HopeLine / Treasure Coast AI – Super Admin Phase 1 Spec

Per-Client Control Panel (Safe, Incremental Upgrade)

READ THIS FIRST
This document defines PHASE 1 of the Super Admin system.

GOAL
Give the super admin (platform owner) full control over each client's chatbot configuration from /super-admin, WITHOUT changing core behavior or breaking the existing /admin experience.

This is a SAFE, incremental update. It must not affect:
- Chatbot conversation core logic
- Appointment booking flow
- Crisis handling
- Analytics calculations
- Email notification logic
- Client-facing /admin routes

Only add/extend configuration and super-admin UI.

SECTION 1 – ARCHITECTURE & PRINCIPLES

1.1 Separation of roles
There are two distinct admin surfaces:

1) Client Admin (/admin/...)
  - Used by the business (e.g., Faith House)
  - Limited controls: view leads, manage appointments, see analytics
  - Possibly edit simple settings like business contact email

2) Super Admin (/super-admin/...)
  - Used ONLY by the platform owner (Treasure Coast AI)
  - Full control over each client's chatbot configuration
  - Controls tone, knowledge, flows, notifications, etc.

Phase 1 focuses ONLY on /super-admin.

1.2 Data model pattern
Introduce or extend a per-client configuration object, e.g.:

ClientConfig:
- id
- clientId (string)
- businessName (string)
- tagline (string)
- businessType (string)
- primaryEmail (string)
- primaryPhone (string)
- websiteUrl (string)
- city (string)
- state (string)
- timezone (string)
- defaultContactMethod (string)
- internalNotes (text)
- appointmentTypes (JSON)
- preIntakeConfig (JSON)
- notificationSettings (JSON)
- knowledgeBase (JSON or related table)

For Phase 1, you may store complex settings in JSON fields for simplicity, as long as the shape is consistent between backend and frontend.

1.3 Safety rules
- Do NOT remove or break existing tables/models.
- Do NOT alter appointment creation, crisis handling, or analytics logic.
- Only extend configuration and super-admin UI so that existing behavior can read from new config where appropriate.

SECTION 2 – SUPER ADMIN ENTRY & CLIENT SELECTION

2.1 Route
- Super admin main entry: /super-admin
- This should show a list of clients OR a client selector.

2.2 Client list (Phase 1 minimal)
Implement a simple client picker at the top of the /super-admin interface:

- Dropdown or sidebar list of clients:
  - Client name
  - Status (Active / Paused)
- When a client is selected, all tabs (General, Knowledge, Hours, Notifications, Branding, Privacy) refer to that client's configuration.

For now, Faith House can be the single client record, but the UI should be ready for more clients later.

2.3 Active client context
- Store the active clientId in super-admin UI state (e.g., React context).
- All API calls from super-admin must include or infer clientId so configuration is scoped per client.

SECTION 3 – GENERAL TAB (SUPER ADMIN)

The General tab should fully configure the high-level business profile for the selected client.

3.1 Fields
Add the following fields in the General section for the active client:

- Business Name (text)
- Tagline (text)
- Business Type (dropdown or text; e.g.: "Sober Living", "Barbershop", "Gym")
- Primary Phone (text)
- Primary Email (text)
- Website URL (text)
- City (text)
- State/Region (text)
- Timezone (dropdown of IANA timezones)
- Default Contact Method (dropdown: "phone", "text", "email")
- Internal Notes (long textarea, visible only to super-admin)

3.2 Behavior
- Fields load from ClientConfig for that client.

- “Save All Changes” persists to backend via a dedicated endpoint, e.g.:
  - PUT /api/super-admin/clients/:clientId/general

3.3 Usage by other parts of the system
- Existing /admin views and notifications may read these values (e.g., businessName, primaryEmail, timezone), but existing behavior must continue to work even if some fields are missing.
- Any mapping from old settings to new ClientConfig should be handled safely in the backend.

SECTION 4 – KNOWLEDGE TAB

The Knowledge tab allows super-admin to control what the bot knows and how it answers client-specific questions, without touching core system prompts.

4.1 Concepts
Two main concepts:

1) FAQ Entries (Q&A pairs)
2) Long-form Knowledge Sections

4.2 FAQ Entries
Provide a table/list of FAQ items for that client:

For each FAQ:
- id
- category (string; e.g.: “Pricing”, “Requirements”, “Program”, “Contact Info”)
- question (string)
- answer (text)
- active (boolean)

UI actions:
- Add FAQ
- Edit FAQ
- Delete FAQ
- Toggle active/inactive

Backend:
- Store in a knowledgeBase collection keyed by clientId, or inside a JSON knowledgeBase field containing an array of FAQ entries.
- API endpoints:
  - GET /api/super-admin/clients/:clientId/knowledge
  - POST /api/super-admin/clients/:clientId/knowledge
  - PUT /api/super-admin/clients/:clientId/knowledge/:faqId
  - DELETE /api/super-admin/clients/:clientId/knowledge/:faqId

4.3 Long-form Sections
Allow super-admin to define a few labeled sections:

Examples:
- “About the Program” (text)
- “House Rules & Expectations” (text)
- “Who This Is For / Not For” (text)
- “Payment & Insurance Info” (text)

UI:
- Show a list of section titles.
- Each section opens a rich textarea for editing content.

Backend:

- Store as part of knowledgeBase JSON, e.g.:
  - longForm: { aboutProgram, houseRules, whoItsFor, paymentInfo }

4.4 Chatbot integration (Phase 1)
- Do NOT change core system prompt behavior right now.
- As a first step, ensure the knowledgeBase is available in the backend where the chat completion is built so that future phases can include retrieval or prompt injection.
- Optionally, if a simple retrieval layer already exists, plug this structured knowledge into that logic.

SECTION 5 – APPOINTMENT TYPES & PRE-INTAKE CONFIG

These may live in a dedicated tab (e.g., “Flows”) or in General for Phase 1. If easier, reuse an existing tab or add a sub-section.

5.1 Appointment Types
For each client, define the appointment types the bot can offer.

Fields per type:
- id
- label (e.g., “Phone Call”, “Tour”, “Family Call”)
- description (optional)
- durationMinutes (number)
- category (e.g., “lead”, “existing_client”)
- active (boolean)

UI:
- Table/list of appointment types.
- Buttons: Add, Edit, Delete, Toggle Active.

Backend:
- Store in ClientConfig.appointmentTypes (JSON array) or a related table keyed by clientId.
- API endpoints:
  - GET /api/super-admin/clients/:clientId/appointment-types
  - PUT /api/super-admin/clients/:clientId/appointment-types (for batch updates), or CRUD endpoints per type.

Usage:
- The chatbot’s booking flow should read from these types when showing options to users.
- For Phase 1, preserve existing defaults and simply allow super-admin to override them.

5.2 Pre-Intake Configuration
Allow super-admin to define the sequence of questions asked in the pre-intake flow.

For each question:
- id
- label (e.g., “Who is this for?”)
- internalKey (e.g., “forWho”, “sobrietyStatus”)
- type (e.g., “single_choice”, “multi_choice”, “text”)
- options (array of { value, label } for choice questions)
- required (boolean)
- order (number)
- active (boolean)

UI:
- List of questions with drag-and-drop reordering preferred (or up/down buttons).

- Editor for each question (type, label, options, required).
- Toggle to enable/disable a question.

Backend:
- Store in ClientConfig.preIntakeConfig (JSON).

Usage:
- The frontend PreIntakeFlow should render dynamically from this config (Phase 1 can keep structure simple and just adjust defaults).

SECTION 6 – NOTIFICATIONS TAB (PHASE 1)

Super admin should control, per client, where and how notifications are sent, without changing the core sending logic.

6.1 Staff endpoints
Fields:
- Staff Emails (multi-value list)
- Staff SMS Numbers (multi-value list, for future SMS)
- Preferred Staff Channel:
  - “email_only”
  - “sms_only”
  - “email_and_sms”

6.2 Notification toggles
Toggles for each event type:
- On new appointment: Email to staff (boolean)
- On new pre-intake with no appointment (future toggle; can be added as placeholder)
- On same-day appointment (future; placeholder)

6.3 Templates (simple Phase 1)
For now, keep templates minimal, but prepare structure:

- Staff email subject template
- Staff email body template

Use tokens like:
- {{clientName}}
- {{leadName}}
- {{appointmentType}}
- {{preferredTime}}

Backing data:
- Store in ClientConfig.notificationSettings JSON with keys:
  - staffEmails: string[]
  - staffPhones: string[]
  - staffChannelPreference: string
  - eventToggles: { newAppointmentEmail: boolean, ... }
  - templates: { staffEmailSubject: string, staffEmailBody: string }

Backend:
- Ensure that when notifications are sent, they read from notificationSettings for that client, with sensible defaults if fields are missing.

SECTION 7 – BACKEND ENDPOINTS (SUPER ADMIN API)

7.1 General pattern

All super-admin configuration endpoints should be namespaced, for example:

- GET /api/super-admin/clients
- GET /api/super-admin/clients/:clientId/general
- PUT /api/super-admin/clients/:clientId/general
- GET /api/super-admin/clients/:clientId/knowledge
- PUT/POST/DELETE /api/super-admin/clients/:clientId/knowledge/...
- GET /api/super-admin/clients/:clientId/appointment-types
- PUT /api/super-admin/clients/:clientId/appointment-types
- GET /api/super-admin/clients/:clientId/pre-intake
- PUT /api/super-admin/clients/:clientId/pre-intake
- GET /api/super-admin/clients/:clientId/notifications
- PUT /api/super-admin/clients/:clientId/notifications

7.2 Authentication
- Use whatever admin auth exists now for /super-admin.
- Ensure that only super-admin users can access these routes.

7.3 Error handling
- Respond with clear error messages on validation failures.
- Do not impact existing /admin endpoints in any way.

SECTION 8 – FRONTEND SUPER ADMIN UI

8.1 Tabs
For Phase 1, make sure these tabs exist and are wired to the active client:

- General
- Knowledge
- Hours (can remain basic for now)
- Notifications
- Branding (can remain stub or minimal)
- Privacy (can remain stub or minimal)

General, Knowledge, and Notifications must be implemented per this spec. Appointment types and pre-intake config may live within General or Knowledge as sub-sections, or as a separate tab (e.g., Flows), depending on current code organization.

8.2 Save behavior
- Each tab should have a “Save” or “Save All Changes” button.
- When clicked, call the appropriate /api/super-admin/* endpoint.
- Show success/error toasts.

8.3 Do not affect client admin
- Ensure that nothing in /super-admin directly exposes sensitive configuration to /admin.
- Client-facing /admin should only allow safe edits (if any), while super-admin remains the place for deep configuration.

SECTION 9 – DONE CRITERIA FOR PHASE 1

Super Admin Phase 1 is COMPLETE when:

- [ ] /super-admin supports selecting a client context (even if there is only one client today).
- [ ] The General tab allows editing full business profile fields and saves them per client.
- [ ] The Knowledge tab allows managing FAQ entries and long-form

sections per client.
- [ ] Appointment types and pre-intake configuration are editable per client and are stored in client configuration (and used by the frontend flows without breaking existing behavior).
- [ ] The Notifications tab allows configuring staff emails, staff SMS numbers (for future use), basic toggles, and simple templates.
- [ ] All configuration is scoped per clientId.
- [ ] /admin behavior for the client still works exactly as before (appointments, analytics, summaries, crisis behavior, email notifications) while now reading from updated config where applicable.
- [ ] No breaking changes or regressions are introduced in the client-facing parts of the app.

Replit should implement this spec step-by-step and verify all existing flows after each major change.