Below is a locked-down “Gold’s Gym agent-in-a-box” blueprint: what it does, what it is *not allowed* to do, what permissions it needs, and an MVP you can ship fast.

The product promise

“Install → connect inbox/phone/calendar → it follows up with leads, books appointments, and keeps staff on script—without exposing the business.”

That means:
• No public skills marketplace.
• No arbitrary code execution.
• No “paste this command.”
• Every action is allowlisted, logged, and reversible.

Core workflows (the 8 that matter most)

These are the “skills,” but you ship them as **built-in modules**:
1. **Lead intake + enrichment**
• Inputs: website form, Facebook/IG leads, call log, walk-in notes.
• Output: creates a lead record + tags (goal: weight loss, strength, tanning, etc.)
2. **Instant follow-up (SMS + email)**
• Sends a 60–120 second response automatically during business hours.
• Uses a short script with 1–2 questions and a CTA to book.
3. **Appointment booking**
• Offers 2–3 time slots based on staff calendar availability.
• Books “tour / intro / consult” on calendar, sends confirmation + reminder.
4. **No-show recovery**
• If they no-show, sends a reschedule message and notifies staff.
• If they reschedule, updates calendar automatically.
5. **Lead nurturing sequences**
• Day 0, 1, 3, 7, 14 touchpoints with controlled copy.
• Stops automatically if they book, join, or opt out.
6. **Member retention saves**
• Detects risk signals: missed check-ins, billing fail, cancellation intent keywords.
• Triggers outreach: “Want to freeze instead?” or “Can I fix billing?”
7. **Front-desk “answer engine” (FAQ only)**
• Answers: hours, prices (if you allow it), amenities, policies.

• Escalates anything sensitive to staff.

8. **Manager reporting**

• Daily summary: new leads, contacted %, booked, show rate, close rate, churn saves.

• Weekly insights: what scripts convert, what times book best.

Locked-down action model (how you keep employers safe)

Instead of “agent can do anything,” you define **a small set of permitted actions**:

Allowed actions (example allowlist)

• Send SMS via ++Twilio++ (or similar)

• Send email via Gmail API / Microsoft Graph

• Create/update calendar events

• Read/write CRM records (your built-in CRM or ++HubSpot++ / ++Salesforce++ integration)

• Post messages into Slack (optional)

• Create tasks for humans (“Call this lead”)

• Generate reports

Disallowed by default

• Shell/terminal execution

• Installing/downloading third-party code

• Reading arbitrary files outside an allowlisted folder

• Browser automation on arbitrary sites

• Accessing payroll, bank accounts, or admin consoles

• Changing pricing, contracts, or billing without human approval

Human-in-the-loop gates (simple, employer-friendly)

• “Send offer/discount” → requires manager approval

• “Cancel membership / freeze” → requires manager approval

• “Refund / charge” → always human

Permissions you’ll request (minimal and explainable)

For a gym “work-only” deployment, this is the clean set:
• **Messaging**: SMS number + email sending identity
• **Calendar**: read availability + create events
• **Lead source**: web form / leads inbox
• **CRM store**: your local database or existing CRM
• **Optional**: knowledge base (policies, hours, amenities) as a locked doc

Nothing else.

The MVP you should ship first (2-week realistic)

You want “magical” without risk.

MVP = Lead follow-up + booking + reporting
• Lead comes in → agent texts within 60s
• Agent asks 1 question + offers 2 times
• Books appointment on calendar
• Sends reminders + no-show recovery
• Daily manager summary

That alone will sell, because it directly increases **speed-to-lead** and **show rate**.

App experience (what the lay person sees)
1. Install app
2. “Connect your work phone/email/calendar”
3. Pick your gym’s hours + staff availability
4. Choose one of 3 scripts (Tone A/B/C)
5. Turn on “Auto-follow-up”
6. Done

Everything else is hidden until needed.

Business model that gyms accept
• **Per location monthly** (simplest): $199–$499/location depending on volume
• Add-ons:
• Extra SMS numbers

• Multi-location dashboard
• Retention module
• CRM integration

What you should build before anything else

These are your “enterprise trust” features that make employers say yes:
• **Audit log**: every message, booking, and data access
• **Role-based access**: staff vs manager
• **Data boundaries**: location-level separation
• **Kill switch**: one toggle to stop all automation instantly
• **Message preview mode** (optional): first week “approve before send,” then autopilot

Why this is a strong wedge for ++Gold’s Gym++ specifically

Gyms live and die by:
• speed to lead
• appointment show rate
• follow-up consistency
• cancellations that could’ve been saved

An agent that enforces those behaviors (without giving it dangerous powers) is exactly what a GM wants.

Below is a **hand-off spec** you can give straight to Codex: data model, event triggers, allowlisted actions, policies, UI screens, scripts, logging, and a tight implementation plan.

After the spec, there’s a **one-shot Codex prompt** that tells it to build ONLY the MVP and to output work as **small diffs/patches** (your preference).

———

Product: “Gym Agent-in-a-Box” V1 (MVP Spec)

0) V1 scope (what exists / what doesn’t)

V1 must do

• Ingest new leads (manual entry + webhook endpoint for a website form)
• Auto-follow-up by SMS within business hours (or schedule next open time)
• 1-question qualification + propose 2–3 booking slots
• Book appointment on shared calendar (or internal calendar in-app)
• Confirmation + reminders + no-show recovery
• Daily manager summary dashboard + exportable CSV
• Full audit log of all actions

V1 must not do

• No marketplace / no external skills installation
• No shell execution
• No arbitrary browsing/automation on websites
• No member billing changes, refunds, contract edits
• No reading arbitrary local files (only app config + optional FAQ text pasted into UI)

———

1) Architecture (simple, shippable, locked down)

Runtime

• **Local desktop app** (macOS-first) with a **local backend**:
• UI: SwiftUI (macOS) *or* Tauri/Electron (cross-platform)
• Backend: Node/Express or Python/FastAPI (pick one)
• Local DB: SQLite
• Optional “cloud” later; V1 can be fully local except for SMS + LLM calls.

Integrations (V1)

• SMS provider: Twilio (or pluggable adapter)
• Calendar: Google Calendar (OAuth) *or* internal calendar only if you want to avoid OAuth in V1
• LLM: OpenRouter / local Ollama (adapter interface; default to OpenRouter)

Hard security boundary

All external side-effects go through a single module: **ActionGateway**
• If it’s not in the allowlist, it cannot happen.
• Every action is logged before and after execution.

———

2) Data Model (SQLite)

Tables

locations
• id (pk)
• name
• timezone
• business_hours_json (Mon–Sun open/close)
• booking_buffer_minutes (default 10)
• booking_slot_minutes (default 30)
• created_at

staff
• id (pk)
• location_id (fk)
• name
• role enum: staff | manager
• calendar_id (nullable if internal calendar)
• active bool

lead
• id (pk)
• location_id (fk)
• first_name
• last_name
• phone_e164
• email (nullable)
• source enum: webform | manual | import
• status enum: new | contacted | qualified | booked | no_show | joined | lost | do_not_contact

• tags_json (e.g., [“weight_loss”,“strength”])
• notes
• created_at
• last_contact_at
• next_action_at (nullable)
• opted_out bool

conversation
• id (pk)
• lead_id (fk)
• channel enum: sms | email
• state enum: idle | awaiting_reply | awaiting_booking_choice | booked | stopped
• last_inbound_at
• last_outbound_at
• state_json (stores short-lived variables, e.g. last proposed times)

message
• id (pk)
• conversation_id (fk)
• direction enum: inbound | outbound
• channel enum: sms | email
• body
• provider_message_id (nullable)
• status enum: queued | sent | delivered | failed
• created_at

appointment
• id (pk)
• lead_id (fk)
• staff_id (fk, nullable if “front desk calendar”)
• start_at
• end_at
• calendar_event_id (nullable)
• status enum: scheduled | completed | no_show | cancelled
• created_at

event_queue
• id (pk)
• type enum: lead_created | inbound_message | scheduled_tick | appointment_no_show | daily_summary
• payload_json
• run_at
• status enum: pending | processing | done | failed
• attempts int
• last_error

audit_log
• id (pk)
• actor enum: system | user
• actor_id (nullable)
• action_type (string) (e.g., SMS_SEND, CAL_CREATE_EVENT)
• target_type (string) (e.g., lead, appointment)
• target_id (string/int)
• request_json
• response_json
• success bool
• created_at

settings
• id (pk) (single row)
• llm_provider enum: openrouter | ollama
• llm_model (string)
• sms_provider enum: twilio
• twilio_account_sid (encrypted at rest if possible)
• twilio_auth_token (encrypted)
• twilio_from_number
• google_oauth_json (encrypted, nullable)
• default_location_id
• script_variant enum: A | B | C
• autopilot_enabled bool
• quiet_hours_policy enum: send_next_open | queue_and_notify
• escalation_policy_json (keywords, staff notify rules)

———

3) Action allowlist (the only permitted side-effects)

Implement ActionGateway with strict methods:

Messaging
• send_sms(to_e164, body, lead_id, conversation_id)
• hard limits: max 480 chars; no links unless allowlisted domain; no attachments
• throttle: do not send more than N messages per lead per day (default 4)

Calendar
• create_calendar_event(calendar_id, title, start_at, end_at, description, attendees?)
• update_calendar_event(calendar_id, event_id, patch)
• cancel_calendar_event(calendar_id, event_id)

CRM / DB
• update_lead_status(lead_id, status)
• set_lead_next_action(lead_id, datetime)
• add_lead_note(lead_id, note)
• mark_opt_out(lead_id)

Notifications (optional in V1)
• notify_staff(location_id, message) (in-app notification only)

Everything routes through ActionGateway and emits an audit_log row.

———

4) Event triggers and state machine

Trigger A: Lead created

Event: lead_created
Goal: Contact within 60 seconds during business hours.

Flow:
1. If opted_out true → stop
2. If outside business hours:
• schedule scheduled_tick at next open time
• optionally notify staff “Lead queued”
3. If inside business hours:
• send initial SMS script (variant A/B/C)
• set lead status contacted
• set conversation state awaiting_reply
• set next_action_at = now + 2 hours (nurture follow-up if no reply)

Trigger B: Inbound message (SMS webhook)

Event: inbound_message
Parse inbound text:
• If contains opt-out words (“stop”, “unsubscribe”) → mark_opt_out
• If conversation.state == awaiting_reply:
• classify intent:
• booking-ready (yes/ready/sure)
• question
• not interested
• If booking-ready or question:
• ask 1 qualifier question OR proceed directly to propose times
• If state == awaiting_booking_choice:
• match a proposed time option “1”, “2”, “3” or “morning/afternoon”
• book appointment + confirm
• If unknown:
• respond with safe default: “Got it — would you like to schedule a quick tour?”

Trigger C: Scheduled tick

Event: scheduled_tick
Used for:
• “No reply” follow-ups
• Sending reminders (24h and 2h prior)
• No-show recovery

Trigger D: Appointment no-show

If appointment start passes by 15 minutes and no “completed” mark:
• mark appointment no_show
• send no-show recovery SMS
• notify staff

Trigger E: Daily summary

At close of business (or 7pm):
• compute metrics
• produce dashboard row + optional email to manager
• export CSV on demand

———

5) Messaging scripts (ship 3 variants)

All scripts must be editable in-app, but V1 can hardcode defaults.

Script A (direct, minimal)

Initial
“Hi {first_name}, this is {gym_name}. Want to come in for a quick tour this week? What’s your main goal right now—fat loss, strength, or just getting back into it?”

If goal given
“Perfect. I can get you set up. What day is best—today/tomorrow, or later this week?”

Propose times
“I have {slot1} or {slot2} available. Reply 1 or 2.”

Confirm
“Booked: {day} at {time}. Address: {address}. Reply YES to confirm or RESCHED to change.”

No reply follow-up (2h)
“Quick nudge—want me to hold a tour time for you? I can do {slot1} or {slot2}. Reply 1 or 2.”

No-show
“No worries—want to reschedule? I can do {slot1} or {slot2}. Reply 1 or 2.”

Script B (friendlier)

(Rewrite same structure; slightly warmer; still short.)

Script C (high-intent, sales)

(Emphasize limited availability + specific benefit; still compliant.)

Hard rules:
• No pricing promises in V1 unless a manager toggles “Allow pricing mention” and supplies a snippet.
• Always include opt-out compliance: if user opts out, stop immediately.

———

6) Booking logic (deterministic, safe)

Calendar availability

V1 approach:
• If Google Calendar connected:
• read availability for selected staff calendars
• propose 2–3 slots within next 3 business days
• enforce buffer and slot length
• If no calendar integration:
• internal “front desk schedule” in SQLite (simple)
• propose next open slot blocks

Slot selection

• Always offer “Reply 1 or 2” options.

• Store proposed slots in conversation.state_json with a short TTL (e.g., 6 hours).

———

7) UI screens (minimal but complete)

Screen 1: Setup Wizard

• Location name, timezone

• Business hours picker

• Connect SMS (Twilio credentials)

• Connect calendar (Google OAuth) OR choose “Internal calendar”

• Pick script variant A/B/C

• Toggle: Autopilot enabled

Screen 2: Leads Inbox

• List leads with status chips

• Quick actions: “Pause automation”, “Mark joined”, “Mark lost”

• Lead detail pane:

• conversation transcript

• manual message send box (goes through ActionGateway)

• notes + tags

Screen 3: Calendar / Appointments

• Today’s tours

• No-shows

• Quick reschedule

Screen 4: Settings / Policies

• Autopilot toggle

• Quiet hours policy

• Daily message cap per lead

• Escalation keywords list

• “Allow pricing mention” toggle + text box (manager-only)

Screen 5: Reporting
• Today: new leads, contacted %, booked, no-show rate
• Last 7 days trends (simple table)
• Export CSV

Screen 6: Audit Log (manager-only)
• Filter by action type / lead
• View request/response payloads

---

8) Safety policies (non-negotiable)
• If inbound includes opt-out keywords → stop, mark DNC, send one confirmation message max.
• If LLM is used:
• LLM may draft text, but final send must pass:
• allowlist checks (length, no forbidden content, no unauthorized claims)
• template conformance (must include gym name, no pricing unless allowed)
• Provide a global **Kill Switch**: immediately disables autopilot and cancels scheduled jobs.

---

9) Implementation plan (MVP order)
1. SQLite schema + repository layer
2. Event queue + worker loop (poll every 2–5 seconds) and scheduler
3. SMS webhook + outbound SMS adapter
4. Lead-created flow + inbound reply parsing + simple state machine
5. Booking slot proposer + calendar adapter (or internal calendar)
6. UI: Setup Wizard + Leads Inbox + Settings + Reporting
7. Audit log + CSV export
8. Hardening: rate limits, DNC, kill switch

One-shot Codex Prompt (build ONLY the MVP)

```
ROLE: You are a senior product engineer building a locked-down
“Gym Agent-in-a-Box” MVP.

GOAL: Build a working V1 desktop app + local backend that:
- Accepts new leads (manual UI + /webhook/lead endpoint)
- Auto-texts new leads within business hours (or schedules next
open time)
- Handles inbound SMS replies via webhook and books an appointment
(Google Calendar if configured; otherwise internal calendar)
- Sends confirmation + reminders + no-show recovery
- Shows a minimal UI: Setup Wizard, Leads Inbox w/ transcript,
Settings, Reporting, Audit Log
- Enforces a strict allowlist of actions and logs every action to
audit_log

HARD CONSTRAINTS (must follow):
- DO NOT add any plugin/marketplace system.
- DO NOT execute shell commands or run arbitrary code.
- DO NOT add features beyond the MVP scope.
- DO NOT change UI style beyond minimal readable layout.
- ALWAYS implement actions through an ActionGateway allowlist.
- ALWAYS write changes as SMALL PATCHES/DIFFS, not full file
rewrites.

TECH CHOICE (pick one and commit):
Option A: Tauri + TypeScript + SQLite + Node/Express backend
Option B: Electron + TypeScript + SQLite + Node/Express backend
Option C: SwiftUI macOS app + embedded local server + SQLite
Choose the simplest for fastest implementation.

DATA MODEL:
Implement the SQLite schema exactly as specified: locations,
staff, lead, conversation, message, appointment, event_queue,
audit_log, settings.

EVENTING:
Implement an event_queue worker that processes:
- lead_created
- inbound_message
- scheduled_tick
- appointment_no_show
```

```
- daily_summary
with the state machine described.

INTEGRATIONS:
- Twilio SMS: inbound webhook endpoint /webhook/sms and outbound
send_sms via ActionGateway.
- Calendar: If Google OAuth configured, create/update calendar
events; else use internal calendar table logic. (If calendar
integration is too heavy for MVP, ship internal calendar only but
keep adapter interface.)

SCRIPTS:
Ship Script A/B/C defaults (editable in Settings). Enforce DNC
opt-out words.

OUTPUT FORMAT:
1) Brief file tree
2) Patch/diff blocks for each file you create/modify
3) Minimal instructions to run locally
4) A short “Test checklist” with 8–12 manual tests (lead create,
inbound reply, booking, reminders, DNC, kill switch, audit log)

START NOW.
```