Treasure Coast AI

Platform Goals, Guardrails, Wants & Don’ts

A single, saveable reference for what the platform is, how we want it to behave, and what we must avoid.

What this is built from

This binder synthesizes the project’s system prompt/brand kit plus the full QA and build-history summaries available inside this project. If you later export and upload full chat transcripts, this can be expanded into a true verbatim archive.

At-a-glance snapshot

Product type	Agency-first, multi-tenant SaaS for AI assistants + website widget
Primary operator	Treasure Coast AI (agency / super-admin) controls assistants and configuration
Client role	Business owner views dashboards (leads, conversations, bookings, analytics) - not a bot builder
End user	Website visitor chats with the assistant, leaves contact info, requests bookings
North-star result	Demo-ready, trustworthy platform that closes recurring clients and stays simple to run

Contents

1. Platform Purpose & North Star

Treasure Coast AI is built to be **sellable** and **demoable**: an agency-first platform that lets the agency rapidly deploy AI assistants for clients, embed them on websites, and show measurable outcomes (leads, conversations, bookings).

North-star outcomes (what “success” looks like):

- A non-technical business owner can see value in under 60 seconds (landing page + demo assistant).
- A visitor chat can reliably: answer FAQs, capture contact info, and create a booking when requested.
- The client dashboard shows real activity: leads, bookings, conversations, and simple analytics.
- The agency can set up a new workspace + assistant quickly (wizard works on laptops and mobile).
- The platform feels trustworthy: no data leaks, no broken states, no scary technical errors.

This platform exists to power the broader Treasure Coast AI business model: start with simple bot + automation services, close recurring clients, then scale into deeper integrations as capacity grows.

2. Audience, Roles, and Boundaries

Primary audiences

- **Agency / Super Admin (Tyler / internal team)**: builds and configures assistants, manages workspaces, manages client logins, and monitors results.
- **Client (Business owner / staff)**: views dashboards and outcomes; can manage basic business info if enabled, but should not be forced to “build AI.”
- **End user (Website visitor)**: wants quick answers, frictionless contact capture, and an easy way to book/call.

Role boundaries (important)

- Agency controls assistants and configurations.
- Clients get dashboards (leads, bookings, conversations, analytics) - **not** a bot builder.
- Every client lives in an isolated workspace; no cross-tenant visibility.

If a feature weakens these boundaries (e.g., giving clients full bot-building power), it should be treated as a deliberate product decision - not an accident.

3. Non-Negotiable Product Principles

Principle	What it means in practice
Agency-first control	The platform is optimized for an agency delivering outcomes, not for DIY end-clients building bots.
Multi-tenant by default	Workspaces enforce strict data isolation; admin can manage across tenants; clients can’t.
Demo-first UX	Everything should look smooth in a live demo: no overflowing modals, no dead controls, no technical errors.
Simple, practical, local-friendly voice	Copy and assistant tone stay human, helpful, and non-corporate.
Momentum over complexity	Ship a smaller, reliable core before adding advanced features like billing, marketplaces, or deep customization.

4. Core Capabilities (What We Want)

MVP capabilities (must work reliably)

- **Landing page** that explains the value fast, with clear CTAs and clean layout on desktop + mobile.
- **Workspace creation** (New Client wizard) that works on laptop screens and scrolls inside the modal, not the page.
- **Assistant configuration** controlled by the agency (prompts, business data, widget settings).
- **Embeddable chat widget** tied to the correct workspace/assistant.
- **Lead capture**: chats can create lead records (name/email/phone) that show up in dashboards.
- **Conversation logging**: full transcripts visible, searchable/reviewable.
- **Bookings**: chat can create bookings (tour/call/etc.) and dashboards reflect them.
- **Basic analytics** per workspace: leads, bookings, conversations/messages (with parity between admin and client views).
- **User management**: create client logins; password reset; “View as Client” impersonation for debugging and demos.

Operational capabilities (must feel polished)

- Responsive tables and dashboards (no overflow, readable columns, stable pagination).
- Clear status/tagging for leads and bookings; obvious timestamps.
- Visible confirmation after key actions (created workspace, created user, booking logged, etc.).
- Demo workspaces that look alive with realistic data and metrics.

5. Experience Standards (Tone, UX, Demo Readiness)

Brand voice and copy

- Approachable, modern, clean, trustworthy, local-friendly.
- Straightforward language; avoid corporate jargon and “AI hype.”
- Focus on outcomes: saved time, faster replies, more leads, easier bookings.

Demo readiness rules

- No hostile or technical error surfaces during a demo (errors should appear as friendly in-chat messages when possible).
- Key demo scenarios must succeed end-to-end: FAQ -> contact capture -> booking -> dashboards update.
- Admin should be able to jump into a workspace fast and use “View as Client” without broken states.

UX quality bar

- No dead controls (buttons/links that don’t persist changes).
- No modal traps: all multi-step flows must be usable on laptop screens without zooming.
- Mobile experience should be functional, not “desktop squished.”
- Consistent validation: field-specific errors, clear next actions, no vague “something went wrong.”

6. Reliability & Error Handling Standards

The platform is allowed to hit external limits (like OpenAI rate limits). What's **not** allowed is making the app look broken.

Required behaviors

- Convert assistant/API failures into **in-character** chat bubbles (not scary red toasts).
- Fail gracefully: keep the conversation stream intact and encourage a fallback action (call/text/email).
- Avoid inconsistent data states: leads and bookings created via chat must appear in dashboards quickly and reliably.
- Admin and client dashboards must report the same underlying metrics for the same workspace.

Observability (without leaking sensitive info)

- Log enough to debug in development.
- In production: do not log password reset tokens or full reset URLs; avoid logging raw chat content if it risks privacy.
- Error messages shown to users should be friendly and non-technical.

7. Security, Privacy & Multi-Tenancy Requirements

Multi-tenant rules

- Every workspace is isolated. No cross-tenant reads, even accidentally.
- Super admin can manage all workspaces; client users are strictly scoped to their workspace.
- “View as Client” must not become a way to jump into the wrong tenant or leak data.

Password reset and auth standards

- Secure reset tokens: long random tokens, stored hashed, expiring within a short window (e.g., 60 minutes).
- Reset endpoints validate token before showing the reset screen and before accepting the new password.
- Production safety: don’t expose reset URLs/tokens in logs; don’t leak internal stack traces.

Data handling standards

- No mixing demo/QA data with real client workspaces.
- Sensitive fields (emails, phone numbers, transcripts) should be treated as private.
- Default posture: least privilege and minimal exposure.

8. Roadmap Priorities (What to build next)

If we stay disciplined, this roadmap keeps the product sellable and stable.

Priority 1: Demo to first paying clients

- Lock down the end-to-end demo flows across 2-3 industries (Faith House + 2 templates).
- Make setup ridiculously fast: new workspace -> assistant -> widget -> first lead.
- Add minimal sales enablement: shareable demo links, simple ROI copy, downloadable one-pagers.

Priority 2: Operational scale (agency efficiency)

- Templates: assistants, FAQ packs, booking types, lead tags/statuses.
- SOP-driven onboarding inside the platform.
- Analytics that matter (not vanity): lead-to-booking, response coverage, missed-intent reporting.

Priority 3: Advanced features (only after core is rock-solid)

- CRM integrations, email/SMS automations, pipelines, AI voice.
- Billing/Stripe if you decide to productize self-serve accounts.

9. Stuff We Don’t Want (Anti-Patterns / Out-of-Scope)

These are the traps that make the platform harder to sell, harder to demo, or harder to maintain.

Hard “don’ts” (avoid completely)

- Giving clients a full bot builder by default (breaks the agency-first model and adds support burden).
- Technical error toasts or stack traces in front of prospects.
- Data leakage across workspaces - even once.
- Dead UI controls (looks like a scam product in demos).
- Layouts that overflow, clip, or hide primary actions (Next/Create buttons off-screen).

Soft “don’ts” (only do if there’s a strong reason)

- Complex billing and payments as a prerequisite to selling (can be optional early).
- Over-engineering analytics before core capture + booking is flawless.
- Overly long AI explanations; customers want results, not theory.
- Generic corporate copy that sounds like every other SaaS.

Product smell tests

- If a feature makes the demo longer or more confusing, it’s probably wrong for now.
- If a feature increases support load without increasing close rate, delay it.
- If a feature risks tenant isolation, it’s not worth it.

10. Acceptance Checklist (Demo + Launch)

Use this as the go/no-go checklist before showing the platform to a prospect.

Demo readiness

- Landing page loads fast and reads clean on mobile and laptop.
- Faith House demo page works: widget loads, answers FAQs, captures contact info, creates booking.
- Lead shows up in both admin and client dashboards with correct fields and timestamps.
- Booking shows up with type/time and is visible in dashboards.
- Conversations list shows transcript; no missing messages.
- Admin analytics match client analytics for the same workspace.
- New Client wizard works on laptop screens; buttons never off-screen; validation is clear.
- Create login + password reset works; production does not leak tokens/URLs in logs.
- Rate-limit/error scenario produces a friendly in-chat message (not a red toast).
- No cross-tenant data leaks (spot-check two workspaces).

If any single item fails, fix it before demoing. The platform wins by looking stable and trustworthy.

Sources used inside this project

This binder was synthesized from these project artifacts:

- MASTER SYSTEM + PROJECT PROMPT — Treasure Coast AI (PDF)
- TREASURE COAST AI — Complete Success Starter Pack (PDF)
- Treasure Coast AI — Full Assets Pack (PDF)
- Treasure Coast AI — Full Conversation, QA, and Build History (Condensed) (PDF)
- Treasure Coast AI — Full Conversation & Work History Summary (Detailed) (PDF)

Note: this is a structured synthesis, not a verbatim transcript of every chat message. If you upload/export full transcripts, we can append them as an indexed appendix.

Treasure Coast AI Platform

Addendum - No Payment Processing (Product Guardrail)

Date: December 12, 2025

Policy statement

The platform must not collect, store, transmit, or process payments on behalf of clients. No card data, bank data, digital wallet credentials, payment tokens, or checkout forms are hosted or handled by us. Payments are always completed inside the client's existing payment or booking provider (or their own website).

What Book Now does

- Shows availability and captures pre-booking details as needed for tracking (e.g., selected service, date/time, location, staff, and basic lead info).
- Creates/updates an internal booking intent record so we can attribute conversions and report performance.
- Redirects the user to the client's external booking/checkout URL to finalize the appointment and payment (if required).
- Does not embed payment fields, run card authorization, or run refunds/chargebacks.

Tracking and attribution requirements

- Every redirect must include UTM parameters or an equivalent attribution payload.
- Log a "booking_redirect" event with timestamp, client, service, and the destination provider.
- If the provider supports it, consume a webhook or confirmation URL to log "booking_confirmed" and reconcile outcomes.
- If confirmation is not available, report redirects as "booking intents" and clearly label them (no inflated claims).

Industries and edge cases

Some industries require payment at the time of booking (e.g., medical aesthetics, fitness studios, rentals, home services with deposits, ticketing, and many professional services). These industries are still eligible as long as the payment step is handled entirely by the client's provider. If a client demands in-platform checkout, that requirement is out of scope.

Allowed integrations (examples)

Redirect and confirmation patterns work with common schedulers and CRMs, including provider-hosted checkout flows. We can integrate via links, calendars, APIs, and webhooks; we do not integrate as a payment processor.

Hard "no" list (non-negotiable)

- No Stripe/PayPal/Square/Braintree checkout hosted by us (even if framed as "simple").
- No storage of cardholder data, payment tokens, or bank account credentials.
- No PCI scope expansion (no SAQ-D type responsibilities).
- No disputes workflow, refunds, chargebacks, or payment reconciliation inside our platform.
- No subscription billing managed by the platform for clients.

Client responsibilities

Clients own their payment terms, taxes, refunds, chargebacks, and compliance requirements. They must provide the booking/checkout destination URL(s) and, where possible, a confirmation mechanism (webhook/receipt page) so reporting can be accurate.

Acceptance checklist

Requirement	Pass condition
No payment UI	Platform UI never collects payment credentials.
Redirect-only checkout	Book Now always sends users to client-hosted booking/checkout.
Attribution	Redirects include UTM (or equivalent) and are logged.
Reporting honesty	Confirmed bookings separated from intents when confirmation is unavailable.
Security scope	No PCI/processor credentials stored or used by platform.