AI Front Desk Agent — Project Roadmap (Restaurants)

Owner: Karson
Goal: Ship a production-ready virtual front desk that answers FAQs, takes reservations without double-booking, handles takeout orders, and offers smooth human handoff.
Initial hosting: Self-hosted middleware on Karson’s PC; AI model via OpenAI API.

---

1) Product Definition

1.1 Problem & Value

- Restaurants miss calls and tie up staff answering repetitive questions.
- Agent handles calls/web chats 24/7, reducing burden and capturing more bookings & orders.

1.2 Scope (MVP)

- **Voice/Phone**: Answer calls on the restaurant’s main number and complete flows.
- **Reservations**: Create/modify/cancel; enforce capacity, party size limits, blackout times.
- **Orders (Takeout)**: Collect items, upsell, confirm pricing/taxes, estimate pickup time, take payment (phase 2).
- **FAQs**: Hours, location, menu, dietary info, parking, wait times, specials.
- **Human Handoff**: “Talk to a person” routes to staff line or voicemail with transcript.
- **Website Option**: If no website, provide a simple site with the same capabilities.

1.3 Non-Goals (for now)

- Delivery logistics, table assignment by section, loyalty points, social-media DMs.

1.4 Success Criteria (MVP)

- ≥ 95% calls answered by agent within 2 rings.
- ≥ 98% reservation conflict-free (no double bookings).
- ≤ 1% abandoned order flows after payment handoff (when payment added).
- CSAT (owner feedback) ≥ 4.5/5 first month.

---

2) User Journeys

Call-in Reservation

Caller → Twilio IVR → STT → GPT dialogue → capacity check → create resv → confirmation SMS/email → CRM/DB update → owner notification (optional).

Website Reservation

Visitor → Web widget → GPT dialogue (text/voice) → capacity check → create resv → confirmation → DB update.

Takeout Order

Caller/Visitor → Menu Q&A → item capture → modifiers → subtotal/taxes → pickup ETA → order record → payment link (phase 2) → confirmation.

FAQ

Caller/Visitor → Ask → Answer from knowledge base (menu/hours/policies) → (optional) escalate to human.

Human Handoff

Caller says "agent → human" or agent low confidence → warm transfer to staff number; if no answer, smart voicemail + transcript + contact info.

---

3) System Architecture

Core - **Conversational AI**: OpenAI GPT (tools: function calling; system prompts per tenant).
- **Telephony**: Twilio Programmable Voice + TwiML webhook; SIP trunk or call-forward from main number.
- **Speech**: STT/TTS via OpenAI (or compatible) for real-time voice; latency target < 400 ms turn-around.
- **Middleware (Self-hosted)**: Python **FastAPI** service orchestrating flows and webhooks.
- **Data**: PostgreSQL for relational (reservations, inventory, hours, blackout rules), Redis for caching/rate-limit/session.
- **Static Site / Dashboard**: React front-end served by the backend (or Vite build + Nginx).
- **Auth**: Owner portal with email+OTP; signed JWT for API.

Services/Modules - **Tenant Manager**: Per-restaurant config (hours, capacity, party size rules, menu, tax rate, pickup buffers, handoff number).
- **Reservation Engine**: Availability search, capacity ledger, conflict detection, hold & commit, modify/cancel.
- **Order Engine**: Menu schema, modifiers, taxes, pickup ETA logic, (phase 2: payments via Stripe).
- **FAQ/KB**: Owner-editable facts; vector search optional in v2; MVP uses structured fields + curated Q/A.
- **Handoff Router**: Transfer logic, fallback voicemail, transcript email/SMS.
- **Notifications**: SMS/Email confirmations, daily digest to owner, error alerts.
- **Analytics/Feedback**: Call outcomes, reservation funnel, order conversion; thumbs-up/down after calls/web.

Integrations (optional per tenant) - Google Business Messages (later), calendar export (.ics), POS webhook (later), Stripe (phase 2).

---

4) Data Model (MVP)

Restaurant(id, name, phone, timezone, address, handoff_number, locale_default, …)

HoursRule(restaurant_id, day_of_week, open_time, close_time)

Blackout(restaurant_id, start_ts, end_ts, reason)

CapacityRule(restaurant_id, start_ts, end_ts, max_covers, max_parties, party_min, party_max)

Reservation(id, restaurant_id, name, party_size, start_ts, end_ts, status[pending|confirmed|cancelled], source[phone|web|staff], contact{phone,email}, notes)

Order(id, restaurant_id, items_json, subtotal, taxes, total, pickup_eta_ts, status[pending|confirmed|ready|cancelled], contact)

MenuItem(id, restaurant_id, name, description, price, active, modifiers_json)

EventLog(ts, restaurant_id, type[call|reservation|order|handoff|error], payload_json)

Feedback(id, restaurant_id, channel[phone|web], rating, comment, contact)

---

5) Reservation Logic (No Double-Booking)

1. **Availability Query**: derive slots from HoursRule ∩ CapacityRule.
2. **Soft Hold (5 min)** when agent proposes a time; stored in Redis with (party_size, slot).
3. **Conflict Check**: `(sum(party_size where overlap) <= max_covers)` and `(count(parties where overlap) <= max_parties)`.
4. **Commit** on confirmation → persist Reservation; clear hold.
5. **Modify/Cancel** updates ledger and frees capacity.
6. **Overbook Guard**: transaction + unique constraint on (restaurant_id, slot_id, shard) with retry.

---

6) Takeout Order Flow (MVP)

- Load active menu, modifiers, and taxes.
- Parse natural language to items/modifiers; repeat order; confirm subtotal and ETA.
- Create Order record; send SMS/email confirmation.
- **Phase 2**: Stripe Payment Link (card on web) during call; reconcile payment webhooks.

---

7) FAQ/Knowledge

- Structured config: hours, parking, dress code, allergy guidance, kid-friendly, alcohol policy, patio, etc.
- Owner UI to edit.
- Agent retrieves facts deterministically (no hallucinations) before fallback to GPT generalization.

---

8) Human Handoff

- Keywords or low-confidence → **warm transfer** to `handoff_number`.
- If no answer: record voicemail, transcribe, email/SMS transcript + callback link.
- Owner can set business hours for auto-voicemail vs. ring-through.

---

9) Onboarding & Implementation

Steps per restaurant 1) Intake form (hours, capacity, menu, policies, preferred language(s), handoff number).
2) Phone setup: port number to Twilio **or** forward existing number to Twilio; verify CNAM.
3) Configure agent: prompts, tools, guardrails, upsell toggles.
4) Data import: existing reservations (CSV) → DB; seed menu.
5) Dry-run in staging number; owner sign-off.
6) Go-live: cutover calls; enable web widget/site.

Website (if needed) - Deploy minimal site: Home, Menu, Reservation/Order widget, Contact/Hours.
- White-label template; per-tenant branding.

---

10) Hosting & Deployment (Self-Hosted Middleware)

- **Runtime**: Python 3.11+, FastAPI, Uvicorn workers, Redis, PostgreSQL.
- **Process Manager**: systemd user services.
- **Reverse Proxy**: Nginx with TLS (Let's Encrypt).
- **Config**: `.env` for secrets; rotate keys; IP allow-list Twilio webhooks.

Environments - `dev` (localhost), `stage` (staging phone number), `prod` (live).
- Separate Twilio numbers per env; separate DB schemas.

Scaling Path - Vertical first (more workers), then light sharding per restaurant.
- If >5–10 tenants with high concurrency → consider cloud VM migration later.

---

11) Monitoring, Logs, & Alerts

- Structured logs (JSON) for all webhook calls; per-call correlation IDs.

- Health endpoints (`/healthz`, `/readiness`).
- Error budgets & alerting: webhook failures, TTS/STT latency spikes, reservation conflicts, payment errors.
- Nightly owner digest: yesterday's calls, reservations, orders, missed handoffs.

---

12) Feedback & Updates

- **Feedback capture**: post-call SMS link (1–3 question micro-survey) and owner portal thumbs-up/down per interaction.
- **Change comms**: email changelog for bugfixes; **paid upgrades** packaged and announced with opt-in.
- **A/B learnings**: store anonymized flow metrics to refine prompts and guardrails.

---

13) Security & Privacy Baseline

- PII minimization; encrypt at rest (Postgres TDE or disk LUKS) and in transit (TLS).
- Secret management (.env + restricted perms); audit logs for admin actions.
- Rate limiting & abuse prevention on web widget.
- Data retention policy (e.g., 18 months by default); export/delete on owner request.
- DNC and consent handling for SMS.

---

14) Internationalization (Roadmap)

- Agent locale per restaurant; bilingual prompts; time/date/number formatting per locale.
- Separate translatable strings in UI; fallback rules.
- Voice: pick TTS voice per language; STT model selection per locale.

---

15) Testing Plan

- **Unit**: reservation conflict logic; order parsing; time-zone math.
- **Integration**: Twilio → webhook → STT/TTS → GPT tool calls → DB writes.
- **Load**: simulate concurrent calls (e.g., lunch rush).
- **UAT**: scripted call scenarios with owner.

---

16) Milestones & Deliverables

M0 — Foundations (Week 1)
- Repo, FastAPI skeleton, Postgres schema, Redis session store, .env wiring, health endpoints.

M1 — Reservations (Week 2)
- Reservation engine (ledger + holds + commit), Twilio voice path, confirmation SMS/email.

M2 — FAQs (Week 3)
- Owner KB UI + deterministic retrieval; fallback to GPT.

M3 — Takeout Orders (Week 4)
- Menu CRUD, order capture, ETA; confirmation. (Stripe in Phase 2.)

M4 — Human Handoff (Week 5)
- Warm transfer + voicemail + transcription; owner notifications.

M5 — Website (Week 6)
- Minimal site + embedded widget; theming/branding per tenant.

M6 — Monitoring & Feedback (Week 7)
- Logs, alerts, analytics dashboard; micro-survey links.

M7 — Pilot & Hardening (Week 8)
- Onboard first restaurant; burn-down bug list; finalize docs.

---

17) Operability Runbooks

- **Incident**: Telephony webhook errors → retry policy; contact Twilio status; fail open to human handoff.
- **Latency**: STT/TTS spikes → switch to backup voices/models; degrade to DTMF menu if needed.
- **Data**: reservation conflict detected post-factum → auto-notify owner + propose resolution window.
- **Secrets**: key rotation steps; revoke + reissue.

---

18) Owner & Staff Docs (MVP)

- Quick-start (phone cutover, portal login).
- Edit hours/menu/blackouts.
- Review reservations/orders.
- Handle handoffs/voicemail.
- FAQ editing best practices.

---

19) Phase 2 Candidates

- Online payments (Stripe).
- POS integration (webhook adapters).
- Calendar sync (Google/Outlook).

• Vector KB + RAG for richer menu/FAQ.
• Multilingual voice packs and locale auto-detect.
• Cloud migration template (IaC) for scaling.

---

20) Acceptance Checklist (Go-Live per Restaurant)

• [ ] All hours/blackouts configured and validated.
• [ ] Capacity rules provide safe headroom (≥10% buffer).
• [ ] Test calls: new reservation, modify, cancel, FAQ, order flow, and human handoff.
• [ ] Confirmation SMS/email verified.
• [ ] Owner portal access verified; staff trained.
• [ ] Monitoring alerts configured; daily digest delivered.
• [ ] Backup number & voicemail transcription tested.