UF Urology Hub Toolkit

Maintenance prompts and replication guide
for an AI-powered residency operations retrieval-augmented workspace

Department of Urology
University of Florida College of Medicine

Jason P. Joseph, MD
jason.joseph@urology.ufl.edu

This is the system we built to maintain our residency program's operational knowledge hub on a retrieval-augmented generation (RAG) platform. It runs on Google NotebookLM and is adaptable to any RAG platform that accepts structured source documents.

We designed it so anyone, program coordinator, PD/APD, chief resident, or any team member comfortable with a word processor and an AI chatbot, can keep the system current. No code. No API. No IT ticket.

One knowledge base instead of seven

A single, queryable knowledge base that answers operational questions for your residents in plain English, grounded entirely in your program's own documents. It bridges the five to seven scattered repositories our residents were searching: email threads, shared drives, group chats, Apple Notes from graduated residents, texts to the coordinator, and whatever the senior told them last July.

This is your program’s documented knowledge made searchable, not an independent clinical advisor. The Hub retrieves and cites what your team has written: service guides, attending preferences, institutional pathways, coverage policies, operational logistics. It does not generate clinical recommendations on its own. Residents are expected to verify source citations and defer to supervising faculty and patient-specific clinical judgment.

Raw uploads are not enough

We found that uploading raw documents to a RAG platform is not enough, particularly when multiple sources overlap on the same item. As an example, date-sensitive operational queries (coverage on a specific Friday, who is on call the third weekend of the month) produce confidently blended answers when sources disagree.

The fix we implemented was a contextualization layer applied to every source before upload: per-source metadata, anticipated user questions, an interpretation guide, and a cross-source priority hierarchy that names which source wins when two of them conflict. The five prompts below build and maintain that layer.

PART 1: Standing It Up

Create the notebook

1. Create a notebook at notebooklm.google.com. Name it something your residents will recognize (e.g., [Program] Hub).

2. Share with your residents and coordinator. View access for everyone; edit access for anyone who maintains sources.

Tier:

- **Free / personal account.** Functional. Uploaded content may be used to improve Google's models. Fine for non-sensitive operational content.
- **Google Workspace Enterprise.** Content stays in your tenant and is not used for model training. The right choice for anything tied to institutional data.

Start with five sources, not fifty

Apply the Tuesday Morning Test: what does a resident covering an unfamiliar service at 6:40 AM on a Tuesday need to know? That is your starting source list.

How to surface candidates:

- **Listen for the questions residents text each other.** The recurring ones (coverage swaps, attending preferences, code and extension lookups) tend to be the highest-yield sources.
- **Ask your coordinator and chief residents what they get asked most.** Anything they answer more than once a week is a source candidate.
- **Engage your power users early.** Residents already maintaining shared notes know which information is missing and which channels need to retire. Build with them; do not replace them.

Where we started: the weekly coverage schedule, our busiest service's cookbook, the on-call escalation policy, a contacts and codes directory, and the faculty away calendar. Five sources at launch; we grew from there as gaps surfaced.

Plan a live-sync layer for high-churn content

Static-file maintenance (download, edit, re-upload) works for most sources. But the highest-churn categories (door codes, faculty preferences, weekly schedule deltas, contact directories) generate enough small edits that a static workflow creates noticeable friction. Pair the static playbook with a live-sync editing layer (e.g., Google Docs that sync into NotebookLM) for your two or three highest-churn categories at launch. Retrofitting it later is harder than building it in.

PART 2: The Five Maintenance Prompts

These are the five prompts we use to maintain the hub. Run each in the LLM of your choice (ChatGPT, Claude, Gemini) by pasting the prompt block at the top, then your inputs below it. The output is the new or updated Markdown source file, ready to upload to NotebookLM.

The structural rules inside Prompt 1 (Header, Metadata, Anticipated User Questions, Contextual Interpretation Guide, Main Content, Source End Marker) are the contextualization layer. Every source in the hub follows that template; the prompts below build and preserve it.

These five prompts reflect our program's operational cadence: a weekly coverage schedule that arrives by email, a faculty away calendar that updates ad hoc, and a source library that grows as gaps surface. Your program's set will map to your own workflows. The pattern (reusable prompts that preserve the contextualization layer) is what transfers; the specific count and scope are yours to adapt.

Prompt 1: Creating a New Source File

Use when adding a document that does not yet exist in the hub: a new policy, a new service guide, a new faculty reference.

The Prompt:

> You are an AI data architect for a residency program's operational knowledge hub. Your task is to process a new document and transform it into a structured Markdown source file. The goal is to make the information highly discoverable for natural language queries.
>
> Your output MUST be only the raw Markdown text. Do not include any in-line citations or explanatory text.
>
> The output MUST strictly follow this format:
>
> 1. Header: Start with a title that clearly states the document's content.
>
> 2. Metadata Block:
> Document Type: [e.g., Program Policy, Faculty Preference, Clinical Protocol, Service Guide, Schedule]
> Original Reference Date: [Find the most recent date in the document. If none, state "Not specified."]
> Source Processed On: [Today's Date]
>
> 3. Anticipated User Questions: List 5-10 specific, natural language questions a resident might ask where this source would be the direct answer.
>
> 4. Contextual Interpretation Guide: Summarize the document's purpose in 2-3 sentences. Extract specific rules, steps, and key definitions. This section tells the RAG system how to weigh and interpret the data.
>
> 5. Main Content: Format the body using clear Markdown. Transcribe all relevant text. If the original contains charts or images, describe them in detail.
>
> 6. Source End Marker: Conclude with ---END OF SOURCE---

To run: Paste the full text of the new document below the prompt.

Prompt 2: Making a Specific Edit to an Existing Source

Use for small, targeted changes: updating a contact name, changing a policy detail, correcting a phone number, adding a new bullet to an existing list.

The Prompt:

> You are an AI data architect for a residency program's operational knowledge hub. Your task is to perform a surgical update to an existing source file.

> I will provide the full text of the old source and a specific instruction for the update. You must integrate the change and output the complete, rewritten source file, perfectly preserving the original Markdown structure, metadata, and formatting.
>
> Your output MUST be only the raw Markdown text. Do not include in-line citations or any text explaining what you changed.

To run:

- OLD SOURCE TEXT: Paste the entire raw Markdown text of the source you want to edit.
- UPDATE INSTRUCTION: Describe the change clearly (e.g., "Change the on-call contact from Dr. Smith to Dr. Davis").

Prompt 3: Faculty Away Calendar Maintenance

Use whenever new faculty leave dates come in. Merges new dates with the existing calendar, de-duplicates, normalizes time entries, and sorts chronologically.

The Prompt:

> You are an AI data architect for a residency program's operational knowledge hub. Your task is to update the canonical "Faculty Away Calendar" source file. You will be given the previous version of the calendar and new information about faculty away time. Intelligently merge this information, eliminate redundancies, normalize time entries, and generate a single, perfectly formatted, chronologically sorted Markdown file.
>
> Your output MUST be only the raw Markdown text.
>
> Phase 1: Parse all entries from both sources.
> Phase 2: Create a single unified list. Remove exact duplicates. Sort chronologically.

> Phase 3: Generate the New Markdown File:
> 1. Header: "Faculty Away Calendar: [Start Month Year] - [End Month Year]"
> 2. Metadata: Document Type / Original Reference Date / Source Processed On
> 3. Anticipated User Questions: Add 1-2 new questions to the existing list.
> 4. Contextual Interpretation Guide: Copy verbatim from previous version.
> 5. Main Content: Bullet per entry. Time normalization: full-day = (All Day); half-day = (Half Day - AM/PM).
> Format: "Dr. [Last Name] ([Initials]): [Reason] - [Date Range] ([Duration])"
> 6. Source End Marker: ---END OF SOURCE---

To run:

- THE PREVIOUS VERSION: Paste the full text of the current Faculty Away Calendar source.
- THE NEW ENTRIES: Paste or describe the new faculty away dates.

Prompt 4: Weekly Schedule Maintenance (Append-Only)

Use every week when the new schedule email arrives. Generates only the new week's block. Paste the output at the bottom of the master schedule file.

Customization required: The Internal Context and Glossary block must be rewritten for your program: every role abbreviation, location code, OR/clinic shorthand, and uncovered-row parsing rule. About 15 minutes once.

The Prompt:

> You are an AI data architect for a residency program's operational knowledge hub. Process a new weekly schedule and generate a clean, standalone Markdown block for only that new week. Your output must match the established structure of the existing schedule source.
>
> Your output MUST be only the raw Markdown text for this new block.
>
> Step 1: Parse using the Internal Context and Glossary below.
> Step 2: Generate the weekly block:
> 1. Main Heading: "Week of [Month Day, Year]"
> 2. Announcements and Student Assignments
> 3. Resident and Fellow Assignments (bolded names, bulleted per day)
> 4. Student Assignments
> 5. Uncovered Assignments (parse per rules below)
> 6. Conferences and Events
> 7. On-Call Schedule

> Internal Context and Glossary (customize for your program):
> Roles: Admin = Administrative day. [Add your codes.]
> Locations: [Map your OR/clinic abbreviations.]
> Uncovered Assignment Rule: [Define parsing rules.]

To run:

- NEW WEEKLY EMAIL TEXT: Paste the coordinator's weekly email.
- NEW WEEKLY SCHEDULE TABLE: Paste from the schedule PDF or spreadsheet.

Prompt 5: Consolidating Multiple Old Sources

Use for periodic cleanup: merging monthly schedules into one annual file, combining outdated sources into a single current reference, or any situation where source sprawl is creating redundancy.

The Prompt:

> You are an AI data architect for a residency program's operational knowledge hub. Consolidate multiple related source files into a single, new, comprehensive master source file by intelligently merging the data and removing redundant information.
>
> Follow all structural rules from the source creation process (Header, Metadata, Anticipated Questions, Contextual Interpretation Guide, Main Content, End Marker).
>
> Merge chronologically. De-duplicate. Update Header and Metadata for the combined content.
>
> Your output MUST be only the raw Markdown text. Do not include in-line citations or explanatory text.

To run: Paste the full text of all related sources below, one after another.

PART 3: Keeping It Running

The hub decays if nobody maintains it. We learned that early. As a reference, below is what our maintenance workflow looks like for our program.

Cadence

Weekly, when the new coverage schedule email arrives (usually Friday), our program coordinator runs Prompt 4 to generate the new week's block and appends it to the master schedule file in NotebookLM. If new faculty away dates came in that week, she runs Prompt 3 to merge them into the calendar. Total weekly time: 5 to 10 minutes.

Small edits come up throughout the month: a contact name changes, a policy gets revised, a door code rotates. Those are Prompt 2. Two to five minutes each, as they arise.

Quarterly, we run Prompt 5 to consolidate accumulated monthly schedule files and review the source list for anything stale or superseded. About 15 to 20 minutes.

Ownership

Our coordinator owns routine maintenance, with PD and APD content oversight. All residents are encouraged to flag inaccuracies when identified from the user side and submit update requests to the coordinator or resident power users. We started with this model at launch and have been broadening edit access to resident leaders for high-churn live-sync sources (service guides via Google Docs) as trust has built, so that residents have the ability to keep things maintainable in real time.

Your program's model will depend on your coordinator's capacity and your residents' engagement. The constraint we would flag: locking maintenance permanently to one role creates a single point of failure when that person is on leave. Opening contribution too early creates conflicting edits and inconsistent metadata. We moved gradually, which for our program, has worked well. This document is not a recipe. It is how we set ours up, the prompts we use, and the ownership model that works for us. Adapt what fits your program.

Free to use, adapt, and share.

jason.joseph@urology.ufl.edu