

HopeLine Assistant – Final Implementation Instructions for Replit

FINAL IMPLEMENTATION INSTRUCTIONS FOR REPLIT HopeLine Assistant – Last Mile Upgrades

Use this PDF TO FINISH the HopeLineAssistant project.

Assume the project already roughly follows:

- FaithHouse_HopeLine_TopTier_Spec.pdf
- Replit_Master_Implementation_Prompt.pdf

Your job now is to close ALL remaining gaps and polish it so it is production-ready.

SECTION 1 – READ THIS FIRST

1. DO NOT rebuild the app from scratch.
2. KEEP all working features and structure.
3. Only add/upgrade what is missing or incomplete:
 - Real notifications (email + SMS)
 - AI conversation summaries
 - Pre-intake <-> appointment linking
 - Admin panel polish
 - Analytics + categories
 - Crisis handling verification

At the end, the app must:

- Build and run without errors.
- Provide real value to The Faith House.
- Be ready to reuse as a template for other clients.

SECTION 2 – REAL EMAIL + SMS NOTIFICATIONS

Goal:

Staff must be automatically notified on NEW appointments, and clients should receive confirmation messages.

2.1 ENVIRONMENT VARIABLES

Add and use these (server-side only):

- EMAIL_NOTIFICATIONS_ENABLED (boolean string: "true"/"false")
- SMS_NOTIFICATIONS_ENABLED (boolean string)
- NOTIFICATION_FROM_EMAIL
- NOTIFICATION_STAFF_EMAIL
- SMTP_HOST
- SMTP_PORT
- SMTP_USER
- SMTP_PASS
- TWILIO_ACCOUNT_SID
- TWILIO_AUTH_TOKEN
- TWILIO_FROM_NUMBER

2.2 STAFF EMAIL NOTIFICATION

In the appointment creation API (e.g. POST /api/appointment):

- After saving an appointment:
 - If EMAIL_NOTIFICATIONS_ENABLED and NOTIFICATION_STAFF_EMAIL are set:
 - Send an email that includes:
 - Name
 - Phone
 - Email

- Appointment type
- Preferred date/time
- Contact preference
- Notes
- Pre-intake answers (if available)
- AI conversation summary (see Section 3)

Use Nodemailer or any mail library that works in this environment.

2.3 STAFF SMS NOTIFICATION

After appointment creation:

- If SMS_NOTIFICATIONS_ENABLED and a staff phone number exists in settings:

- Use Twilio (or equivalent) to send an SMS like:

"New Faith House inquiry from {name}. Type: {appointmentType}. Preferred: {preferredTime}. Check your admin panel for details."

2.4 CLIENT SMS CONFIRMATION

If the client provided a valid phone number and prefers text or has agreed to SMS:

- Send an SMS like:

"Hi {name}, this is The Faith House. We received your request for a {appointmentType} at {preferredTime}. If anything changes, please contact us at {businessPhone}."

Later, reminders can be added, but not required in this step.

SECTION 3 – AI-GENERATED CONVERSATION SUMMARY

Goal:

Staff should see a short, AI-generated summary for each appointment, so they don't have to read full chat logs.

3.1 DATA MODEL

- Add a new field to the appointments table/model:
 - summary (TEXT / string)

3.2 SUMMARY GENERATION

In the appointment creation handler, AFTER the appointment is saved:

1. Load the last 20–40 messages for this sessionId (user + assistant).
2. Call OpenAI with a prompt like:

"Summarize this chat for a staff member at a sober living home.

Include:

- Who is reaching out (self or loved one)
 - Basic situation
 - How urgent it seems
 - What they are hoping for
 - Any key context the staff should know before the first call.
- Make it 3–6 sentences, clear and neutral."

3. Store the resulting summary text in appointment.summary.

4. Include this summary in:

- Staff email notification
- Admin appointment detail view

If the summary generation fails, log the error but DO NOT block appointment creation.

SECTION 4 – PRE-INTAKE <-> APPOINTMENT LINKING

Goal:

All pre-intake answers must be visible in the admin panel and tied to the appointment.

4.1 DATA MODEL

- Create a preIntake table or add fields to appointment such as:
 - forWho
 - sobrietyStatus
 - financialSupport
 - timeline
 - preIntakeNotes

OR

- Store pre-intake as a JSON field on appointments (e.g. preIntakeData JSONB/text with serialized JSON).

4.2 FRONTEND BEHAVIOR

- When the pre-intake flow is submitted:
 - Save data for the current session (e.g. via API /api/pre-intake or stored in session).
- When an appointment is created from the same session:
 - Attach this pre-intake data to the appointment (fields or JSON).

4.3 ADMIN VIEW

In the admin appointments list/detail:

- Show pre-intake answers in a readable block, for example:

Pre-Intake:

- For who: Myself
- Sobriety: Currently sober
- Support for fees: Yes
- Timeline: As soon as possible
- Notes: [text]

SECTION 5 – ADMIN PANEL POLISH

Goal:

Make the business admin page feel like a real app the staff will want to use daily.

5.1 APPOINTMENT LIST

Ensure each row shows at least:

- Name
- Appointment type
- Preferred time
- Status
- Notes (or indicator)
- Created date
- Summary preview (e.g. first 80 chars)
- Clickable "View details" or expandable row

5.2 FILTERS & SEARCH

Add:

- Filter by status (New, Contacted, Scheduled, Completed,

Cancelled)

- Search by name, email, or phone

5.3 STATUS & NOTES

- Status should be editable via dropdown or button group.
- Notes should be editable (inline or in a modal).
- All changes must persist via API.

5.4 CONTACT LINKS

- Phone numbers: use "tel:" links.
- Emails: use "mailto:" links.

5.5 EXPORT

- Add a simple "Export CSV" button that exports current filtered appointments to a CSV file (in browser).

SECTION 6 – ANALYTICS & MESSAGE CATEGORIES

Goal:

Provide basic analytics that show how well the assistant is working.

6.1 MESSAGE LOGGING

Ensure every message is logged with:

- sessionId
- role ("user" or "assistant")
- content (trimmed if very long)
- timestamp
- category (string)

6.2 CATEGORIES

Implement a simple categorization (rule-based or model-assisted) for messages:

Categories:

- faq_general
- pricing
- availability
- requirements
- application_process
- pre_intake
- crisis_redirect
- contact_info
- other

You can categorize based on keywords or use a small helper function.

6.3 ANALYTICS SCREEN

Create an Analytics section in the admin (or expand existing one) that shows for a selected timeframe:

- Total sessions
- Total appointments
- Conversion rate = appointments / unique sessions
- A count or percentage breakdown of categories
- Simple peak-hours info (e.g. messages by hour of day)

Charts can be very simple (text or minimal visuals). Functionality is more important than fancy UI.

SECTION 7 – CRISIS HANDLING VERIFICATION

Goal:

Ensure the bot NEVER tries to handle emergencies and ALWAYS redirects to real crisis services.

7.1 PROMPT CHECK

- Confirm the system prompts for EN and ES contain the crisis instructions (988/911, no crisis counseling).

7.2 BEHAVIOR TESTING

- Add or update a simple unit/integration test (if test framework exists) or a manual checklist that verifies:

- If user mentions "suicide", "kill myself", "hurt myself", "want to die", etc.:

- The response always:
 - Acknowledges distress
 - States it cannot handle emergencies
 - Directs to 988 or 911
 - Does NOT continue casual conversation about it

If no tests exist, add a TODO comment and ensure this behavior is strongly enforced in the prompt.

SECTION 8 – MULTI-TENANT PREP LIGHT PASS

If not already done, prepare the schema for future multiple clients (light level):

- Create a clients table with id and name (and slug/domain if needed).
- Add clientId to settings, appointments, and admin users.
- For now, it is acceptable if only one client (Faith House) exists and is assigned clientId=1.

Do NOT overcomplicate this step—just lay the groundwork.

SECTION 9 – SECURITY & CLEANUP

- Require authentication for all admin/analytics pages.
- Ensure no secrets or API keys are exposed to frontend.
- Remove any leftover console.log debugging of sensitive data.
- Confirm widget clearly states: "Not an emergency service. For crisis, call or text 988, or call 911."

SECTION 10 – FINAL PASS & REPORT

Before finishing, do a final pass against:

- FaithHouse_HopeLine_TopTier_Spec.pdf
- Replit_Master_Implementation_Prompt.pdf
- This Final Implementation Instructions PDF

Make sure:

- No "..." placeholder text remains.
- All core flows (chat, pre-intake, appointment, notifications, admin, analytics) work without crashing.
- The code is reasonably organized and commented where helpful.

At the end, output a short report summarizing:

1. Files changed.
2. New environment variables needed.
3. How to run, test, and use the system end-to-end.