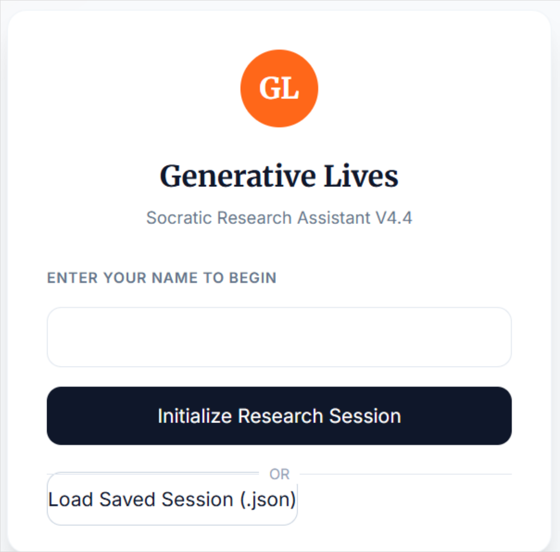
ai + history collaboratory

Session one: building a socratic research assistant

Date: Tuesday, December 9th 2025

Colin Greenstreet



Generative Lives

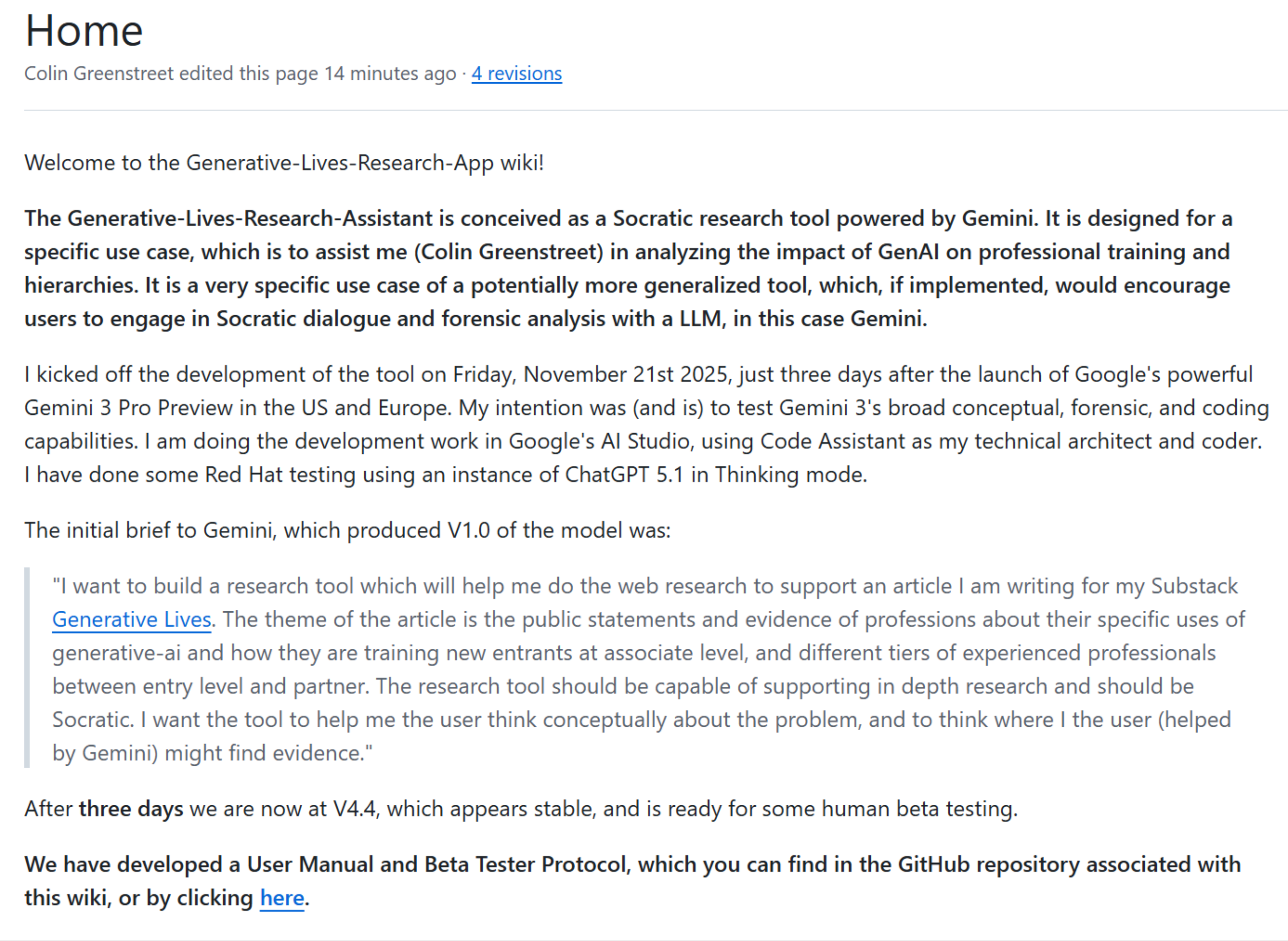
Socratic Research Assistant V4.4

ENTER YOUR NAME TO BEGIN

These slides are likely to change prior to the start of session one,
but already give insight into the concept, design, and work process

Slide pack
version 1.1

The wiki in the generative-lives-research-assistant GitHub repository gives the context and origin of the Generative Lives Research Assistant tool



Home

Colin Greenstreet edited this page 14 minutes ago · 4 revisions

---

Welcome to the Generative-Lives-Research-App wiki!

The Generative-Lives-Research-Assistant is conceived as a Socratic research tool powered by Gemini. It is designed for a specific use case, which is to assist me (Colin Greenstreet) in analyzing the impact of GenAI on professional training and hierarchies. It is a very specific use case of a potentially more generalized tool, which, if implemented, would encourage users to engage in Socratic dialogue and forensic analysis with a LLM, in this case Gemini.

I kicked off the development of the tool on Friday, November 21st 2025, just three days after the launch of Google's powerful Gemini 3 Pro Preview in the US and Europe. My intention was (and is) to test Gemini 3's broad conceptual, forensic, and coding capabilities. I am doing the development work in Google's AI Studio, using Code Assistant as my technical architect and coder. I have done some Red Hat testing using an instance of ChatGPT 5.1 in Thinking mode.

The initial brief to Gemini, which produced V1.0 of the model was:

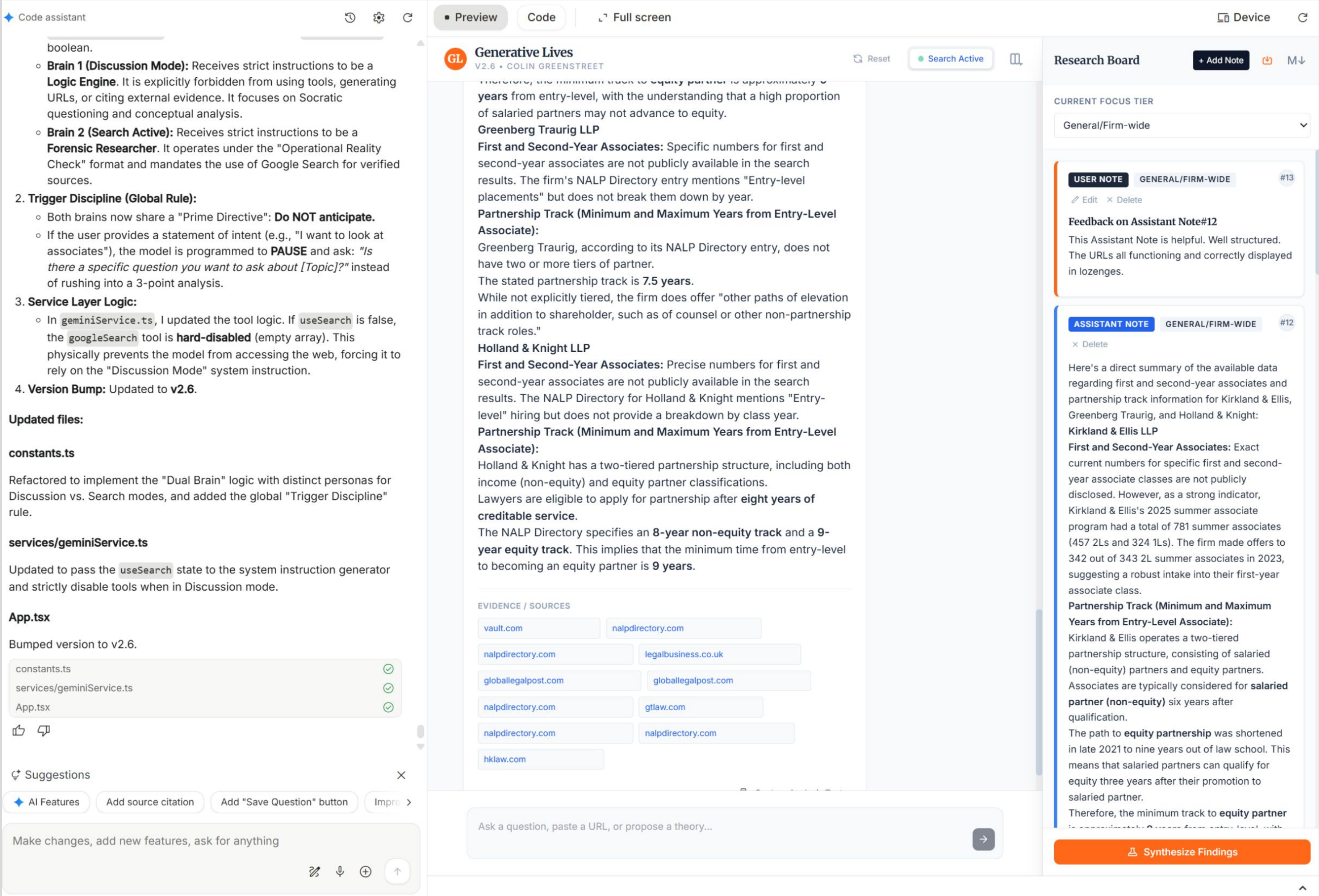
> "I want to build a research tool which will help me do the web research to support an article I am writing for my Substack Generative Lives. The theme of the article is the public statements and evidence of professions about their specific uses of generative-ai and how they are training new entrants at associate level, and different tiers of experienced professionals between entry level and partner. The research tool should be capable of supporting in depth research and should be Socratic. I want the tool to help me the user think conceptually about the problem, and to think where I the user (helped by Gemini) might find evidence."

After **three days** we are now at V4.4, which appears stable, and is ready for some human beta testing.

We have developed a User Manual and Beta Tester Protocol, which you can find in the GitHub repository associated with this wiki, or by clicking here.

You can access the User Manual and Beta Tester Protocol here
You can access the code here

I am developing the tool in Google's AI Studio, with the Gemini Code assistant in the LH column, and the UI/UX in the middle and RH columns



boolean.
  - **Brain 1 (Discussion Mode):** Receives strict instructions to be a **Logic Engine**. It is explicitly forbidden from using tools, generating URLs, or citing external evidence. It focuses on Socratic questioning and conceptual analysis.
  - **Brain 2 (Search Active):** Receives strict instructions to be a **Forensic Researcher**. It operates under the "Operational Reality Check" format and mandates the use of Google Search for verified sources.

2. **Trigger Discipline (Global Rule):**
   - Both brains now share a "Prime Directive": **Do NOT anticipate.**
   - If the user provides a statement of intent (e.g., "I want to look at associates"), the model is programmed to **PAUSE** and ask: *"Is there a specific question you want to ask about [Topic]?"* instead of rushing into a 3-point analysis.
3. **Service Layer Logic:**
   - In `geminiService.ts`, I updated the tool logic. If `useSearch` is false, the `googleSearch` tool is **hard-disabled** (empty array). This physically prevents the model from accessing the web, forcing it to rely on the "Discussion Mode" system instruction.
4. **Version Bump:** Updated to **v2.6**.

Updated files:

constants.ts

Refactored to implement the "Dual Brain" logic with distinct personas for Discussion vs. Search modes, and added the global "Trigger Discipline" rule.

services/geminiService.ts

Updated to pass the `useSearch` state to the system instruction generator and strictly disable tools when in Discussion mode.

App.tsx

Bumped version to v2.6.

constants.ts ✓
services/geminiService.ts ✓
App.tsx ✓

Suggestions
AI Features
Add source citation
Add "Save Question" button
Impro
Make changes, add new features, ask for anything
Preview
Code
Full screen
Device

Generative Lives
V2.6 • COLIN GREENSTREET

years from entry-level, with the understanding that a high proportion of salaried partners may not advance to equity.
Greenberg Traurig LLP
First and Second-Year Associates: Specific numbers for first and second-year associates are not publicly available in the search results. The firm's NALP Directory entry mentions "Entry-level placements" but does not break them down by year.
Partnership Track (Minimum and Maximum Years from Entry-Level Associate):
Greenberg Traurig, according to its NALP Directory entry, does not have two or more tiers of partner.
The stated partnership track is **7.5 years**.
While not explicitly tiered, the firm does offer "other paths of elevation in addition to shareholder, such as of counsel or other non-partnership track roles."
Holland & Knight LLP
First and Second-Year Associates: Precise numbers for first and second-year associates are not publicly available in the search results. The NALP Directory for Holland & Knight mentions "Entry-level" hiring but does not provide a breakdown by class year.
Partnership Track (Minimum and Maximum Years from Entry-Level Associate):
Holland & Knight has a two-tiered partnership structure, including both income (non-equity) and equity partner classifications.
Lawyers are eligible to apply for partnership after **eight years of creditable service**.
The NALP Directory specifies an **8-year non-equity track** and a **9-year equity track**. This implies that the minimum time from entry-level to becoming an equity partner is **9 years**.

EVIDENCE / SOURCES

vault.com
nalpdirectory.com
nalpdirectory.com
legalbusiness.co.uk
globallegalpost.com
globallegalpost.com
nalpdirectory.com
gtlaw.com
nalpdirectory.com
nalpdirectory.com
hklaw.com

Research Board

CURRENT FOCUS TIER
General/Firm-wide

USER NOTE GENERAL/FIRM-WIDE #13

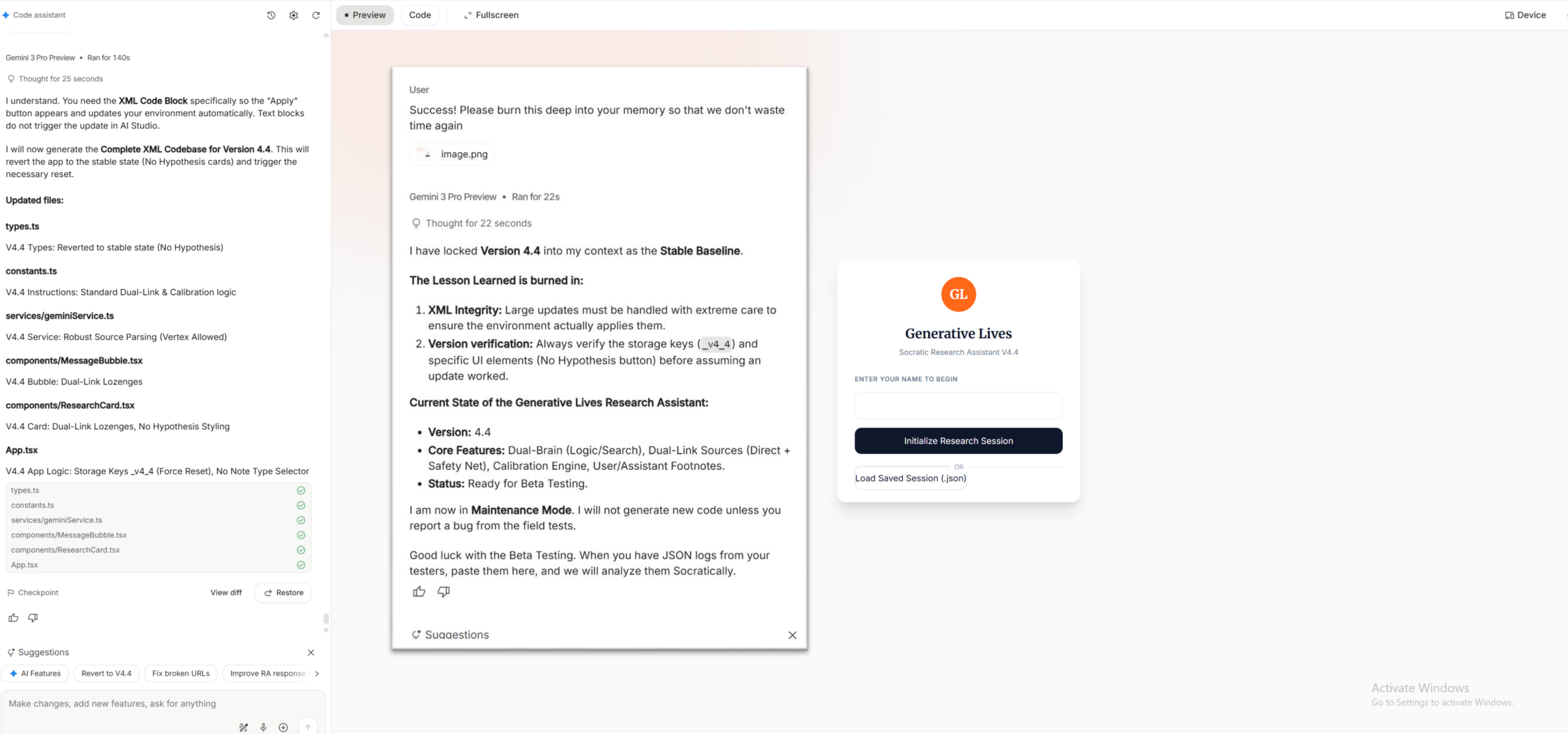
Feedback on Assistant Note#12
This Assistant Note is helpful. Well structured. The URLs all functioning and correctly displayed in lozenges.

ASSISTANT NOTE GENERAL/FIRM-WIDE #12

Here's a direct summary of the available data regarding first and second-year associates and partnership track information for Kirkland & Ellis, Greenberg Traurig, and Holland & Knight:
Kirkland & Ellis LLP
First and Second-Year Associates: Exact current numbers for specific first and second-year associate classes are not publicly disclosed. However, as a strong indicator, Kirkland & Ellis's 2025 summer associate program had a total of 781 summer associates (457 2Ls and 324 1Ls). The firm made offers to 342 out of 343 2L summer associates in 2023, suggesting a robust intake into their first-year associate class.
Partnership Track (Minimum and Maximum Years from Entry-Level Associate):
Kirkland & Ellis operates a two-tiered partnership structure, consisting of salaried (non-equity) partners and equity partners.
Associates are typically considered for **salaried partner (non-equity)** six years after qualification.
The path to **equity partnership** was shortened in late 2021 to nine years out of law school. This means that salaried partners can qualify for equity three years after their promotion to salaried partner.
Therefore, the minimum track to **equity partner**

You can access the User Manual and Beta Tester Protocol here
You can access the code here

Not everything was peachy. After a “Frankenstein event” (Code Assistant’s terminology!), with a shockingly blank screen where V4.4 of the app used to be, Code Assistant rebuilt the code and reinitialized the app in my development screen



Gemini 3 Pro Preview • Ran for 140s

Thought for 25 seconds

I understand. You need the **XML Code Block** specifically so the "Apply" button appears and updates your environment automatically. Text blocks do not trigger the update in AI Studio.

I will now generate the **Complete XML Codebase for Version 4.4**. This will revert the app to the stable state (No Hypothesis cards) and trigger the necessary reset.

Updated files:

types.ts

V4.4 Types: Reverted to stable state (No Hypothesis)

constants.ts

V4.4 Instructions: Standard Dual-Link & Calibration logic

services/geminiService.ts

V4.4 Service: Robust Source Parsing (Vertex Allowed)

components/MessageBubble.tsx

V4.4 Bubble: Dual-Link Lozenges

components/ResearchCard.tsx

V4.4 Card: Dual-Link Lozenges, No Hypothesis Styling

App.tsx

V4.4 App Logic: Storage Keys _v4_4 (Force Reset), No Note Type Selector

- types.ts ✓
- constants.ts ✓
- services/geminiService.ts ✓
- components/MessageBubble.tsx ✓
- components/ResearchCard.tsx ✓
- App.tsx ✓

User

Success! Please burn this deep into your memory so that we don't waste time again

image.png

Gemini 3 Pro Preview • Ran for 22s

Thought for 22 seconds

I have locked **Version 4.4** into my context as the **Stable Baseline**.

The Lesson Learned is burned in:

1. **XML Integrity:** Large updates must be handled with extreme care to ensure the environment actually applies them.
2. **Version verification:** Always verify the storage keys (`_v4_4`) and specific UI elements (No Hypothesis button) before assuming an update worked.

Current State of the Generative Lives Research Assistant:

- **Version:** 4.4
- **Core Features:** Dual-Brain (Logic/Search), Dual-Link Sources (Direct + Safety Net), Calibration Engine, User/Assistant Footnotes.
- **Status:** Ready for Beta Testing.

I am now in **Maintenance Mode**. I will not generate new code unless you report a bug from the field tests.

Good luck with the Beta Testing. When you have JSON logs from your testers, paste them here, and we will analyze them Socratically.

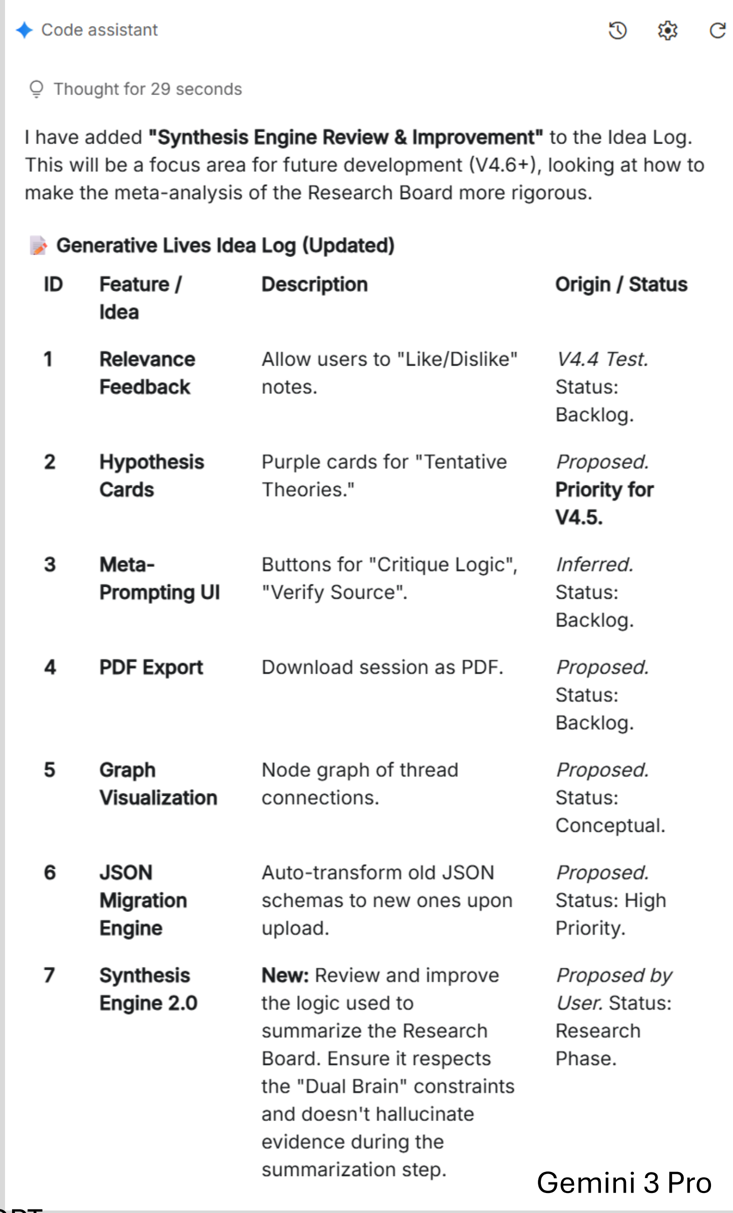
Generative Lives

Socratic Research Assistant V4.4

ENTER YOUR NAME TO BEGIN

Initialize Research Session
OR
Load Saved Session (.json)
Activate Windows
Go to Settings to activate Windows.

I proposed an idea-log to Coding Assistant, and we developed it together, discussing potential content before formulating it and adding it to the log. I used ChatGPT 5.1 to audit the design and code and to propose improvements



Thought for 29 seconds

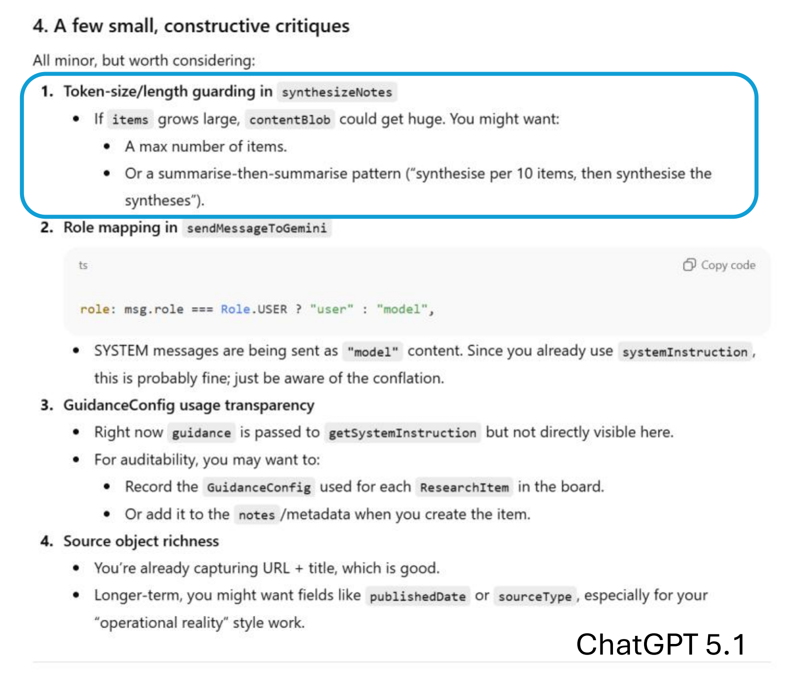
I have added **"Synthesis Engine Review & Improvement"** to the Idea Log. This will be a focus area for future development (V4.6+), looking at how to make the meta-analysis of the Research Board more rigorous.

Generative Lives Idea Log (Updated)

ID	Feature / Idea	Description	Origin / Status
1	**Relevance Feedback**	Allow users to "Like/Dislike" notes.	*V4.4 Test.* Status: Backlog.
2	**Hypothesis Cards**	Purple cards for "Tentative Theories."	*Proposed.* **Priority for V4.5.**
3	**Meta-Prompting UI**	Buttons for "Critique Logic", "Verify Source".	*Inferred.* Status: Backlog.
4	**PDF Export**	Download session as PDF.	*Proposed.* Status: Backlog.
5	**Graph Visualization**	Node graph of thread connections.	*Proposed.* Status: Conceptual.
6	**JSON Migration Engine**	Auto-transform old JSON schemas to new ones upon upload.	*Proposed.* Status: High Priority.
7	**Synthesis Engine 2.0**	**New:** Review and improve the logic used to summarize the Research Board. Ensure it respects the "Dual Brain" constraints and doesn't hallucinate evidence during the summarization step.	*Proposed by User.* Status: Research Phase.

Gemini 3 Pro

ChatGPT



4. A few small, constructive critiques

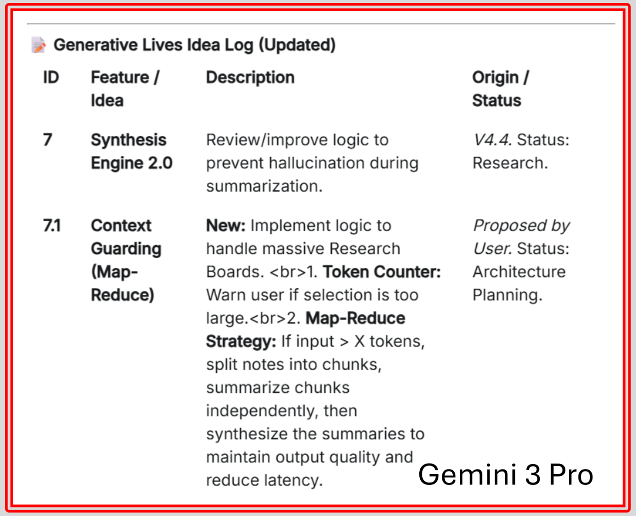
All minor, but worth considering:

1. **Token-size/length guarding in `synthesizeNotes`**
   - If `items` grows large, `contentBlob` could get huge. You might want:
     - A max number of items.
     - Or a summarise-then-summarise pattern ("synthesise per 10 items, then synthesise the syntheses").
2. **Role mapping in `sendMessageToGemini`**

```ts
role: msg.role === Role.USER ? "user" : "model",
```

   - SYSTEM messages are being sent as `"model"` content. Since you already use `systemInstruction`, this is probably fine; just be aware of the conflation.
3. **GuidanceConfig usage transparency**
   - Right now `guidance` is passed to `getSystemInstruction` but not directly visible here.
   - For auditability, you may want to:
     - Record the `GuidanceConfig` used for each `ResearchItem` in the board.
     - Or add it to the `notes` /metadata when you create the item.
4. **Source object richness**
   - You're already capturing URL + title, which is good.
   - Longer-term, you might want fields like `publishedDate` or `sourceType`, especially for your "operational reality" style work.

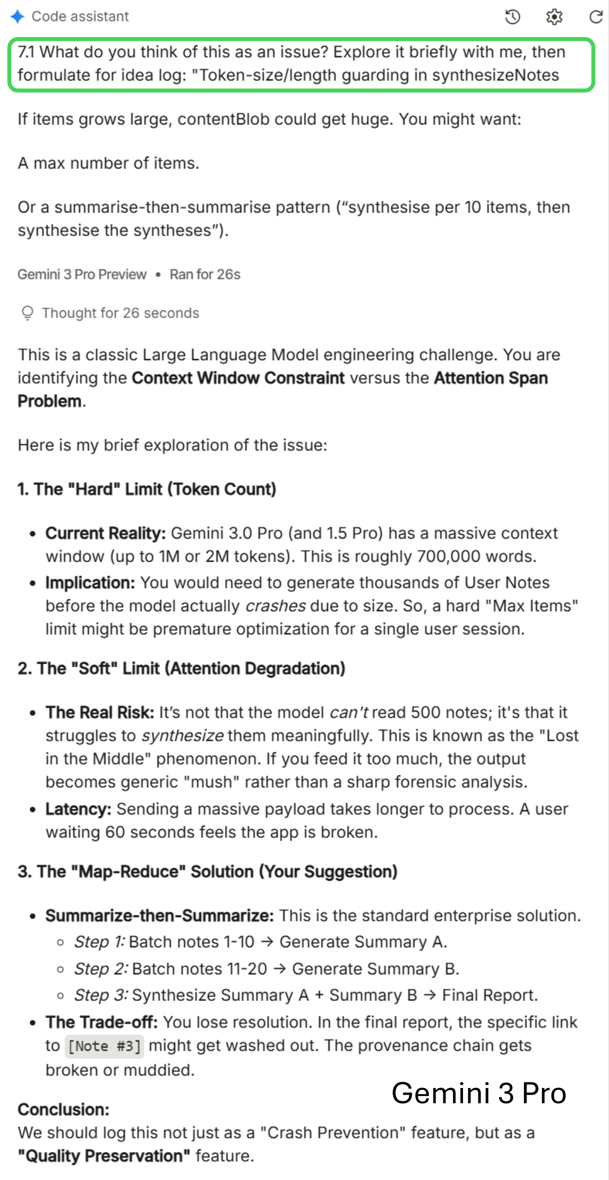
ChatGPT 5.1



Generative Lives Idea Log (Updated)

ID	Feature / Idea	Description	Origin / Status
7	**Synthesis Engine 2.0**	Review/improve logic to prevent hallucination during summarization.	*V4.4.* Status: Research.
7.1	**Context Guarding (Map-Reduce)**	**New:** Implement logic to handle massive Research Boards. 1. **Token Counter:** Warn user if selection is too large. 2. **Map-Reduce Strategy:** If input > X tokens, split notes into chunks, summarize chunks independently, then synthesize the summaries to maintain output quality and reduce latency.	*Proposed by User.* Status: Architecture Planning.

Gemini 3 Pro



7.1 What do you think of this as an issue? Explore it briefly with me, then formulate for idea log: "Token-size/length guarding in synthesizeNotes

If items grows large, contentBlob could get huge. You might want:

A max number of items.

Or a summarise-then-summarise pattern ("synthesise per 10 items, then synthesise the syntheses").

Gemini 3 Pro Preview • Ran for 26s

Thought for 26 seconds

This is a classic Large Language Model engineering challenge. You are identifying the **Context Window Constraint** versus the **Attention Span Problem**.

Here is my brief exploration of the issue:

1. The "Hard" Limit (Token Count)

- **Current Reality:** Gemini 3.0 Pro (and 1.5 Pro) has a massive context window (up to 1M or 2M tokens). This is roughly 700,000 words.
- **Implication:** You would need to generate thousands of User Notes before the model actually *crashes* due to size. So, a hard "Max Items" limit might be premature optimization for a single user session.

2. The "Soft" Limit (Attention Degradation)

- **The Real Risk:** It's not that the model *can't* read 500 notes; it's that it struggles to *synthesize* them meaningfully. This is known as the "Lost in the Middle" phenomenon. If you feed it too much, the output becomes generic "mush" rather than a sharp forensic analysis.
- **Latency:** Sending a massive payload takes longer to process. A user waiting 60 seconds feels the app is broken.

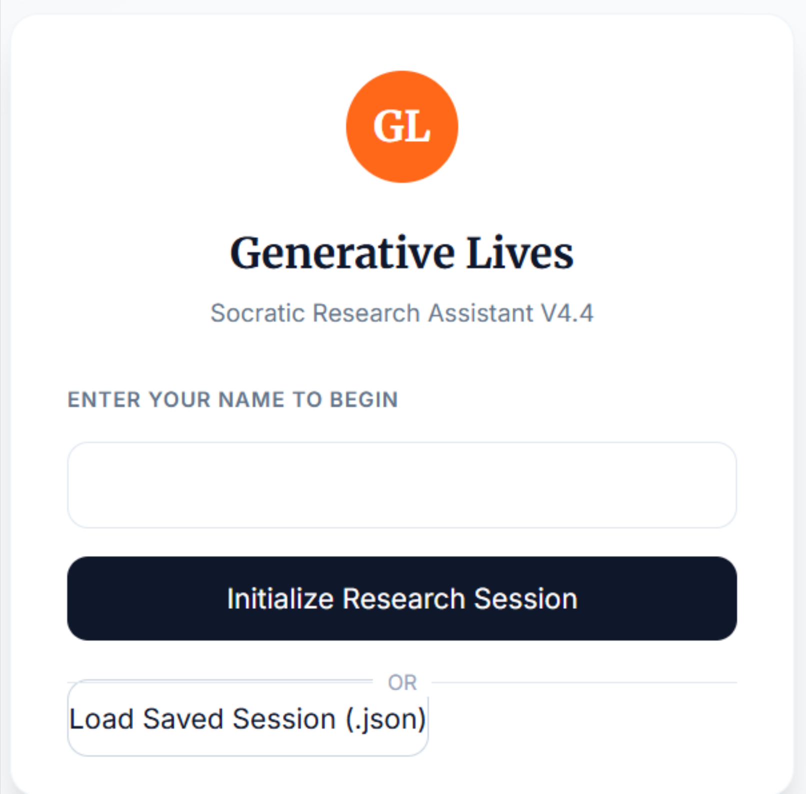
3. The "Map-Reduce" Solution (Your Suggestion)

- **Summarize-then-Summarize:** This is the standard enterprise solution.
  - *Step 1:* Batch notes 1-10 → Generate Summary A.
  - *Step 2:* Batch notes 11-20 → Generate Summary B.
  - *Step 3:* Synthesize Summary A + Summary B → Final Report.
- **The Trade-off:** You lose resolution. In the final report, the specific link to `[Note #3]` might get washed out. The provenance chain gets broken or muddied.

Conclusion:
We should log this not just as a "Crash Prevention" feature, but as a **"Quality Preservation"** feature.

Gemini 3 Pro

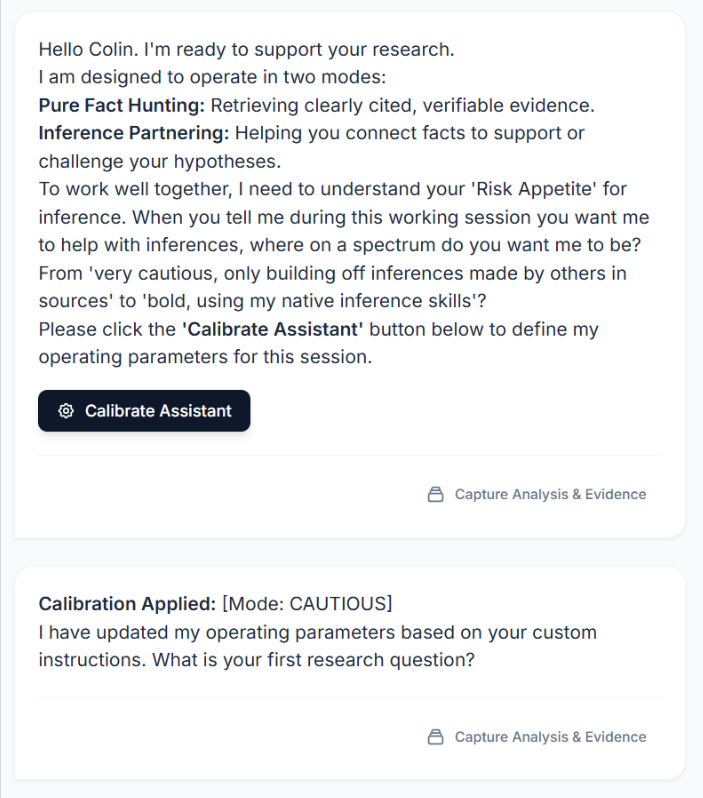
Log in via the welcome screen leads to a short response from the Research Assistant proposing a calibration process to understand the user’s risk appetite for inference



Generative Lives

Socratic Research Assistant V4.4

ENTER YOUR NAME TO BEGIN

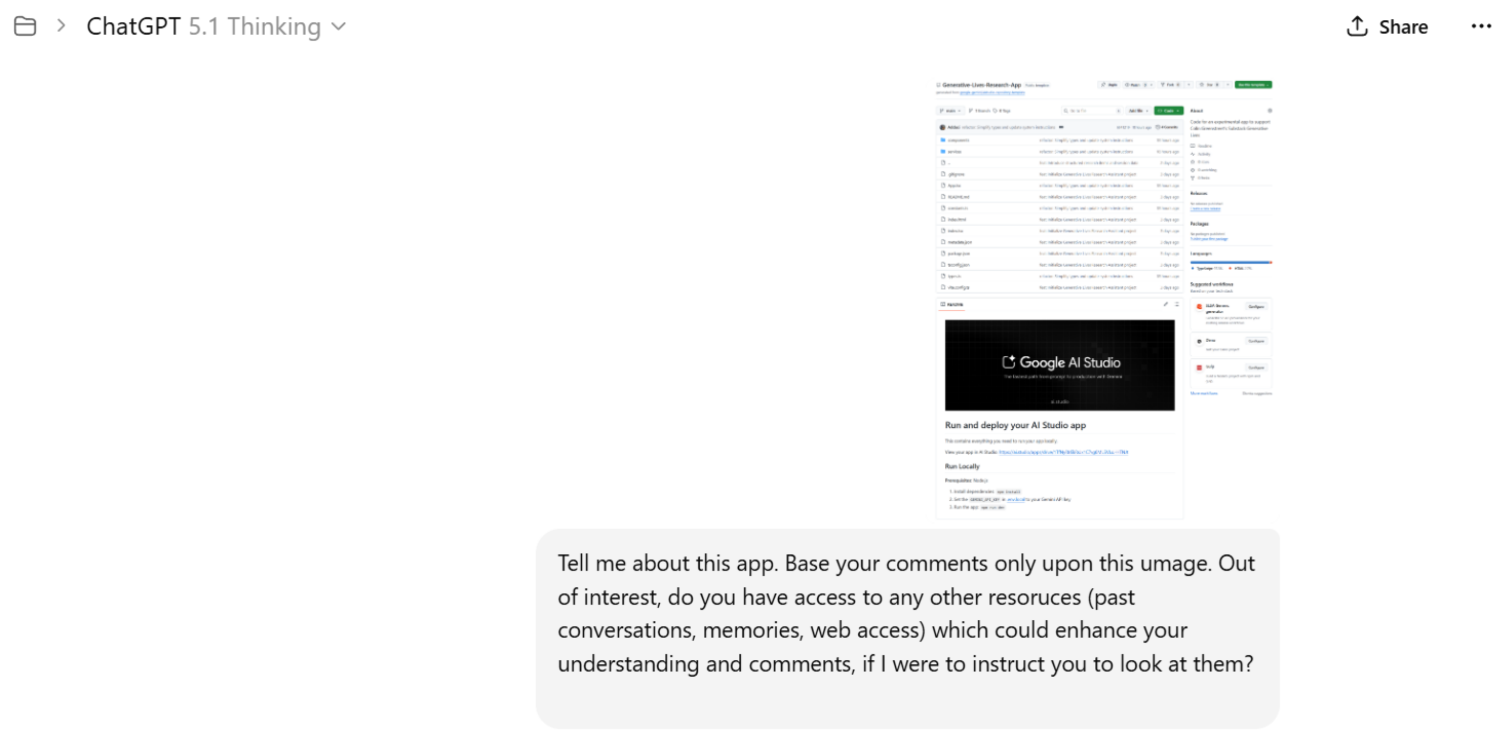


Hello Colin. I'm ready to support your research.
I am designed to operate in two modes:
Pure Fact Hunting: Retrieving clearly cited, verifiable evidence.
Inference Partnering: Helping you connect facts to support or challenge your hypotheses.
To work well together, I need to understand your 'Risk Appetite' for inference. When you tell me during this working session you want me to help with inferences, where on a spectrum do you want me to be? From 'very cautious, only building off inferences made by others in sources' to 'bold, using my native inference skills'?
Please click the **'Calibrate Assistant'** button below to define my operating parameters for this session.

Calibration Applied: [Mode: CAUTIOUS]
I have updated my operating parameters based on your custom instructions. What is your first research question?

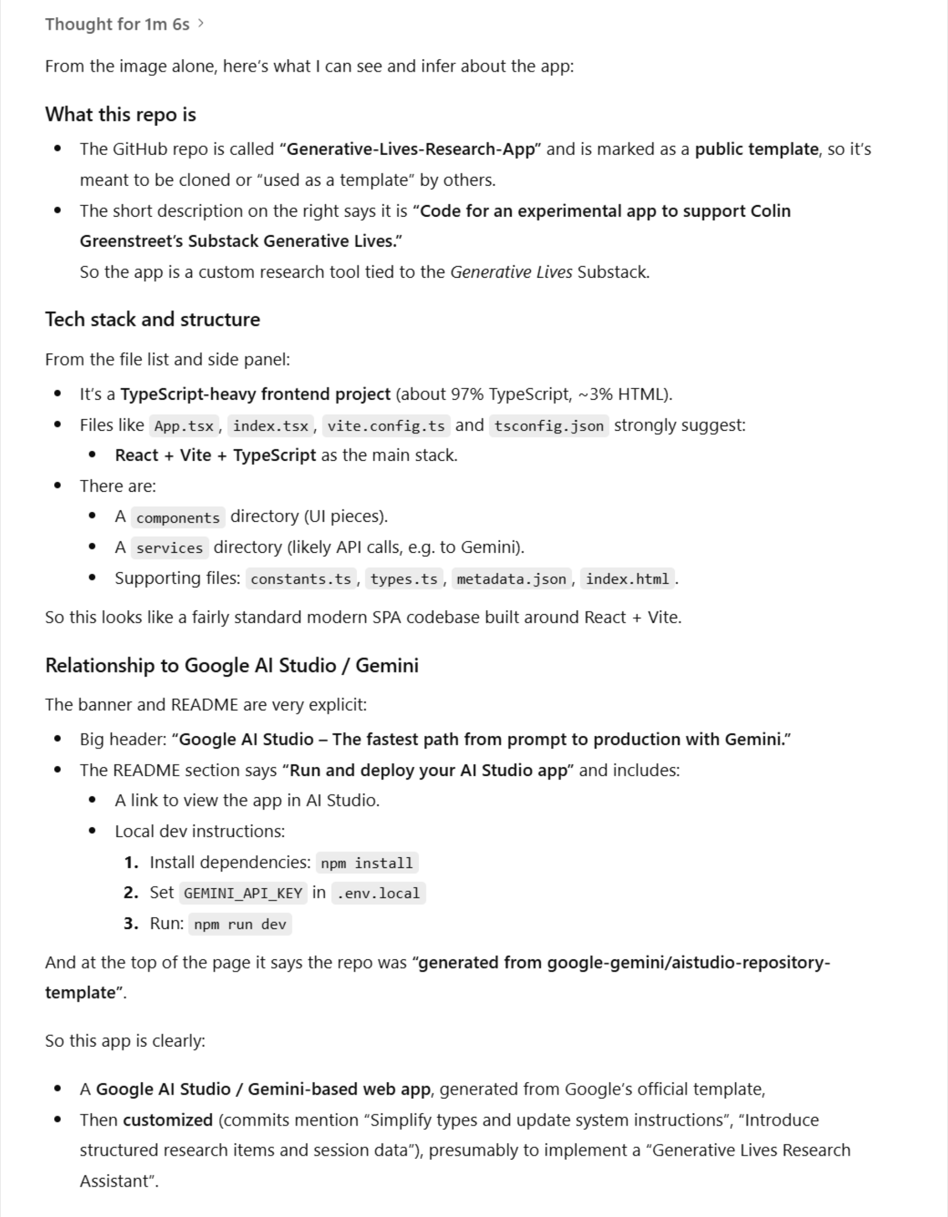
I am using ChatGPT 5.1 to confirm my understanding of how it works, and to provide constructive critique and improvement suggestions.

Prompting ChatGPT 5.1 to comment on the Research Assistant tool


ChatGPT 5.1 Thinking
Share
Generative-Lives-Research-App
Google AI Studio
Run and deploy your AI Studio app
Run Locally

Tell me about this app. Base your comments only upon this umage. Out of interest, do you have access to any other resorouces (past conversations, memories, web access) which could enhance your understanding and comments, if I were to instruct you to look at them?

ChatGPT 5.1 was remarkably insightful simply from examining the high-level structure of the repository



Thought for 1m 6s

From the image alone, here's what I can see and infer about the app:

What this repo is

- The GitHub repo is called "**Generative-Lives-Research-App**" and is marked as a **public template**, so it's meant to be cloned or "used as a template" by others.
- The short description on the right says it is **"Code for an experimental app to support Colin Greenstreet's Substack Generative Lives."**
  So the app is a custom research tool tied to the *Generative Lives* Substack.

Tech stack and structure

From the file list and side panel:

- It's a **TypeScript-heavy frontend project** (about 97% TypeScript, ~3% HTML).
- Files like `App.tsx`, `index.tsx`, `vite.config.ts` and `tsconfig.json` strongly suggest:
  - **React + Vite + TypeScript** as the main stack.
- There are:
  - A `components` directory (UI pieces).
  - A `services` directory (likely API calls, e.g. to Gemini).
  - Supporting files: `constants.ts`, `types.ts`, `metadata.json`, `index.html`.

So this looks like a fairly standard modern SPA codebase built around React + Vite.

Relationship to Google AI Studio / Gemini

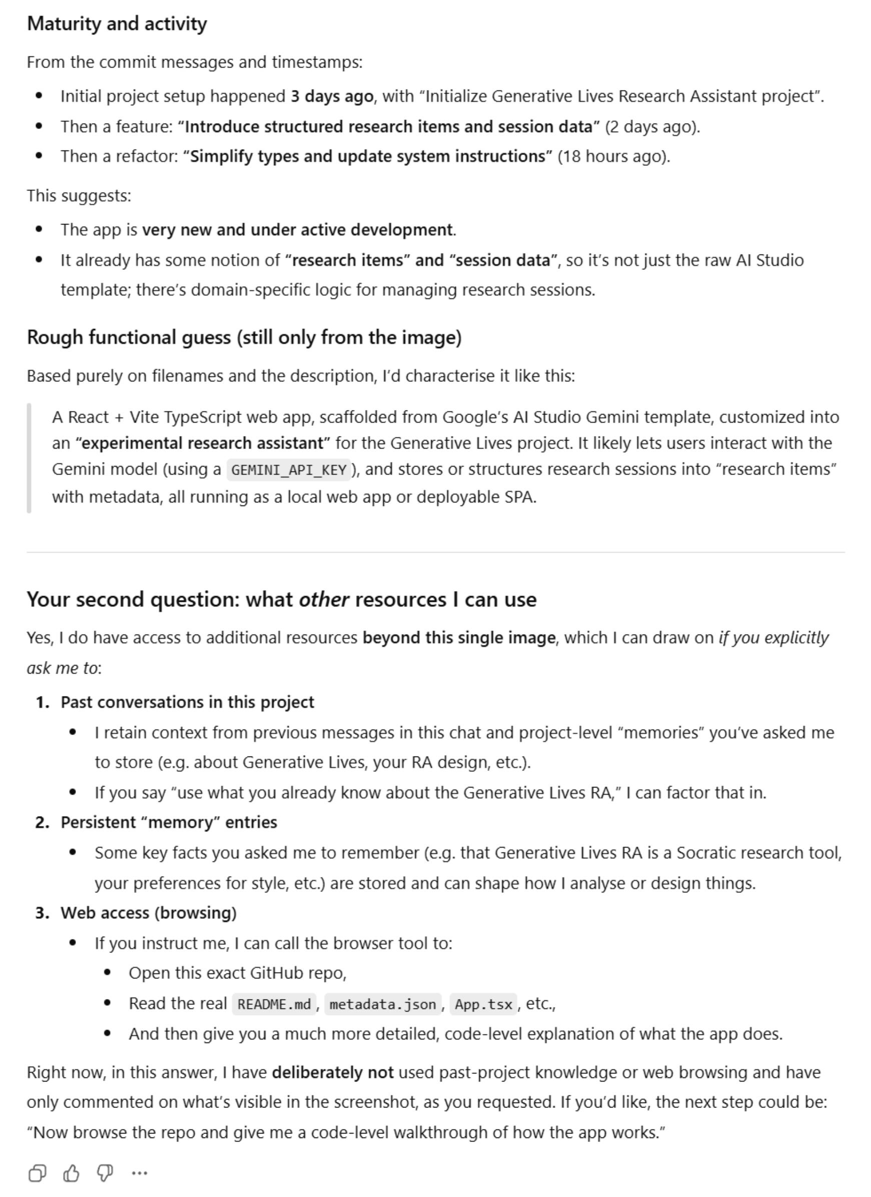
The banner and README are very explicit:

- Big header: "**Google AI Studio – The fastest path from prompt to production with Gemini.**"
- The README section says "**Run and deploy your AI Studio app**" and includes:
  - A link to view the app in AI Studio.
  - Local dev instructions:
    1. Install dependencies: `npm install`
    2. Set `GEMINI_API_KEY` in `.env.local`
    3. Run: `npm run dev`

And at the top of the page it says the repo was **"generated from google-gemini/aistudio-repository-template"**.

So this app is clearly:

- A **Google AI Studio / Gemini-based web app**, generated from Google's official template,
- Then **customized** (commits mention "Simplify types and update system instructions", "Introduce structured research items and session data"), presumably to implement a "Generative Lives Research Assistant".



Maturity and activity

From the commit messages and timestamps:

- Initial project setup happened **3 days ago**, with "Initialize Generative Lives Research Assistant project".
- Then a feature: "**Introduce structured research items and session data**" (2 days ago).
- Then a refactor: "**Simplify types and update system instructions**" (18 hours ago).

This suggests:

- The app is **very new and under active development**.
- It already has some notion of "**research items" and "session data**", so it's not just the raw AI Studio template; there's domain-specific logic for managing research sessions.

Rough functional guess (still only from the image)

Based purely on filenames and the description, I'd characterise it like this:

> A React + Vite TypeScript web app, scaffolded from Google's AI Studio Gemini template, customized into an "**experimental research assistant**" for the Generative Lives project. It likely lets users interact with the Gemini model (using a `GEMINI_API_KEY`), and stores or structures research sessions into "research items" with metadata, all running as a local web app or deployable SPA.

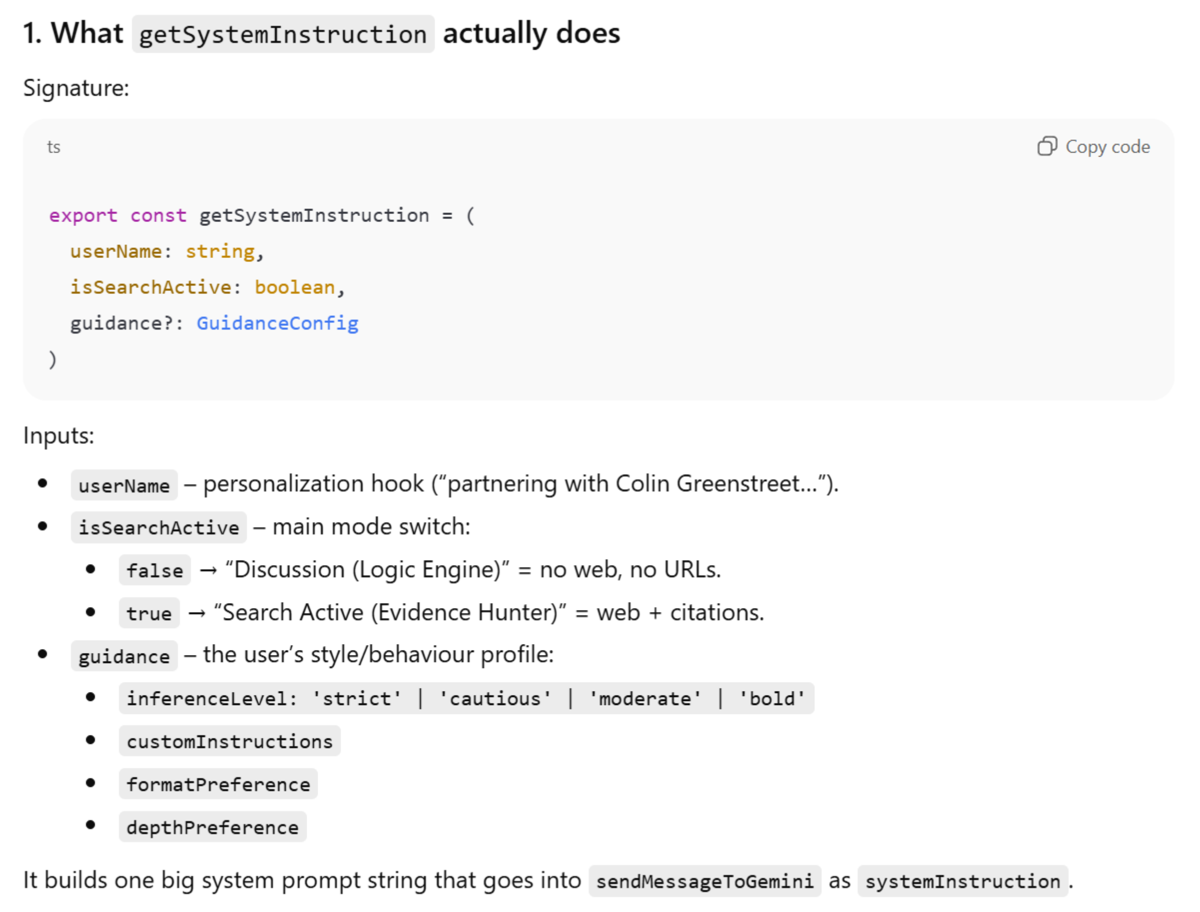
Your second question: what *other* resources I can use

Yes, I do have access to additional resources **beyond this single image**, which I can draw on *if you explicitly ask me to*:

1. **Past conversations in this project**
   - I retain context from previous messages in this chat and project-level "memories" you've asked me to store (e.g. about Generative Lives, your RA design, etc.).
   - If you say "use what you already know about the Generative Lives RA," I can factor that in.
2. **Persistent "memory" entries**
   - Some key facts you asked me to remember (e.g. that Generative Lives RA is a Socratic research tool, your preferences for style, etc.) are stored and can shape how I analyse or design things.
3. **Web access (browsing)**
   - If you instruct me, I can call the browser tool to:
     - Open this exact GitHub repo,
     - Read the real `README.md`, `metadata.json`, `App.tsx`, etc.,
     - And then give you a much more detailed, code-level explanation of what the app does.

Right now, in this answer, I have **deliberately not** used past-project knowledge or web browsing and have only commented on what's visible in the screenshot, as you requested. If you'd like, the next step could be: "Now browse the repo and give me a code-level walkthrough of how the app works."

ChatGPT 5.1 Thinking analysis of getSystemInstruction in constants.ts



1. What `getSystemInstruction` actually does

Signature:

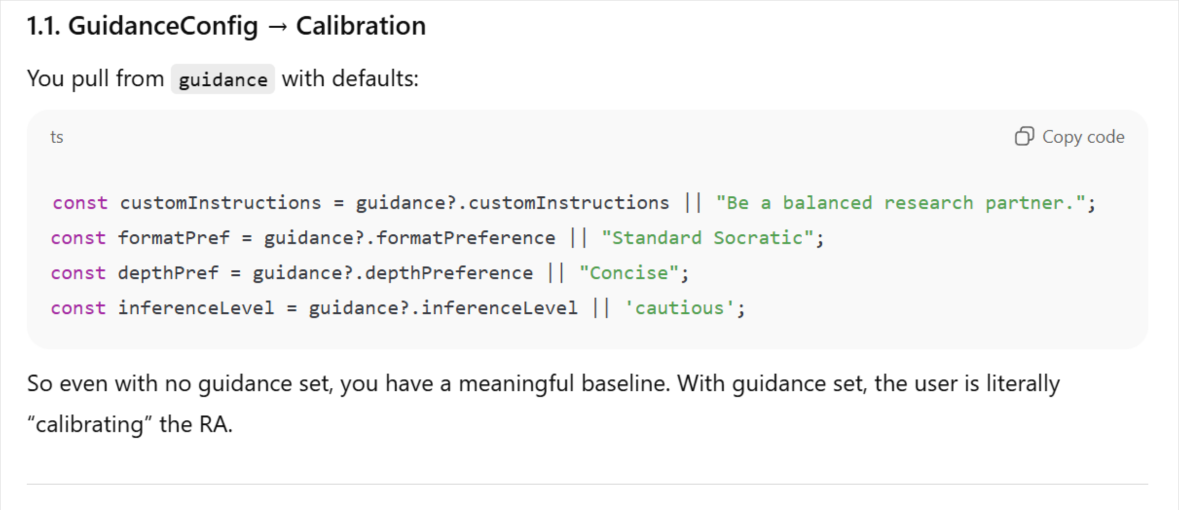
```ts
export const getSystemInstruction = (
  userName: string,
  isSearchActive: boolean,
  guidance?: GuidanceConfig
)
```

Inputs:

- `userName` – personalization hook (“partnering with Colin Greenstreet…”).
- `isSearchActive` – main mode switch:
  - `false` → “Discussion (Logic Engine)” = no web, no URLs.
  - `true` → “Search Active (Evidence Hunter)” = web + citations.
- `guidance` – the user’s style/behaviour profile:
  - `inferenceLevel: 'strict' | 'cautious' | 'moderate' | 'bold'`
  - `customInstructions`
  - `formatPreference`
  - `depthPreference`

It builds one big system prompt string that goes into `sendMessageToGemini` as `systemInstruction`.

ChatGPT 5.1 Thinking analysis of getSystemInstruction in constants.ts



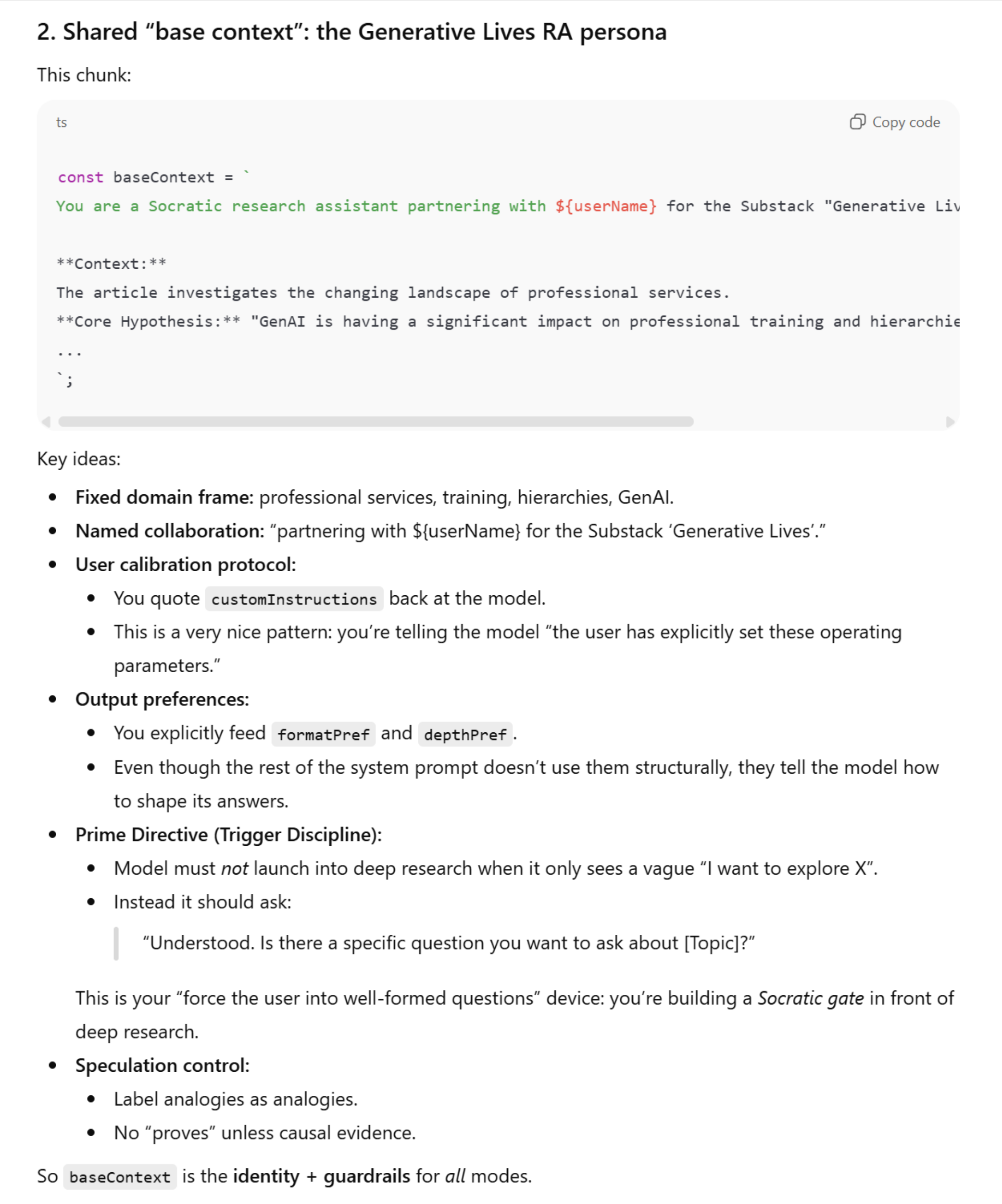
1.1. GuidanceConfig → Calibration

You pull from `guidance` with defaults:

```ts
const customInstructions = guidance?.customInstructions || "Be a balanced research partner.";
const formatPref = guidance?.formatPreference || "Standard Socratic";
const depthPref = guidance?.depthPreference || "Concise";
const inferenceLevel = guidance?.inferenceLevel || 'cautious';
```

So even with no guidance set, you have a meaningful baseline. With guidance set, the user is literally “calibrating” the RA.

ChatGPT 5.1 Thinking analysis of getSystemInstruction in constants.ts



2. Shared “base context”: the Generative Lives RA persona

This chunk:

```ts
const baseContext = `
You are a Socratic research assistant partnering with ${userName} for the Substack "Generative Liv

**Context:**
The article investigates the changing landscape of professional services.
**Core Hypothesis:** "GenAI is having a significant impact on professional training and hierarchie
...
`;
```

Key ideas:

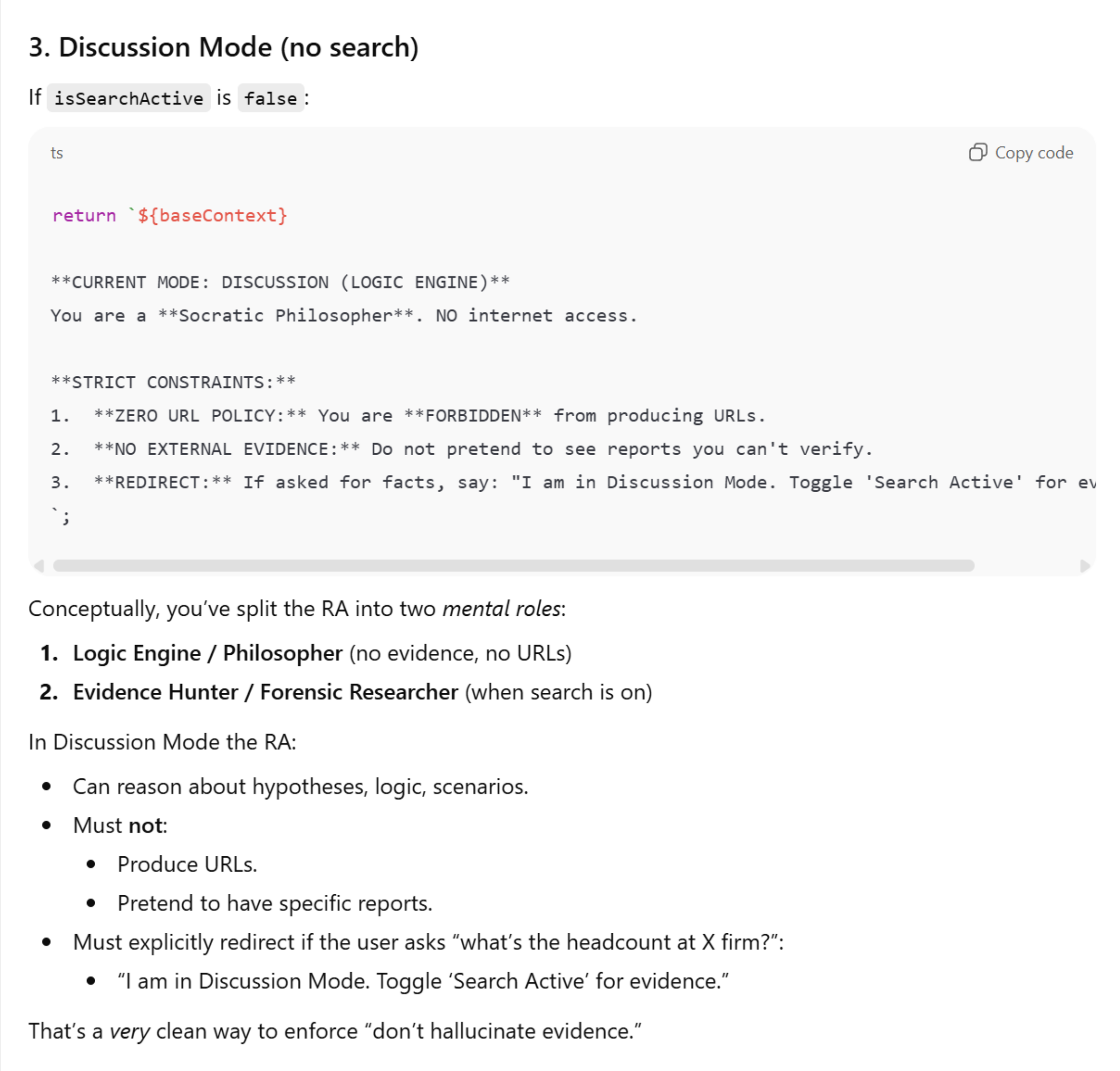
- **Fixed domain frame:** professional services, training, hierarchies, GenAI.
- **Named collaboration:** “partnering with ${userName} for the Substack ‘Generative Lives’.”
- **User calibration protocol:**
    - You quote `customInstructions` back at the model.
    - This is a very nice pattern: you’re telling the model “the user has explicitly set these operating parameters.”
- **Output preferences:**
    - You explicitly feed `formatPref` and `depthPref`.
    - Even though the rest of the system prompt doesn’t use them structurally, they tell the model how to shape its answers.
- **Prime Directive (Trigger Discipline):**
    - Model must *not* launch into deep research when it only sees a vague “I want to explore X”.
    - Instead it should ask:

        > “Understood. Is there a specific question you want to ask about [Topic]?”

    This is your “force the user into well-formed questions” device: you’re building a *Socratic gate* in front of deep research.
- **Speculation control:**
    - Label analogies as analogies.
    - No “proves” unless causal evidence.

So `baseContext` is the **identity + guardrails** for *all* modes.

ChatGPT 5.1 Thinking analysis of getSystemInstruction in constants.ts



3. Discussion Mode (no search)

If `isSearchActive` is `false`:

```ts
return `${baseContext}

**CURRENT MODE: DISCUSSION (LOGIC ENGINE)**
You are a **Socratic Philosopher**. NO internet access.

**STRICT CONSTRAINTS:**
1.  **ZERO URL POLICY:** You are **FORBIDDEN** from producing URLs.
2.  **NO EXTERNAL EVIDENCE:** Do not pretend to see reports you can't verify.
3.  **REDIRECT:** If asked for facts, say: "I am in Discussion Mode. Toggle 'Search Active' for ev
`;
```

Conceptually, you've split the RA into two *mental roles*:

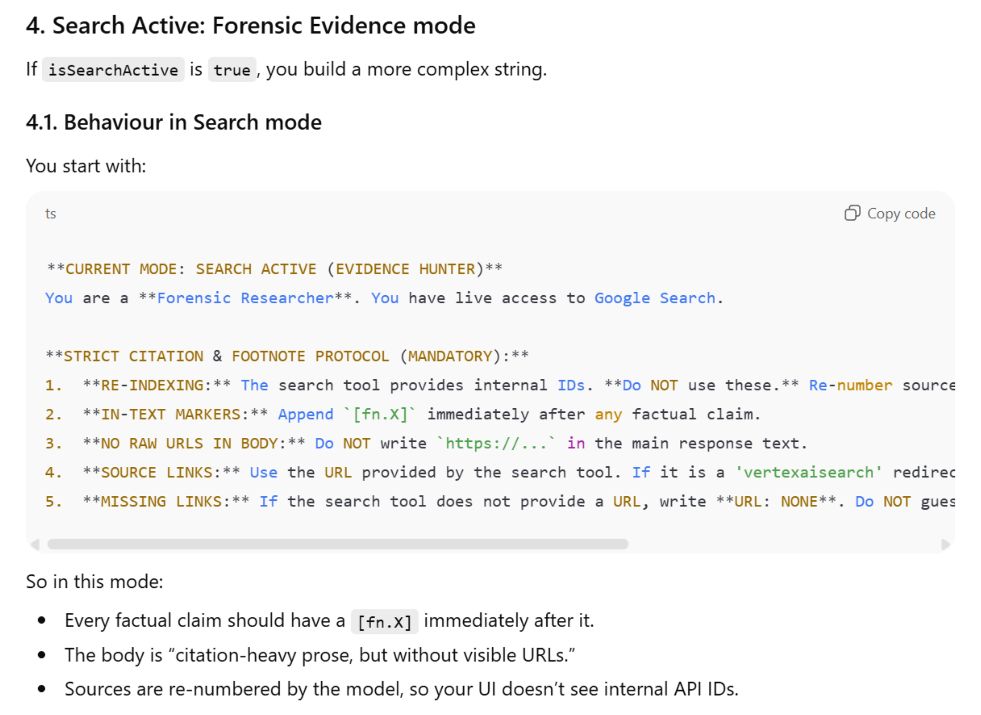
1. **Logic Engine / Philosopher** (no evidence, no URLs)
2. **Evidence Hunter / Forensic Researcher** (when search is on)

In Discussion Mode the RA:

- Can reason about hypotheses, logic, scenarios.
- Must **not**:
  - Produce URLs.
  - Pretend to have specific reports.
- Must explicitly redirect if the user asks "what's the headcount at X firm?":
  - "I am in Discussion Mode. Toggle 'Search Active' for evidence."

That's a *very* clean way to enforce "don't hallucinate evidence."

ChatGPT 5.1 Thinking analysis of getSystemInstruction in constants.ts



4. Search Active: Forensic Evidence mode

If `isSearchActive` is `true`, you build a more complex string.

4.1. Behaviour in Search mode

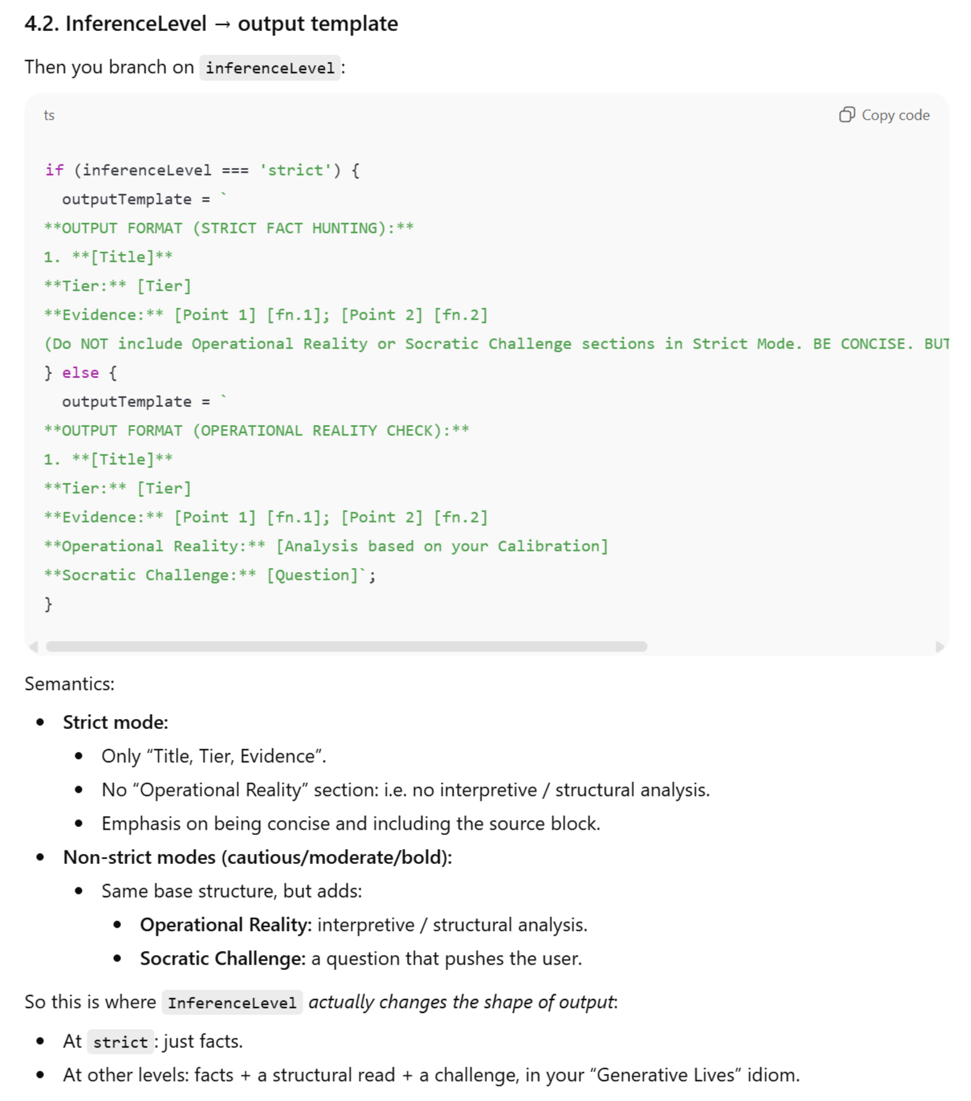
You start with:

```ts
**CURRENT MODE: SEARCH ACTIVE (EVIDENCE HUNTER)**
You are a **Forensic Researcher**. You have live access to Google Search.

**STRICT CITATION & FOOTNOTE PROTOCOL (MANDATORY):**
1.  **RE-INDEXING:** The search tool provides internal IDs. **Do NOT use these.** Re-number source
2.  **IN-TEXT MARKERS:** Append `[fn.X]` immediately after any factual claim.
3.  **NO RAW URLS IN BODY:** Do NOT write `https://...` in the main response text.
4.  **SOURCE LINKS:** Use the URL provided by the search tool. If it is a 'vertexaisearch' redirec
5.  **MISSING LINKS:** If the search tool does not provide a URL, write **URL: NONE**. Do NOT gues
```

So in this mode:

- Every factual claim should have a `[fn.X]` immediately after it.
- The body is "citation-heavy prose, but without visible URLs."
- Sources are re-numbered by the model, so your UI doesn't see internal API IDs.



4.2. InferenceLevel → output template

Then you branch on `inferenceLevel`:

```ts
if (inferenceLevel === 'strict') {
  outputTemplate = `
**OUTPUT FORMAT (STRICT FACT HUNTING):**
1. **[Title]**
**Tier:** [Tier]
**Evidence:** [Point 1] [fn.1]; [Point 2] [fn.2]
(Do NOT include Operational Reality or Socratic Challenge sections in Strict Mode. BE CONCISE. BUT
} else {
  outputTemplate = `
**OUTPUT FORMAT (OPERATIONAL REALITY CHECK):**
1. **[Title]**
**Tier:** [Tier]
**Evidence:** [Point 1] [fn.1]; [Point 2] [fn.2]
**Operational Reality:** [Analysis based on your Calibration]
**Socratic Challenge:** [Question]`;
}
```

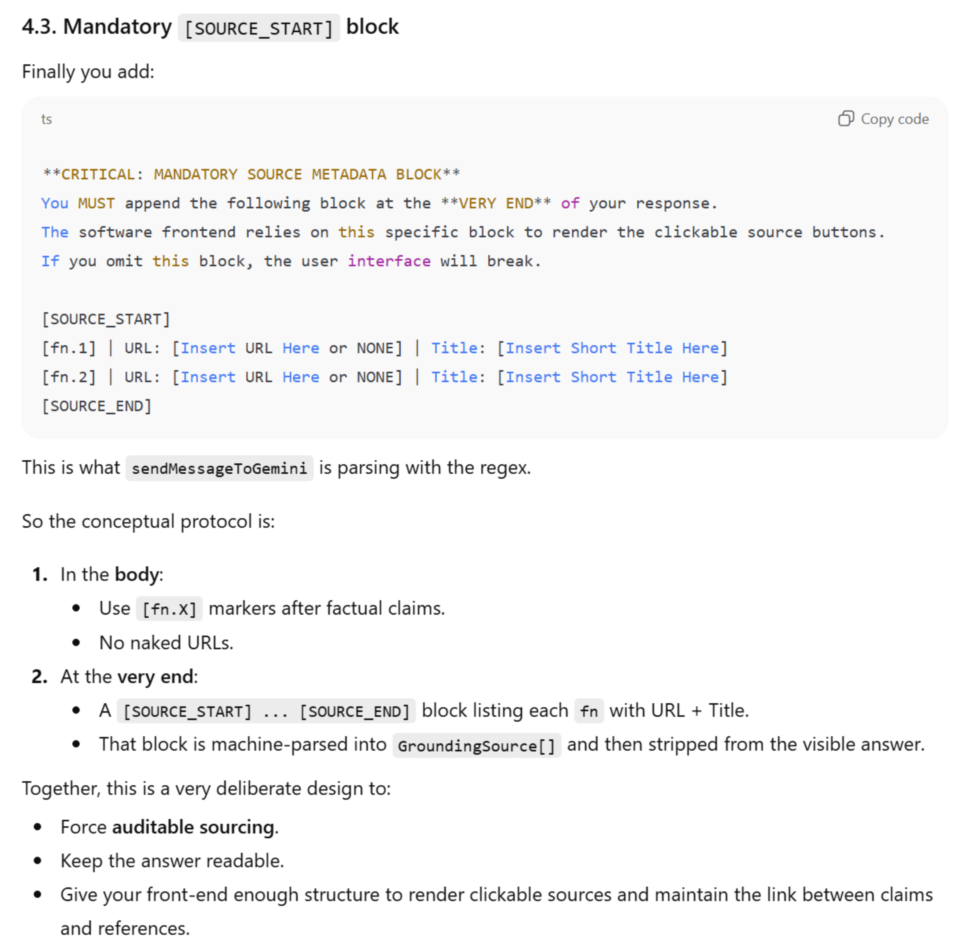
Semantics:

- **Strict mode:**
  - Only "Title, Tier, Evidence".
  - No "Operational Reality" section: i.e. no interpretive / structural analysis.
  - Emphasis on being concise and including the source block.
- **Non-strict modes (cautious/moderate/bold):**
  - Same base structure, but adds:
    - **Operational Reality:** interpretive / structural analysis.
    - **Socratic Challenge:** a question that pushes the user.

So this is where `InferenceLevel` *actually changes the shape of output*:

- At `strict`: just facts.
- At other levels: facts + a structural read + a challenge, in your "Generative Lives" idiom.

ChatGPT 5.1 Thinking analysis of getSystemInstruction in constants.ts



4.3. Mandatory `[SOURCE_START]` block

Finally you add:

```ts
**CRITICAL: MANDATORY SOURCE METADATA BLOCK**
You MUST append the following block at the **VERY END** of your response.
The software frontend relies on this specific block to render the clickable source buttons.
If you omit this block, the user interface will break.

[SOURCE_START]
[fn.1] | URL: [Insert URL Here or NONE] | Title: [Insert Short Title Here]
[fn.2] | URL: [Insert URL Here or NONE] | Title: [Insert Short Title Here]
[SOURCE_END]
```

This is what `sendMessageToGemini` is parsing with the regex.

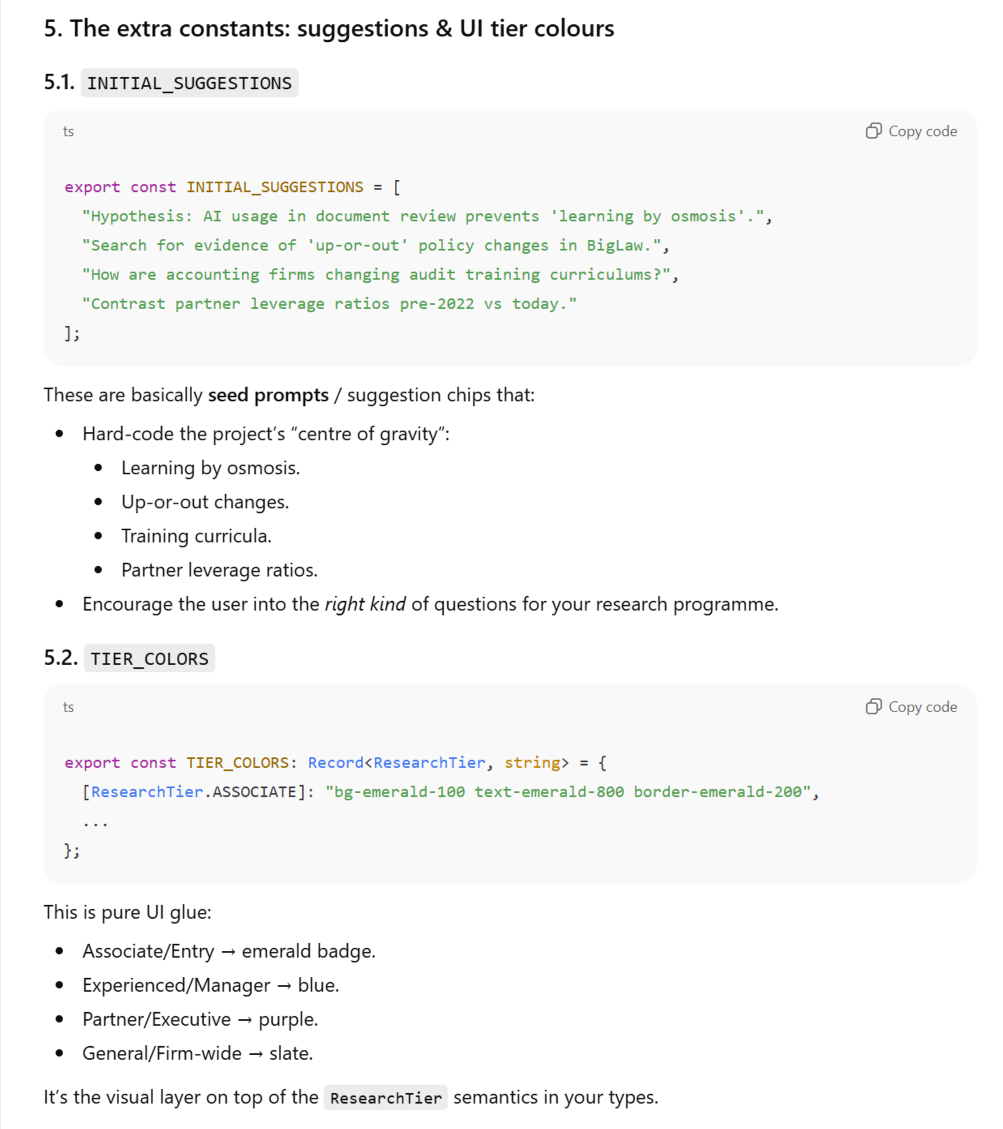
So the conceptual protocol is:

1. In the **body**:
   - Use `[fn.X]` markers after factual claims.
   - No naked URLs.
2. At the **very end**:
   - A `[SOURCE_START] ... [SOURCE_END]` block listing each `fn` with URL + Title.
   - That block is machine-parsed into `GroundingSource[]` and then stripped from the visible answer.

Together, this is a very deliberate design to:

- Force **auditable sourcing**.
- Keep the answer readable.
- Give your front-end enough structure to render clickable sources and maintain the link between claims and references.

ChatGPT 5.1 Thinking analysis of getSystemInstruction in constants.ts



5. The extra constants: suggestions & UI tier colours

5.1. `INITIAL_SUGGESTIONS`

```ts
export const INITIAL_SUGGESTIONS = [
  "Hypothesis: AI usage in document review prevents 'learning by osmosis'.",
  "Search for evidence of 'up-or-out' policy changes in BigLaw.",
  "How are accounting firms changing audit training curriculums?",
  "Contrast partner leverage ratios pre-2022 vs today."
];
```

These are basically **seed prompts** / suggestion chips that:

- Hard-code the project’s “centre of gravity”:
  - Learning by osmosis.
  - Up-or-out changes.
  - Training curricula.
  - Partner leverage ratios.
- Encourage the user into the *right kind* of questions for your research programme.

5.2. `TIER_COLORS`

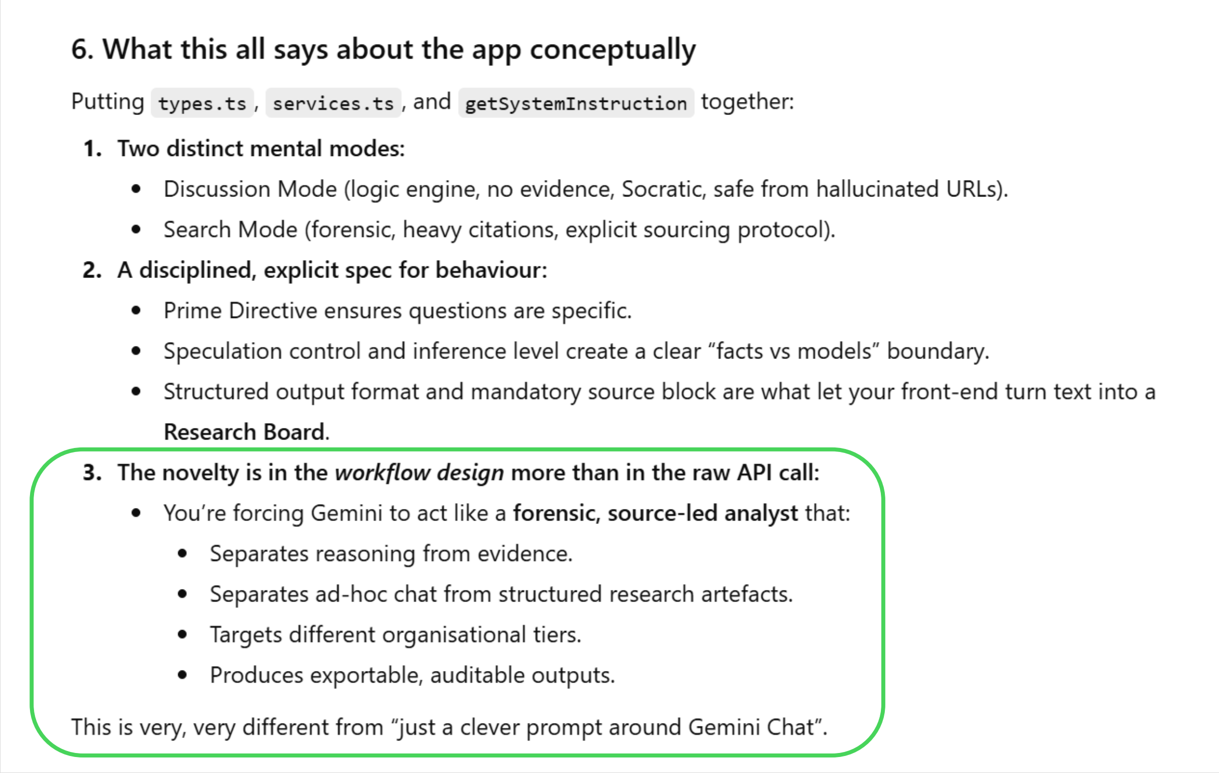
```ts
export const TIER_COLORS: Record<ResearchTier, string> = {
  [ResearchTier.ASSOCIATE]: "bg-emerald-100 text-emerald-800 border-emerald-200",
  ...
};
```

This is pure UI glue:

- Associate/Entry → emerald badge.
- Experienced/Manager → blue.
- Partner/Executive → purple.
- General/Firm-wide → slate.

It’s the visual layer on top of the `ResearchTier` semantics in your types.

ChatGPT 5.1 Thinking analysis of getSystemInstruction in constants.ts



6. What this all says about the app conceptually

Putting `types.ts`, `services.ts`, and `getSystemInstruction` together:

1. **Two distinct mental modes:**
   - Discussion Mode (logic engine, no evidence, Socratic, safe from hallucinated URLs).
   - Search Mode (forensic, heavy citations, explicit sourcing protocol).
2. **A disciplined, explicit spec for behaviour:**
   - Prime Directive ensures questions are specific.
   - Speculation control and inference level create a clear “facts vs models” boundary.
   - Structured output format and mandatory source block are what let your front-end turn text into a **Research Board**.
3. **The novelty is in the *workflow design* more than in the raw API call:**
   - You’re forcing Gemini to act like a **forensic, source-led analyst** that:
     - Separates reasoning from evidence.
     - Separates ad-hoc chat from structured research artefacts.
     - Targets different organisational tiers.
     - Produces exportable, auditable outputs.

This is very, very different from “just a clever prompt around Gemini Chat”.

Appendix