OAuth against five different services, are what a workflow tool solves out of the box, with a visual execution log that doubles as a debugging and demo surface.

Two workflows are schedule-triggered (Detection and the Weekly digest); the other four are webhook-triggered and chain off each other or off an external event, such as a Vapi call ending or the voice agent calling its escalate_to_human tool. Schedule-triggered workflows are self-contained and easy to test in isolation, while the webhook chain has to be defensive: every webhook receiver returns success immediately and does its real work asynchronously, so a slow downstream step never causes an upstream retry storm.



Six workflows: two schedule-triggered, four webhook-triggered

7. Functional requirements

7.1 Detection (Workflow 1)

Trigger	Schedule, every 6 hours.

Logic	Scores all customers via the churn-detection endpoint, filters to high/critical risk not contacted in the last 48 hours, caps the run at 5 customers.
Output	Triggers Workflow 2 once per qualifying customer, spaced 90 seconds apart.
Edge cases	Skips its own run if the previous execution is still in progress, so overlapping schedule triggers can't double-flag the same customer.

7.2 Voice call attempt (Workflow 2)

Trigger	Webhook, from Workflow 1.
Logic	Pulls the customer profile, 30-day usage, and open tickets; computes days since login, usage trend, and unused features; places an outbound Vapi call with those as live prompt variables.
Output	Logs the attempt, updates last_contacted_at, and triggers Workflow 4 on no-answer or voicemail.
Key fields	assistantId, phoneNumberId, customer.number, assistantOverrides.variableValues including customer_name, company, plan, days_since_login, usage_trend, open_tickets, unused_features.

7.3 Post-call processing (Workflow 3)

Trigger	Webhook, forwarded from Vapi's end-of-call report via a Vercel proxy endpoint.
Logic	Classifies the transcript for outcome, sentiment, friction points, and a one-line summary via an Anthropic-backed LLM chain node, updates the call record, and adjusts the customer's health score.
Output	Routes by outcome: Sheets + Slack (saved), Workflow 5 (escalated), Sheets + Gmail autopsy (churned), 3-day retry via Workflow 2 (pending).
Edge cases	Health-score delta is symmetric and clamped to 0-100. A pending outcome retries once via Workflow 2 after 3 days rather than looping indefinitely.

7.4 Multichannel fallback (Workflow 4)

Trigger	Webhook, from Workflow 2 on no-answer or voicemail.

Logic	Sends a personalized WhatsApp check-in via Twilio; waits 24 hours; if there's still no login activity, sends a follow-up email.
Output	Touchpoint logged for every channel attempt.

7.5 Human escalation (Workflow 5)

Trigger	Webhook, from Workflow 3's escalated branch, or directly from the voice agent's escalate_to_human tool mid-call.
Logic	Summarizes the situation in 3 bullets, posts to #customer-success, books a follow-up on the CS calendar, sends the customer a confirmation email.
Output	Sets escalated_at on the customer record; logs touchpoints for Slack and email.
Edge cases	Can be triggered twice for the same customer (live escalation, then a second escalated outcome from Workflow 3); escalated_at should not be overwritten if already set within the same day.

7.6 Weekly digest (Workflow 6)

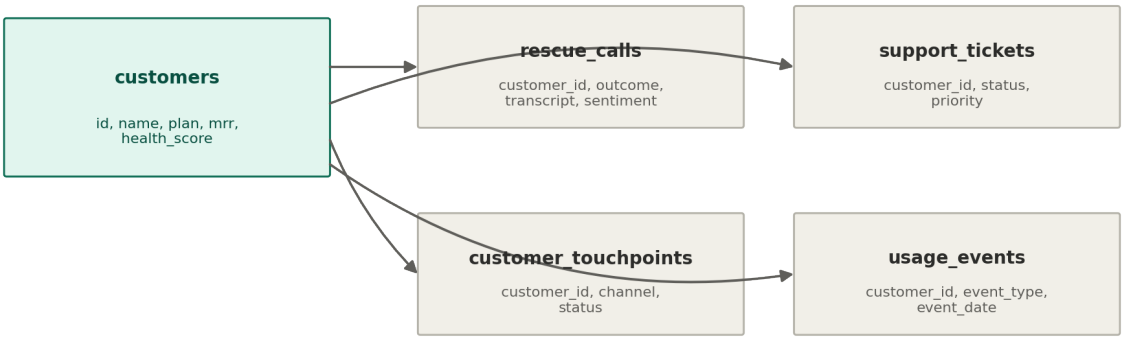
Trigger	Schedule, Monday 9am.
Logic	Pulls 7-day dashboard stats and writes a narrative summary.
Output	Emails the founder; appends a row to the Executive Metrics sheet.

7.7 Dashboard

Trigger	Auto-refresh, every 60 seconds.
Logic	Aggregates from a single Supabase endpoint rather than querying per-widget.
Output	Surfaces at-risk count, save rate, revenue protected, rescue funnel, signal precision, friction points, channel performance, and cohort comparison.

8. Data model

Five core tables, all keyed off customer_id.



Core tables and their relationships

customers

id, name, phone, email, company, plan, mrr, health_score, last_contacted_at, escalated_at

rescue_calls

id, customer_id, vapi_call_id, call_status, call_duration, transcript, outcome, sentiment, friction_points, solution_offered, created_at

customer_touchpoints

id, customer_id, channel (voice/whatsapp/email/slack), direction, content, status, metadata, created_at

support_tickets

id, customer_id, subject, priority, status, created_at

usage_events

id, customer_id, event_type, feature, event_date

9. Non-functional requirements

Reliability	Every webhook receiver (end-of-call, escalation) returns success immediately and processes asynchronously, so a slow or failed downstream step never causes the upstream caller (Vapi) to retry and double-process.

Security	Calls from n8n back into Next.js tool endpoints are authenticated with a shared header secret (x-api-secret / INTERNAL_API_SECRET), not left open.
Performance	Dashboard aggregation happens in Supabase, not in-memory in the API route, since the endpoint is polled every 60 seconds by every open dashboard tab.
Cost control	Call volume is hard-capped at 5 per detection cycle; LLM classification runs on Haiku, not a larger model, to keep per-call cost predictable.
Auditability	Every customer-facing action, across every channel, is written to customer_touchpoints, so nothing the system does to a customer goes untracked.

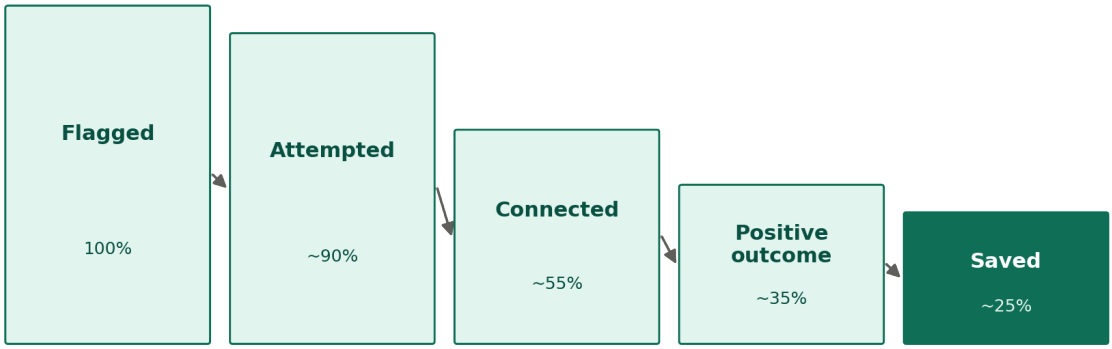
10. Assumptions and dependencies

Service	Used for	Setup needed
Vapi	Outbound AI voice calls and the end-of-call webhook.	Account, assistant config, phone number.
Twilio	WhatsApp fallback messages.	Trial account, WhatsApp sandbox activation.
Slack	Escalation alerts and save celebrations.	Workspace with #customer-success, #wins, #churn-autopsy channels.
Google Sheets / Calendar	Metrics logging and escalation follow-up bookings.	OAuth via n8n; no separate setup.
Gmail	Churn autopsy emails and customer confirmations.	OAuth via n8n; no separate setup.
Anthropic	Call classification and generated copy.	API key.
Supabase	Primary data store and RAG knowledge base.	Project, schema, service role key.

11. Success metrics

The rescue funnel is the primary health metric for the system. Figures below are targets used to size the build, not measured results:

Target rescue funnel (illustrative example)



- **Call answer rate** - percentage of attempted calls that connect.
- **Connect-to-save conversion** - of connected calls, what fraction resolve as saved.
- **Revenue protected** - summed MRR of customers with a saved outcome in the period.
- **Signal precision** - of customers flagged by each individual risk signal, what fraction actually churn or need to be saved; used to prune noisy signals over time.
- **Channel response rate** - WhatsApp and email response rates, tracked separately to validate the fallback ordering.

12. Rollout plan

Phase	Scope	Checkpoint
0-3	Project setup, Supabase schema, seed data, churn detection API, RAG knowledge base.	Detection API returns ranked risk list.
4-6	Vapi tool endpoints, Vercel deploy, Vapi assistant config, post-call webhook receiver.	First successful test call end to end.
7	n8n environment setup: service accounts, credentials, folder structure.	All credentials connected in n8n.
8	Build and activate Workflows 1-3 (core rescue flow).	Full test call: outcome, transcript, and health score all update correctly.
9	Build and activate Workflows 4-5 (fallback and escalation).	Escalation test: Slack, calendar, and email all fire from a live call.
10	Build and activate Workflow 6 (weekly digest).	First Monday digest email received.
11-12	Dashboard, landing page, README and demo assets, launch.	Dashboard auto-refreshes with live data; demo script rehearsed.

13. Risks and mitigations

Cost overrun	Vapi/LLM spend scales with call volume; mitigated by the 5-call cap per detection cycle and credit tracking during build.
Feels invasive	Calling based on private usage data can read as surveillance; mitigated by the no-analytics-mention rule and 48-hour cooldown.
Duplicate processing	n8n webhook failures could cause Vapi to retry and double-process a call; mitigated by returning 200 immediately from the Vercel proxy before n8n finishes.
False positives	An overly broad risk filter wastes calls and annoys healthy customers; mitigated by the signal-precision dashboard panel, used to prune weak signals.

Voice quality / latency	A slow or robotic-sounding agent undermines the whole premise of a genuine check-in; mitigated by tuning the Vapi prompt against real call recordings before wider rollout.
Webhook spoofing	The post-call and escalation webhooks are public URLs, not yet protected by a shared secret; a spoofed request could write false call outcomes.
Single point of failure	An n8n outage halts all 6 workflows at once; not yet mitigated, see open questions below.

14. Open questions

- Should multi-tenant support be a v2 requirement, or should ChurnGuard stay single-workspace?
- What's the acceptable false-positive rate before the risk filter needs retuning, and who owns that tuning?
- Should the escalation SLA (currently within one business day) be configurable per plan tier?
- Does an n8n outage need a failover path, or is manual restart an acceptable v1 risk?
- Should the public post-call and escalation webhooks be signed or secret-protected before this handles real customer data?

Appendix: environment variables

N8N_ESCALATION_WEBHOOK	Workflow 5's webhook URL; called from the escalate_to_human tool endpoint.
N8N_POST_CALL_WEBHOOK	Workflow 3's webhook URL; called from the end-of-call proxy endpoint.
VAPI_API_KEY	Server-side key for creating outbound calls via api.vapi.ai.
VAPI_ASSISTANT_ID / VAPI_PHONE_NUMBER_ID	Identify which assistant and outbound number Workflow 2 uses.
TWILIO_ACCOUNT_SID / TWILIO_AUTH_TOKEN / TWILIO_WHATSAPP_NUMBER	WhatsApp send credentials for Workflow 4.

ANTHROPIC_API_KEY	Used by n8n's Anthropic Chat Model node for classification and generated copy.
INTERNAL_API_SECRET	Shared secret n8n sends as x-api-secret when calling back into Next.js endpoints.