RSV POC Website + Dual-Mode Chatbot
Product Requirements Document (PRD)

Version 0.1 | 16 Dec 2025 | Internal POC

1. Executive summary

Build a simple 7-page placeholder microsite that demonstrates navigation and a single chatbot widget with a mode switch. The chatbot supports (A) Guided-only Q&A (no free text) and (B) Free-text Q&A using LLM-based intent classification, but responses must always come from an approved response bank. The key proof point is deep-linking from chatbot answers to relevant internal POC pages and safe/compliant fallback behaviour for unsupported queries.

2. Goals and non-goals

2.1 Goals

- Demonstrate a multi-page website experience (7 pages) with working navigation.
- Provide a single chatbot widget with a clear mode switch: Guided vs Free text.
- Guided mode: users can only click pre-defined questions; bot returns approved answers and suggests next steps.
- Free-text mode: user text is classified by an LLM into an approved intent; the bot returns the response-bank answer for that intent (no improvisation).
- Deep-linking: bot answers can include 1–2 ‘Read more’ buttons that navigate to specific internal POC pages.
- Compliant fallbacks: for out-of-scope queries, redirect the user to supported topics and/or to appropriate support pathways as defined in the response bank.
- No analytics/logging requirement (beyond basic console logs for debugging).

2.2 Non-goals

- Pixel-perfect styling or visual cloning of the reference public website.
- A full production-grade content-management system.
- User authentication, role-based access, or hosting hardening beyond basic API key handling.
- Generating new medical advice or free-form medical content beyond the approved response bank.

3. Target users and stakeholders

Primary users: internal stakeholders reviewing a proof-of-concept (product, medical, compliance, engineering).

Secondary users: demo audiences who will click through pages and test chatbot flows.

4. Site map (7 pages)

Pages are placeholders with minimal content (title, short description, and a few internal links).

Page	Route	Purpose (POC)
Home	/	Landing page; links to all pages; short 'About this POC' blurb.
RSV basics	/rsv-basics	What RSV is, who it can affect; placeholder sections.
Symptoms in adults	/symptoms-adults	Common symptoms; prompts to speak to healthcare professional.
Spread and prevention	/spread-prevention	How RSV spreads; general prevention tips (placeholder).
How long RSV lasts	/duration	Duration/contagiousness placeholders; links to symptoms and what-to-do.
What to do next	/what-to-do	General 'seek advice' guidance; emergency/support pathways (placeholder).
About the campaign / support	/about-support	Who is behind campaign (placeholder); contact/support links.

5. Navigation and layout

Layout should remain intentionally simple and consistent across pages:

- Top navigation bar with links to all 7 pages.

- Main content column + a persistent chatbot area (recommended: right sidebar or Streamlit sidebar).
- All navigation is internal within the POC (no external-link modal).
- Optional: breadcrumb or ‘Back to Home’ link on each page.

6. Chatbot requirements

6.1 Widget structure

A single chatbot widget with a mode switch (tabs or segmented control):

- Mode A: Guided (no free text).
- Mode B: Free text (LLM intent classification).

Both modes share:

- Common response rendering (answer + optional redirects + optional deep links).
- Conversation history per mode (kept in session state).
- Clear ‘Start over / Clear chat’ control.

6.2 Guided mode (no free text)

- Show user-friendly categories (derived from response bank metadata).
- Within each category, show a list of pre-defined question buttons.
- On click, append user question + bot answer to the chat history.
- After answering, show 2–3 ‘Next best questions’ buttons (if available in the bank).
- After answering, show 1–2 deep-link buttons to relevant internal pages (based on intent-to-page mapping).
- No text input should be available in this mode (only buttons).

6.3 Free-text mode (LLM classified)

- Provide a text input box + Send button.
- Before calling the LLM, run simple safety routing rules for obvious emergency or adverse-event phrases (route directly to the corresponding response-bank intent).
- Otherwise call OpenAI Responses API with Structured Outputs to classify the user message into one approved intent (and optional slots).
- Render only the approved response-bank answer for the chosen intent; do not generate novel medical advice.
- If classification confidence is below threshold or no intent fits: show a response-bank-aligned fallback (e.g., ‘I can’t help with that here’) and present the supported topic buttons.

6.4 Deep-linking behaviour (core proof point)

Each response-bank intent can optionally map to 0–2 internal pages.

When present, the bot shows buttons like:

- Read more: RSV basics
- What to do next

Clicking should navigate within the Streamlit multi-page app (no full reload, no external modal).

7. Response bank and data model

All user-facing answers must come from the approved response bank provided by the project owner. The application should load this bank from a JSON file committed in the repository.

7.1 Response bank JSON schema (repo file: data/response_bank.json)

```
{
  "intents": [
    {
      "intent_id": "string_unique_id",
      "category": "string",
      "display_question": "string (for guided button)",
      "example_questions": ["string", "..."],
      "answer_markdown": "string (rendered to user)",
      "redirects": ["string", "..."],
      "guardrails": ["string", "..."],
      "next_best_intent_ids": ["string", "..."],
      "page_links": [
        { "label": "string", "route": "/path" }
      ]
    }
  ]
}
```

Notes:
- answer_markdown may include phone numbers / URLs as plain text per the bank.
- ‘monitoring team’ notes must not be shown to users; store them separately if needed.
- page_links are the internal deep links shown after the answer.

8. LLM intent classification spec

The LLM is used only for classification and slot extraction, not for generating user-facing medical content.

8.1 API choice

Use OpenAI Responses API and Structured Outputs (JSON schema) to force machine-readable classification results. Prefer SDK helpers (Pydantic) to avoid schema drift. References: OpenAI Responses API and Structured Outputs docs.

8.2 Classifier output schema (Pydantic)

```
class ClassifiedIntent(BaseModel):
  intent_id: str
  confidence: float  # 0.0 to 1.0
  slots: dict[str, str] = {}  # optional placeholders like symptom='cough'
  rationale: str  # short, non-user-facing reason for the choice
```

8.3 Classification rules

- The classifier MUST choose intent_id from the loaded response_bank intents, or return intent_id='__NO_MATCH__'.
- If the user asks for personal medical advice, diagnosis, or treatment recommendations, the classifier should prefer a bank intent that provides compliant redirection; otherwise return __NO_MATCH__.
- If user text suggests an adverse event report or emergency, route to the relevant adverse-event / emergency intent (hard-rule override).
- The application should implement a confidence threshold (default 0.70). Below threshold: treat as __NO_MATCH__.

9. Functional requirements

1. FR1: Multi-page navigation across 7 pages via top nav.
2. FR2: Chatbot widget present on every page; maintains history per mode.
3. FR3: Guided mode question buttons are generated from response_bank.json (no hard-coded questions in UI).
4. FR4: Free-text mode uses OpenAI API to classify into response-bank intents; answer returned from bank.
5. FR5: Deep links in responses navigate internally to relevant POC pages.
6. FR6: Clear fallback path for unknown queries showing supported topics and compliant guidance from bank.
7. FR7: No persistent user tracking or analytics storage.

10. Non-functional requirements

- NFR1: App should run locally with a single command and minimal setup.
- NFR2: No API keys stored in code. Load from environment variable OPENAI_API_KEY.
- NFR3: LLM calls should be server-side only (Streamlit backend). Do not expose API key to the browser.

- NFR4: Keep response latency acceptable for demos (target < 2 seconds average for classification).
- NFR5: Provide a simple README with setup steps.

11. Technical approach (recommended)

Framework: Streamlit multi-page app (Python).

- App structure uses Streamlit 'pages/' directory for routing.
- Chat widget implemented in sidebar or fixed right column; mode switch via tabs/segmented control.
- Data files in data/: response_bank.json + optional intent_to_page_map.json.
- OpenAI SDK used for classification. Use responses.parse with a Pydantic model for structured outputs.
- Provide a script to ingest/normalize the provided response-bank document into JSON (optional but recommended).

12. Acceptance criteria (definition of done)

- User can navigate across all 7 pages without errors.
- Chat widget is visible on all pages and supports switching between Guided and Free text modes.
- Guided mode shows categories and question buttons sourced from response_bank.json.
- Selecting a guided question shows the correct answer and offers next-best questions.
- At least 10 intents demonstrate deep-link buttons that navigate to the correct internal page.
- Free-text mode successfully classifies at least 10 sample inputs into correct intents and returns bank answers.
- Out-of-scope free-text queries reliably hit fallback and show supported topics without generating new advice.
- API key is read from environment variable; application runs without key by disabling free-text mode gracefully.

13. Risks and mitigations

Risk	Mitigation
Response bank contains 'monitoring team' operational notes	Strip or hide these notes from user-facing output; store as metadata if needed.
LLM misclassification	Use structured outputs + confidence threshold + hard-rule overrides for safety-critical topics; fallback to guided topics.

Streamlit navigation limitations	Use st.page_link / st.switch_page; keep routes stable and mapped in config.

Appendix A. Suggested repository structure

```
rsv-poc/
 app.py
 pages/
  1_Home.py
  2_RSV_Basics.py
  3_Symptoms_in_Adults.py
  4_Spread_and_Prevention.py
  5_Duration.py
  6_What_to_Do_Next.py
  7_About_Support.py
 components/
  chatbot_widget.py
  intent_classifier.py
  response_bank.py
 data/
  response_bank.json
  intent_to_page_map.json
 scripts/
  ingest_response_bank_docx.py  (optional)
 README.md
 requirements.txt
```