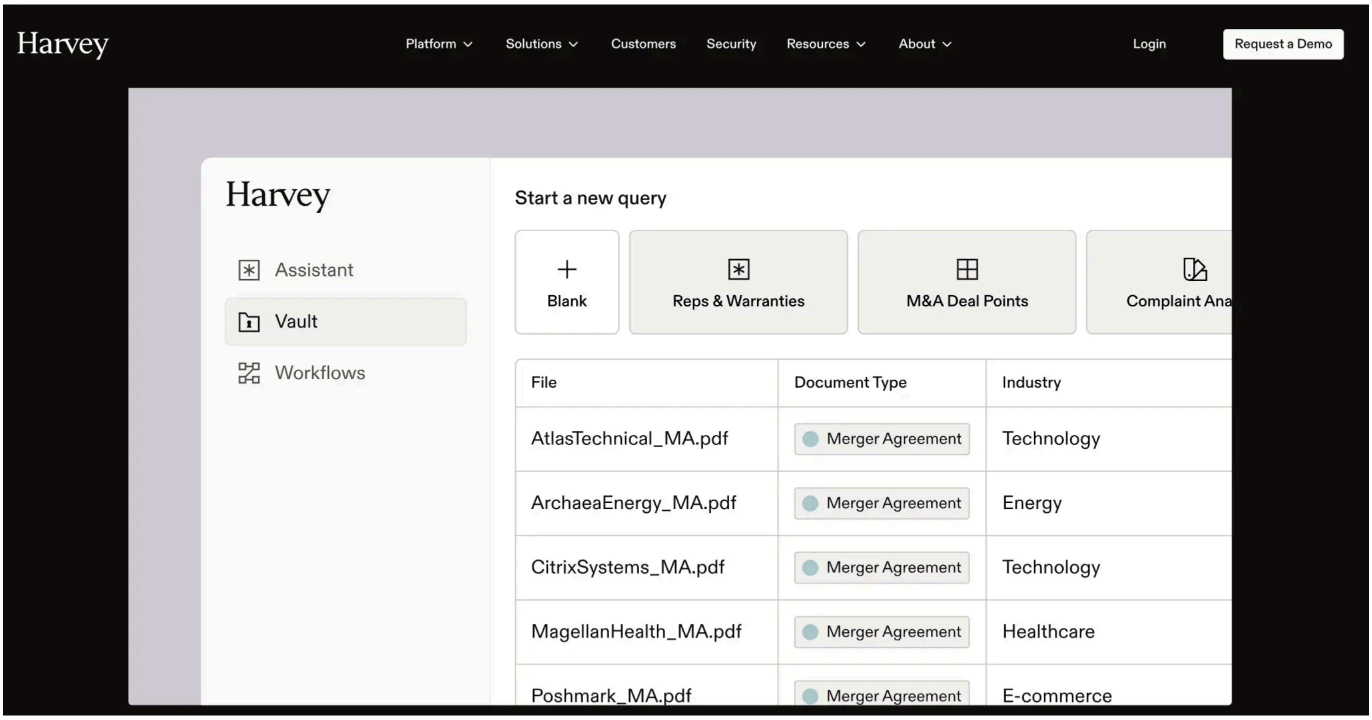
How to "Vibe Code" Harvey's and Legora's Tabular Review Tool

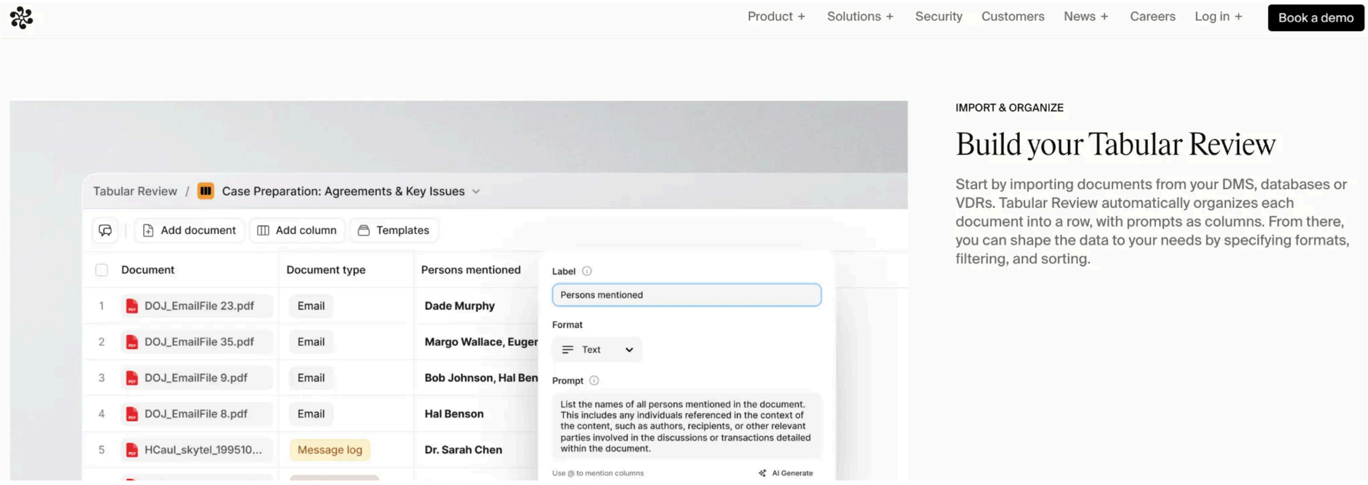
Step 1: Find Public Links to the Product

Use Google or Perplexity to gather all publicly available information about Harvey’s and Legora’s tabular review features.

- **Harvey's Vault function:** https://www.harvey.ai/platform/vault
- **Legora's Product Page:** https://legora.com/product/tabular-review
- **LinkedIn Announcement:** https://www.linkedin.com/feed/update/urn:li:activity:7325187800062935040/
- **YouTube Demo:** https://www.youtube.com/watch?v=_sI_QMd5oG4



File	Document Type	Industry
AtlasTechnical_MA.pdf	Merger Agreement	Technology
ArchaeaEnergy_MA.pdf	Merger Agreement	Energy
CitrixSystems_MA.pdf	Merger Agreement	Technology
MagellanHealth_MA.pdf	Merger Agreement	Healthcare
Poshmark_MA.pdf	Merger Agreement	E-commerce



	Document	Document type	Persons mentioned
1	DOJ_EmailFile 23.pdf	Email	Dade Murphy
2	DOJ_EmailFile 35.pdf	Email	Margo Wallace, Euger
3	DOJ_EmailFile 9.pdf	Email	Bob Johnson, Hal Ben
4	DOJ_EmailFile 8.pdf	Email	Hal Benson
5	HCaul_skytel_199510...	Message log	Dr. Sarah Chen

Label: Persons mentioned

Format: Text

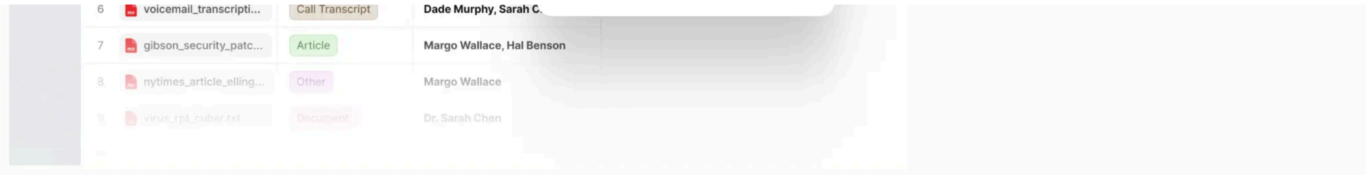
Prompt: List the names of all persons mentioned in the document. This includes any individuals referenced in the context of the content, such as authors, recipients, or other relevant parties involved in the discussions or transactions detailed within the document.

Use @ to mention columns

IMPORT & ORGANIZE

Build your Tabular Review

Start by importing documents from your DMS, databases or VDRs. Tabular Review automatically organizes each document into a row, with prompts as columns. From there, you can shape the data to your needs by specifying formats, filtering, and sorting.



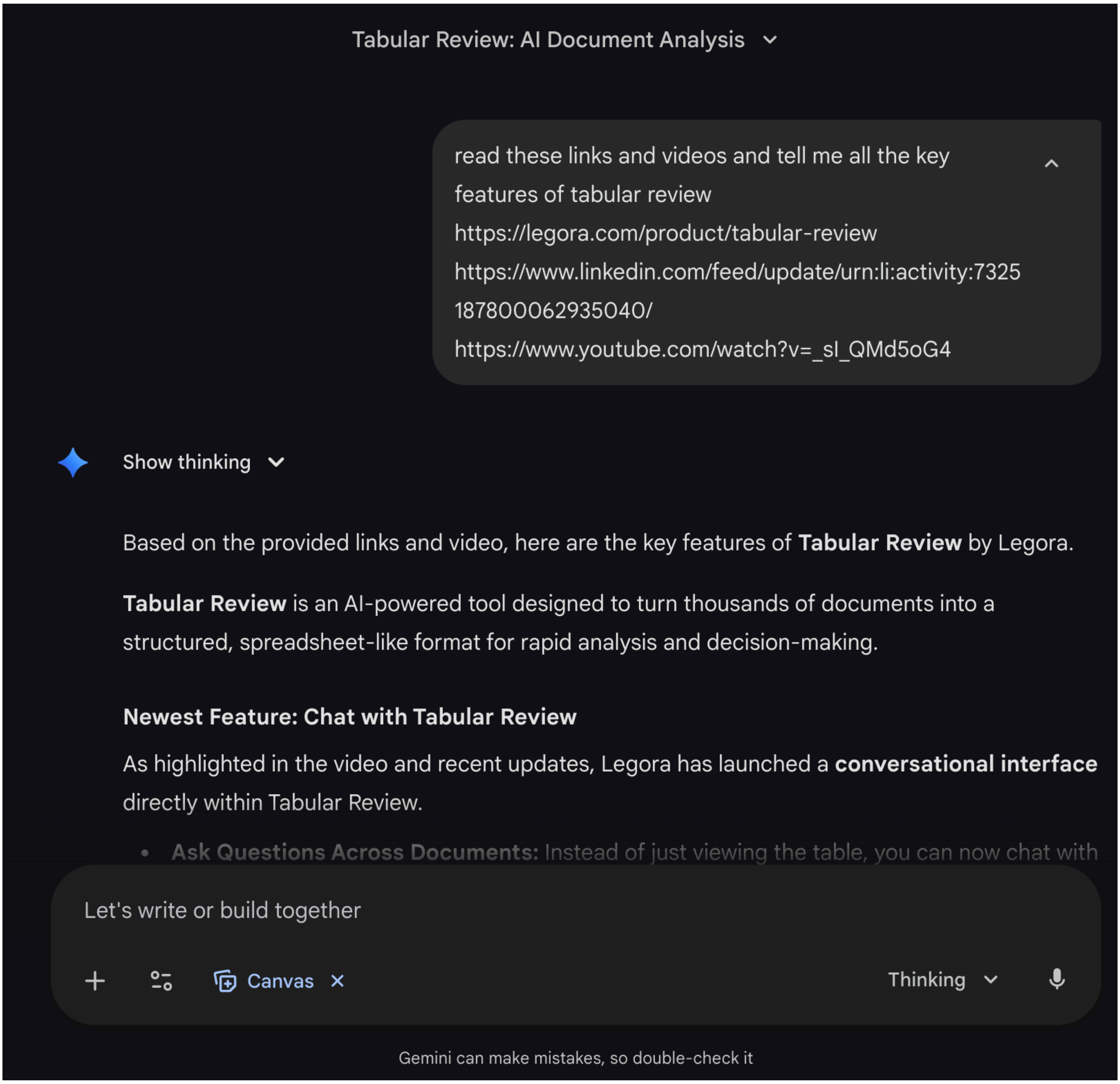
6	voicemail_transcripti...	Call Transcript	Dade Murphy, Sarah C.
7	gibson_security_patc...	Article	Margo Wallace, Hal Benson
8	nytimes_article_elling...	Other	Margo Wallace
9	virus_rpt_cuber.txt	Document	Dr. Sarah Chen

Step 2: Ask Gemini to Analyse Product Features

Prompt for Gemini 3 (https://gemini.google.com/app):

Plain Text

```
read these links and videos and tell me all the key features of tabular
review. (paste product page and demo links)
```



read these links and videos and tell me all the key features of tabular review https://legora.com/product/tabular-review https://www.linkedin.com/feed/update/urn:li:activity:7325187800062935040/ https://www.youtube.com/watch?v=_sl_QMd5oG4

Based on the provided links and video, here are the key features of **Tabular Review** by Legora.

Tabular Review is an AI-powered tool designed to turn thousands of documents into a structured, spreadsheet-like format for rapid analysis and decision-making.

Newest Feature: Chat with Tabular Review

As highlighted in the video and recent updates, Legora has launched a **conversational interface** directly within Tabular Review.

- **Ask Questions Across Documents:** Instead of just viewing the table, you can now chat with

Let's write or build together
Canvas
Thinking
Gemini can make mistakes, so double-check it

Step 3: Ask Gemini to generate product specs

Prompt for Gemini 3:

```
Now i want to build this on Google AI studio, Give me the product specs for
this.
```

Expected Output:

`Product Specifications: AI-Powered Tabular Review Tool`

Version: 1.0

Platform: Web (React/Next.js) + Google Gemini API

Target Audience: Legal professionals, Auditors, Procurement teams

1. Product Overview

An AI-powered document review workspace that ingests unstructured documents (PDFs, Word) and converts them into a structured, queryable database. Users define "columns" via natural language questions, and the AI fills the "rows" (documents) with extracted answers. The system includes a chat interface to reason across the structured data.

1. Core Feature Requirements

2.1 Document Ingestion

Spec: User must be able to upload multiple files (PDF, TXT, DOCX) simultaneously.

Constraint: System must handle file parsing (PDF to text/image) before sending to Gemini, or use Gemini's native file API if using the File API.

Limit: MVP capability of 50-100 documents per batch.

2.2 Dynamic Schema (The "Magic Columns")

Spec: Users create columns by typing a natural language prompt.

Example: Column Name: "Termination Notice", Prompt: "How many days notice is required to terminate?"

Data Types: The AI must be able to extract:

Text (Summaries, clauses)

Dates (ISO format)

Currency/Numbers

Boolean (Yes/No)

Select/Enum (e.g., "Governing Law": [NY, CA, UK])

2.3 Bulk Extraction Engine

Spec: Upon adding a column, the system triggers a parallel extraction job for all documents.

Verification Layer: For every extracted cell, the AI must return:

value: The answer (e.g., "30 Days").

reasoning: A short explanation.

source_text: The exact quote from the document.

page_number: Location in the doc.

2.4 The "Grid" Interface

Spec: A high-density table view (similar to Excel/Airtable).

Interactions:

Click-to-Verify: Clicking a cell opens a sidebar showing the original document scrolled to the specific clause (using the source_text or page_number).

Manual Override: User can edit AI-generated cells.

2.5 Chat-to-Data (The "Analyst")

Spec: A chat pane that has read-access to the generated table.

Capabilities:

Aggregation: "How many contracts expire in 2025?"

Comparison: "Which contract has the strictest liability clause?"

Outlier Detection: "Show me contracts that deviate from standard payment terms."

1. Technical Architecture (Google AI Studio Stack)

3.1 AI Models

Extraction Model: gemini-1.5-pro

Why: Large context window allows passing full documents without complex RAG chunking. High reasoning capability for complex legal queries.

Chat Model: gemini-1.5-flash

Why: Lower latency and cost for rapid back-and-forth chat once the data is extracted.

3.2 Prompt Engineering Strategy

The Extraction Prompt (JSON Mode)

You must use response_mime_type: "application/json" to ensure reliability.

System Instruction:

You are a legal data extraction engine. You will be given a document and a specific extraction query. You must respond in the following JSON format ONLY: { "value": "The extracted answer", "confidence": "High/Medium/Low", "quote": "Verbatim text from document supporting the answer", "page": Integer }

User Prompt:

Document: [Insert Document Text or File URI] Query: "What is the governing law of this contract?"

3.3 Data Structure (Frontend State)

documents Array:

[{ "id": "doc_1", "name": "Vendor_Agmt_A.pdf", "content": "..." }, { "id": "doc_2", "name": "Vendor_Agmt_B.pdf", "content": "..." }]

`schema Array (Columns):`

`[ { "id": "col_1", "name": "Contract Value", "prompt": "What is the total total monetary value?" }, { "id": "col_2", "name": "Renewal", "prompt": "Is there auto-renewal?" } ]`

`extractionResults (The Grid Data):`

`{ "doc_1": { "col_1": { "value": "$50,000", "quote": "Total fees shall not exceed...", "page": 2 }, "col_2": { "value": "Yes", "quote": "This agreement auto-renews for 1 year terms", "page": 5 } } }`

1. `User Flow (UX)`

`Upload: User drags & drops 10 PDF contracts.`

`Preprocessing: App converts PDFs to text (or uses multimodal input).`

`Define Columns:`

`User clicks "+" on the table header.`

`User types: "Governing Law".`

`User types prompt: "Extract the jurisdiction state/country".`

`Processing: Progress bars appear on the column.`

`Review: Table populates. User notices one row says "Unknown".`

`Verification: User clicks "Unknown". Sidebar opens. User manually finds the clause and updates the cell.`

`Chat: User opens Chat. "Summarize the risk in the Governing Law column."`

`Response: AI reads the column values and generates a summary.`

1. `Roadmap for MVP`

`Phase 1 (Skeleton): React Table + File Upload + Hardcoded extraction prompt.`

`Phase 2 (Dynamic): Allow users to input custom prompts for columns.`

`Phase 3 (Verification): Implement the UI to show "Reasoning/Quote" on cell click.`

`Phase 4 (Chat): Implement RAG over the JSON table data.`

Step 4: Build the App in AI Studio

Go to https://aistudio.google.com/welcome, paste your product specifications from Step 3 into the chat box, and use the following prompt:

Plain Text

```
Can you build this for me? (paste product specifications above)
```

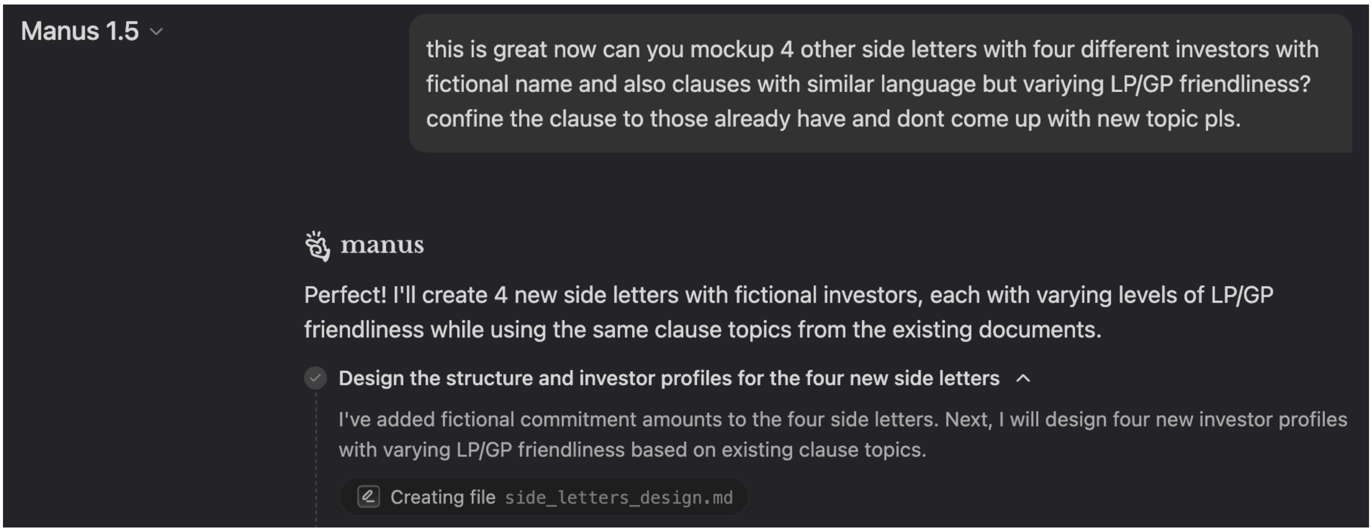
Step 5: Generate Fictional Test Documents

Use a general AI agent like Manus (https://manus.im/app) to create realistic but fictional legal documents in docx format.

Prompt Example:

Plain Text

```
Create three fictional documents in .docx format: a fund side letter, an
employment agreement, and a share purchase agreement.
```

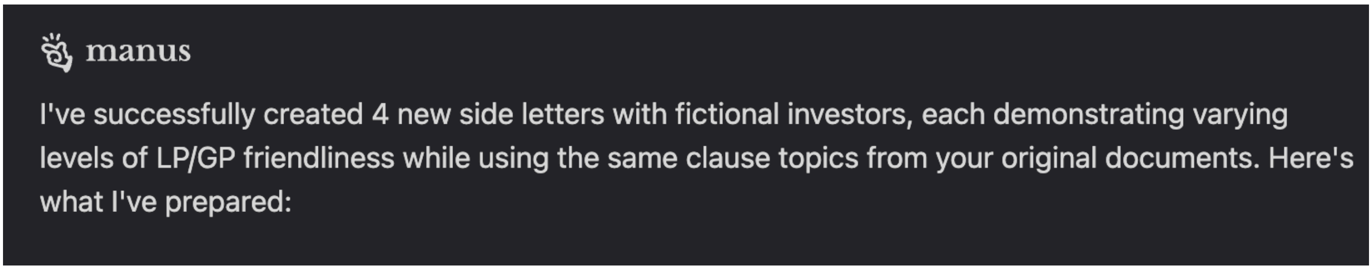


this is great now can you mockup 4 other side letters with four different investors with fictional name and also clauses with similar language but variying LP/GP friendliness? confine the clause to those already have and dont come up with new topic pls.

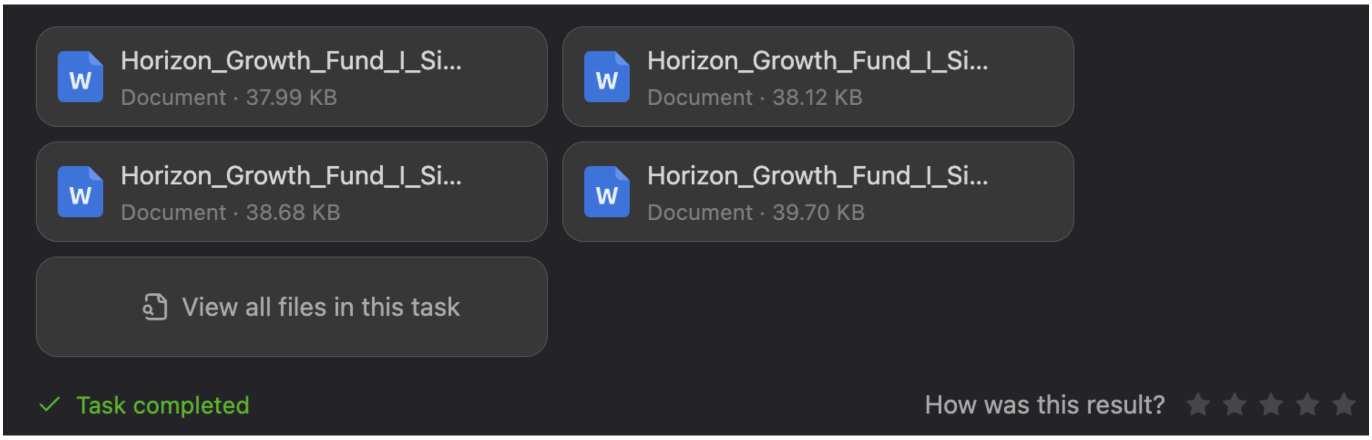
Perfect! I'll create 4 new side letters with fictional investors, each with varying levels of LP/GP friendliness while using the same clause topics from the existing documents.

Design the structure and investor profiles for the four new side letters

I've added fictional commitment amounts to the four side letters. Next, I will design four new investor profiles with varying LP/GP friendliness based on existing clause topics.



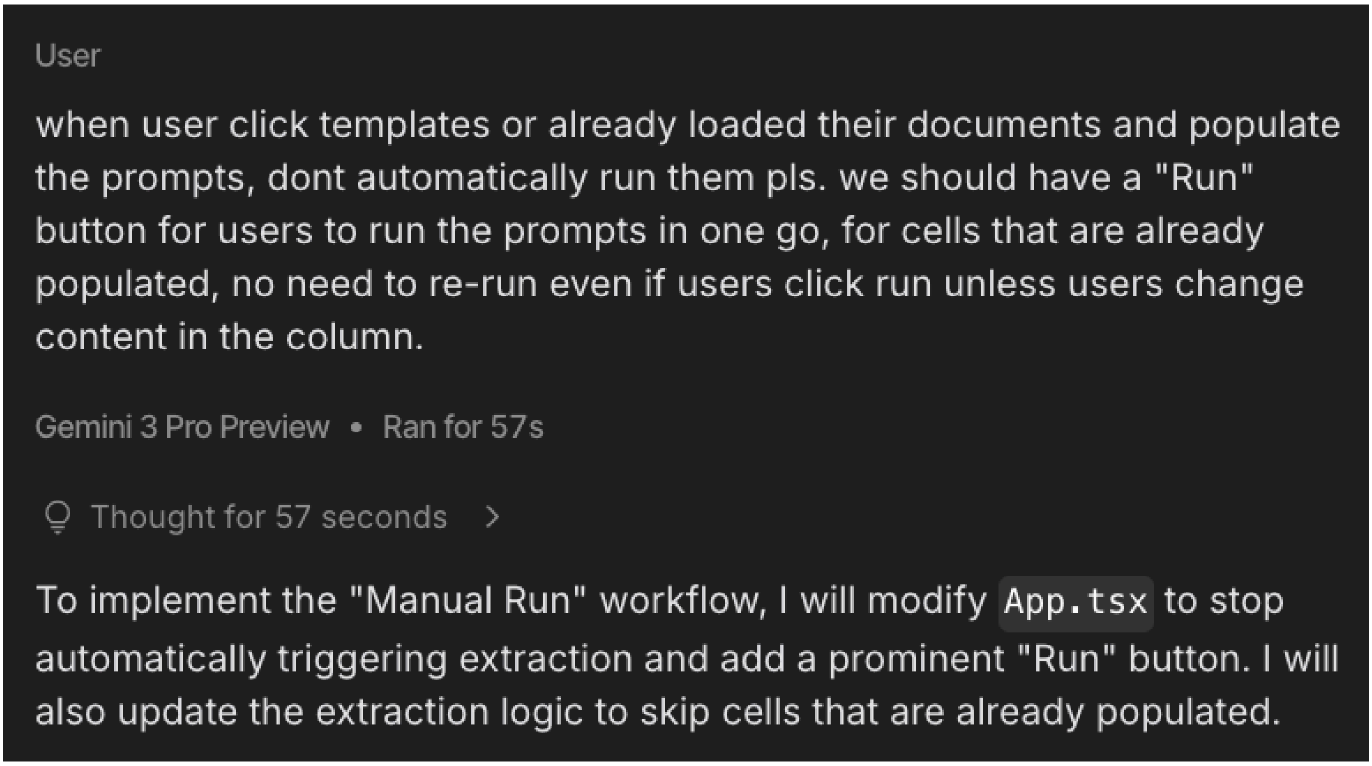
I've successfully created 4 new side letters with fictional investors, each demonstrating varying levels of LP/GP friendliness while using the same clause topics from your original documents. Here's what I've prepared:


Horizon_Growth_Fund_I_Si...
Document · 37.99 KB
Horizon_Growth_Fund_I_Si...
Document · 38.12 KB
Horizon_Growth_Fund_I_Si...
Document · 38.68 KB
Horizon_Growth_Fund_I_Si...
Document · 39.70 KB
View all files in this task
Task completed
How was this result?

Step 6: Test and Debug the Application

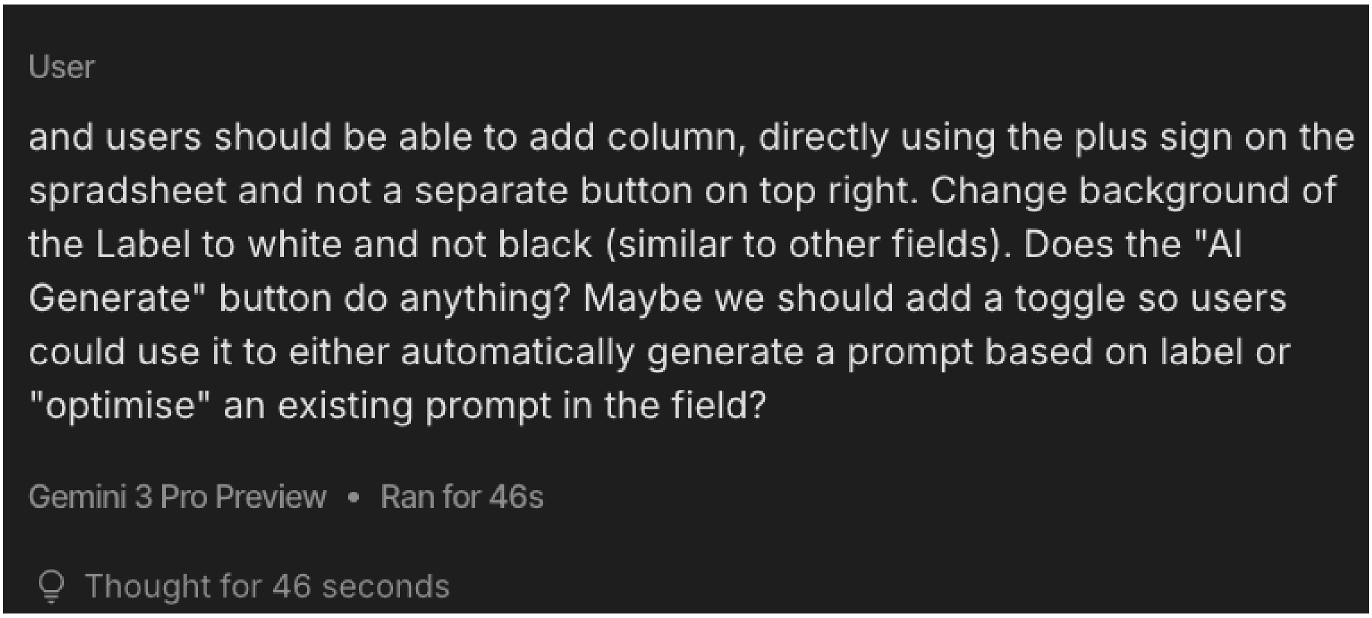
Go back to the app built in AI Studio and test the app with the fictional `.docx` and `.pdf` files you created. Iterate with more prompts to identify and fix any errors until you achieve feature parity.

Testing/Debugging Prompt Examples:



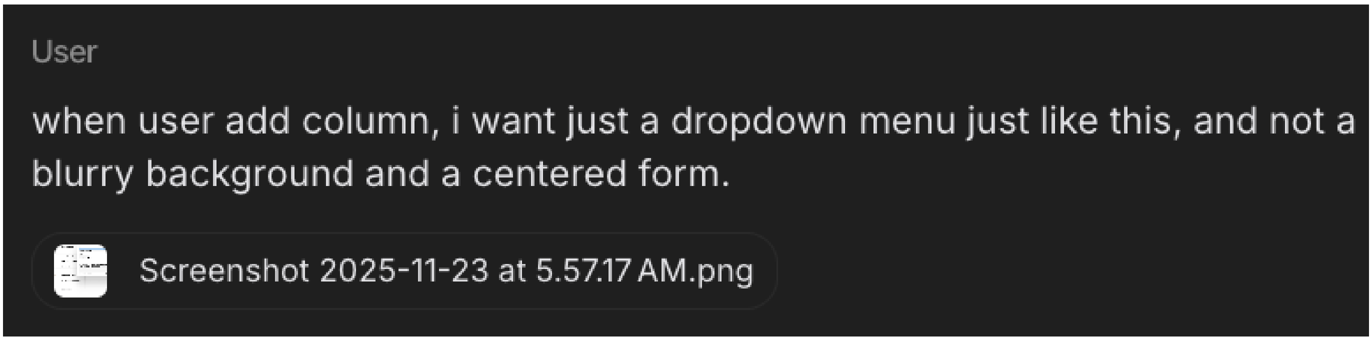
when user click templates or already loaded their documents and populate the prompts, dont automatically run them pls. we should have a "Run" button for users to run the prompts in one go, for cells that are already populated, no need to re-run even if users click run unless users change content in the column.

To implement the "Manual Run" workflow, I will modify `App.tsx` to stop automatically triggering extraction and add a prominent "Run" button. I will also update the extraction logic to skip cells that are already populated.



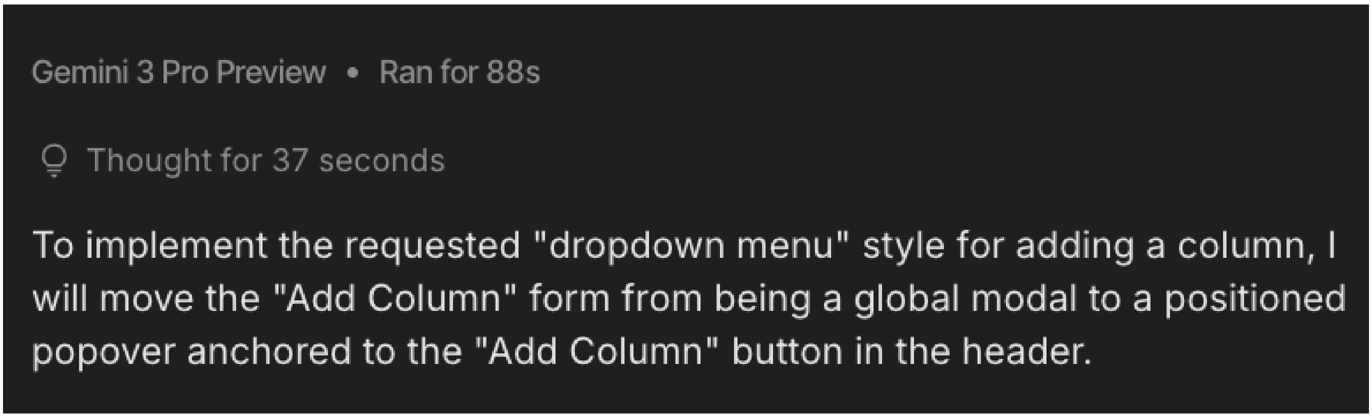
and users should be able to add column, directly using the plus sign on the spradsheet and not a separate button on top right. Change background of the Label to white and not black (similar to other fields). Does the "AI Generate" button do anything? Maybe we should add a toggle so users could use it to either automatically generate a prompt based on label or "optimise" an existing prompt in the field?

Gemini 3 Pro Preview • Ran for 46s
Thought for 46 seconds



when user add column, i want just a dropdown menu just like this, and not a blurry background and a centered form.

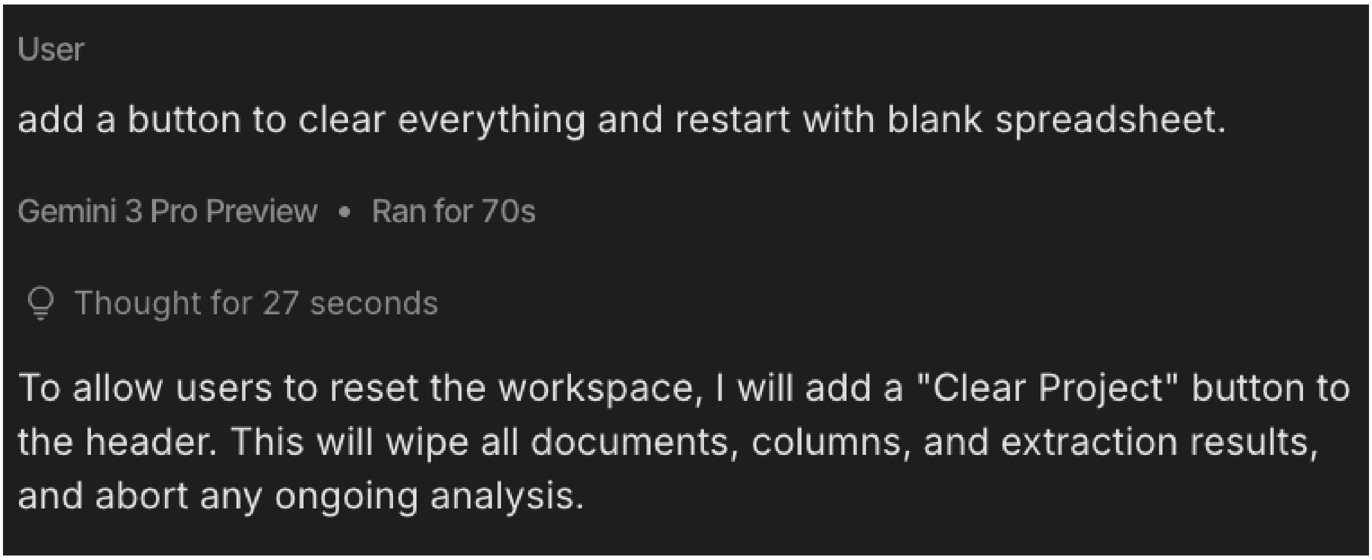
Screenshot 2025-11-23 at 5.57.17 AM.png



Gemini 3 Pro Preview • Ran for 88s

Thought for 37 seconds

To implement the requested "dropdown menu" style for adding a column, I will move the "Add Column" form from being a global modal to a positioned popover anchored to the "Add Column" button in the header.



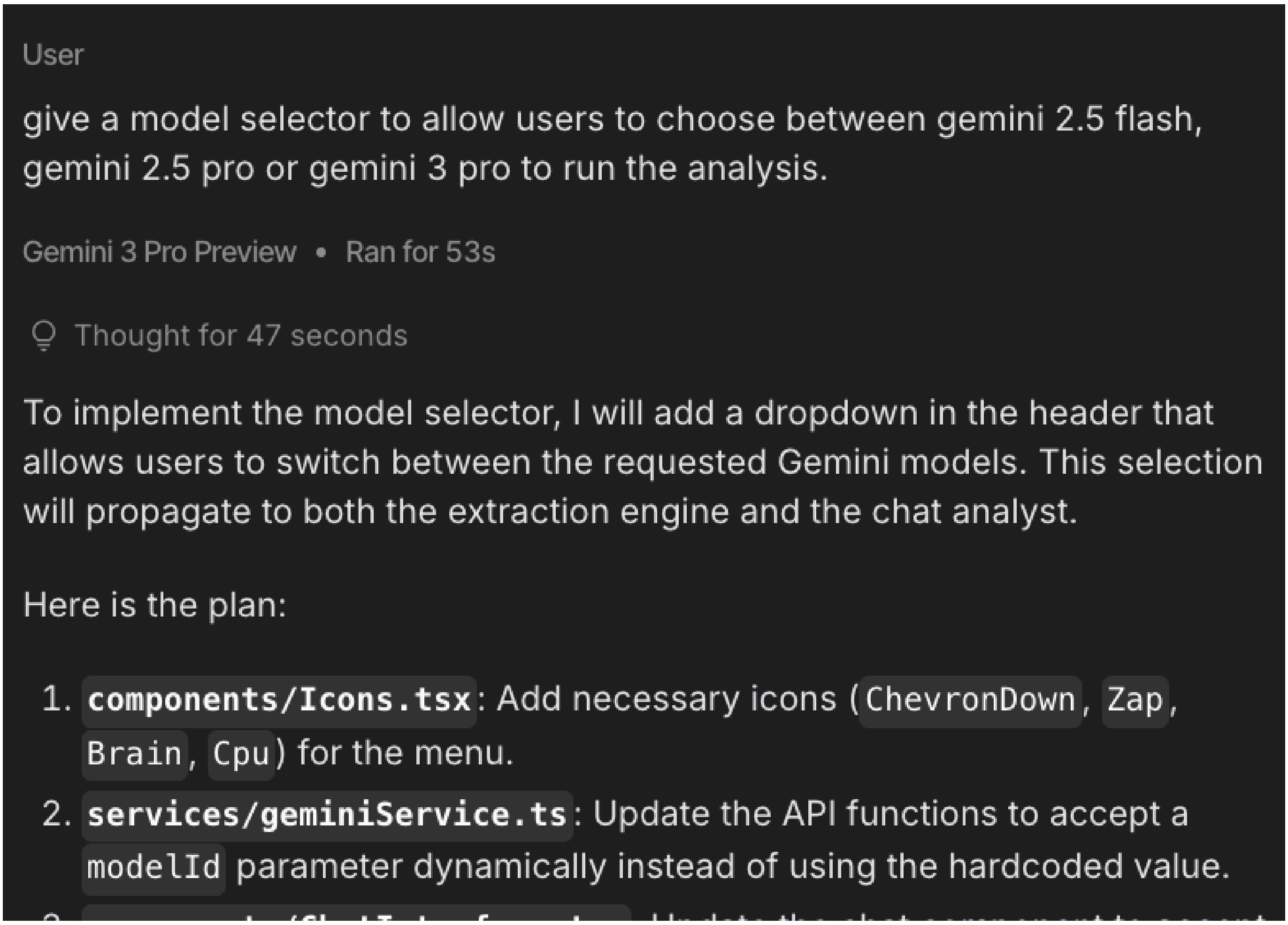
User

add a button to clear everything and restart with blank spreadsheet.

Gemini 3 Pro Preview • Ran for 70s

Thought for 27 seconds

To allow users to reset the workspace, I will add a "Clear Project" button to the header. This will wipe all documents, columns, and extraction results, and abort any ongoing analysis.



User

give a model selector to allow users to choose between gemini 2.5 flash, gemini 2.5 pro or gemini 3 pro to run the analysis.

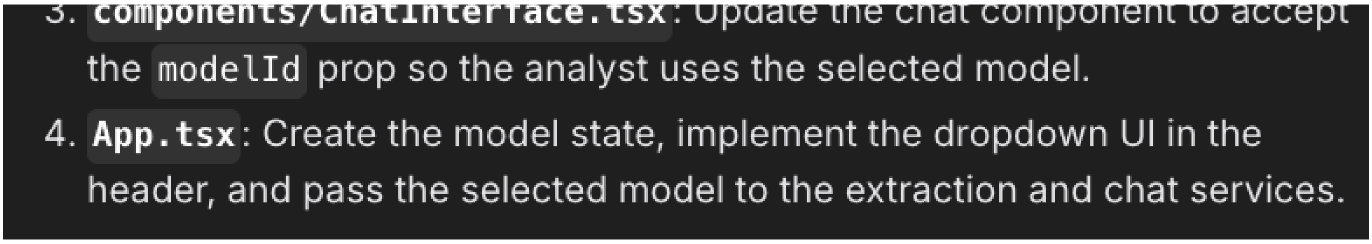
Gemini 3 Pro Preview • Ran for 53s

Thought for 47 seconds

To implement the model selector, I will add a dropdown in the header that allows users to switch between the requested Gemini models. This selection will propagate to both the extraction engine and the chat analyst.

Here is the plan:

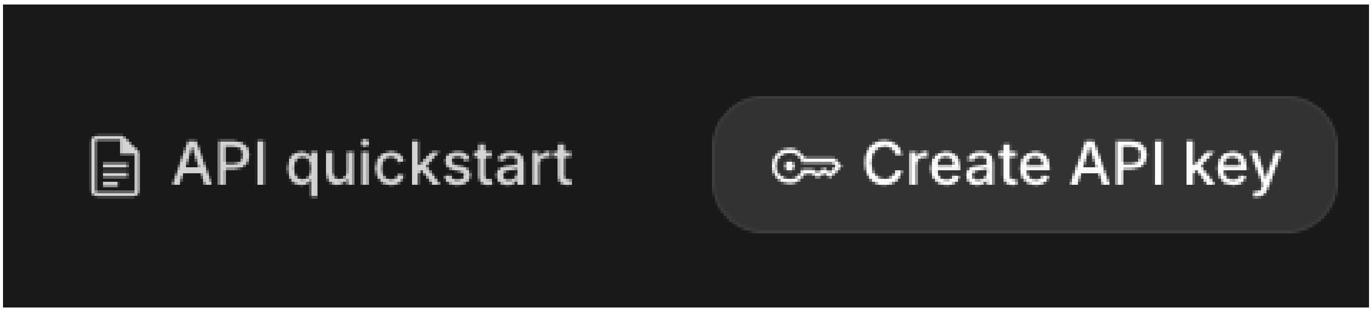
1. **`components/Icons.tsx`**: Add necessary icons (`ChevronDown`, `Zap`, `Brain`, `Cpu`) for the menu.
2. **`services/geminiService.ts`**: Update the API functions to accept a `modelId` parameter dynamically instead of using the hardcoded value.

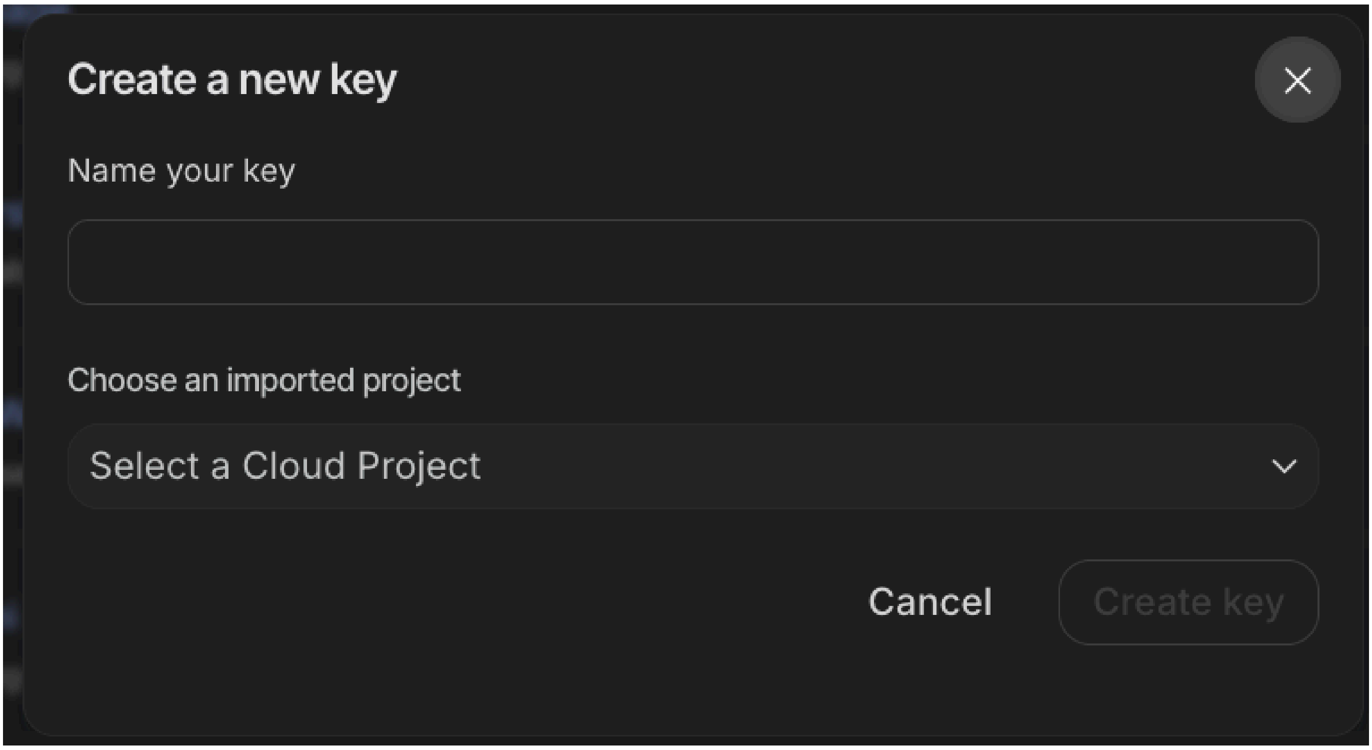


3. `components/ChatInterface.tsx`: Update the chat component to accept the `modelId` prop so the analyst uses the selected model.
4. `App.tsx`: Create the model state, implement the dropdown UI in the header, and pass the selected model to the extraction and chat services.

Step 7: (Optional) Get Your API Key for Local Deployment

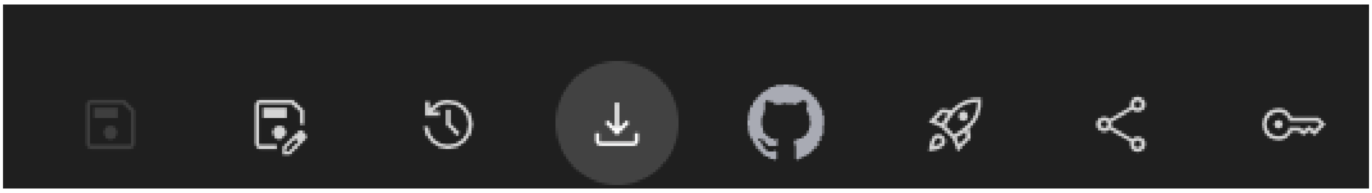
For advanced "vibe coders" who want to run the app on their own machine, you'll need an API key. Go to https://aistudio.google.com/api-keys and generate a new API key.

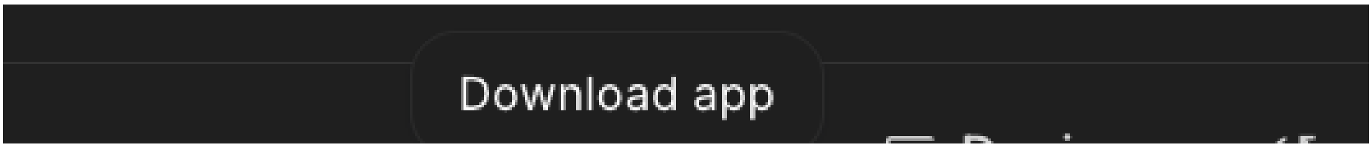




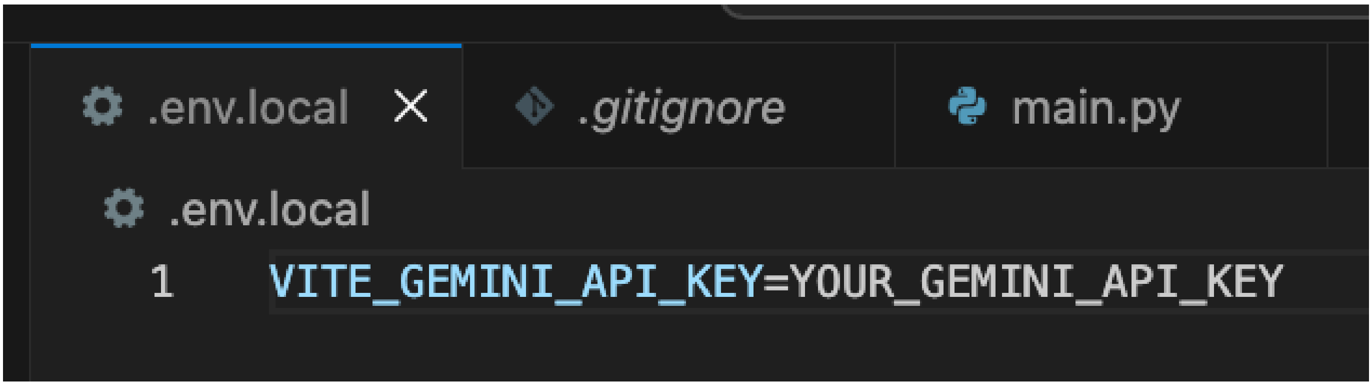
Step 8: Run the App Locally

1. Download the application files from AI Studio.



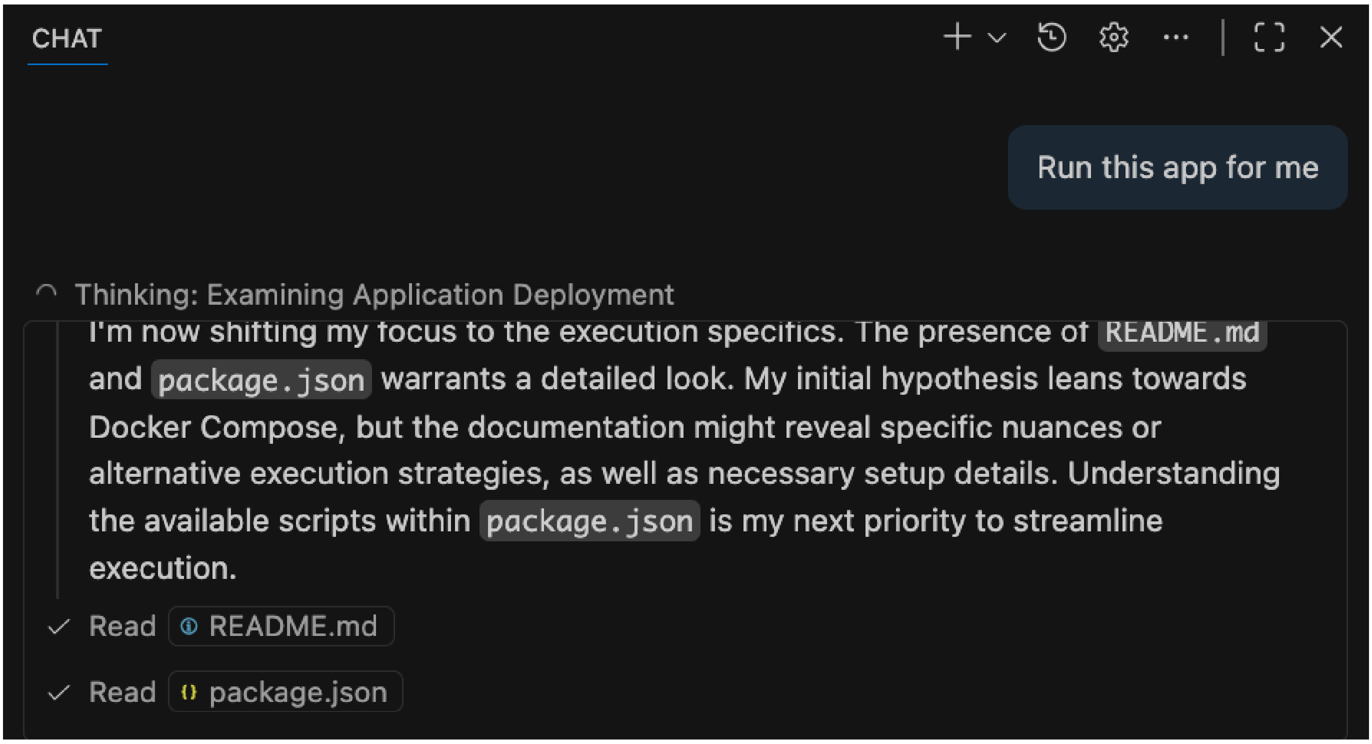


1. Install a code editor like VS Code (https://code.visualstudio.com/)
2. Find the `.env` file in the project folder and replace the placeholder API key with your own Google API key from Step 7.



```
VITE_GEMINI_API_KEY=YOUR_GEMINI_API_KEY
```

1. Open the project in VS Code and use an AI coding assistant like GitHub Copilot with the prompt: "Run this app for me".



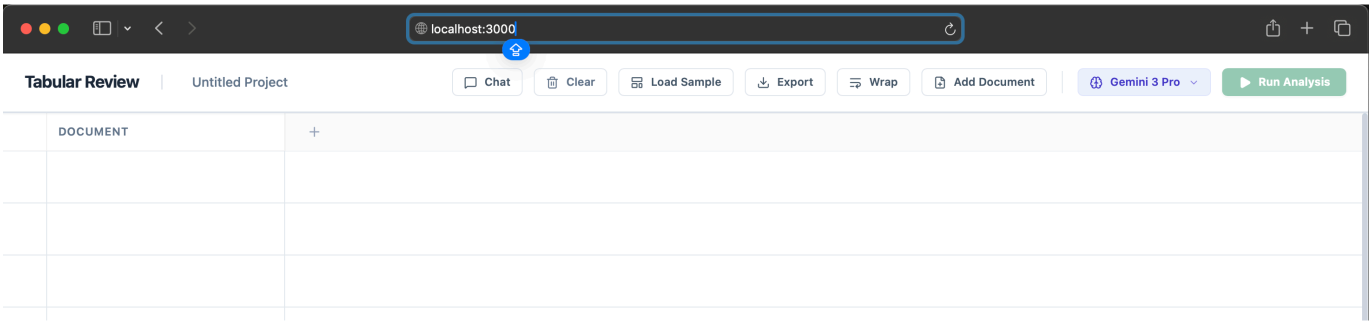
Run this app for me

Thinking: Examining Application Deployment

I'm now shifting my focus to the execution specifics. The presence of `README.md` and `package.json` warrants a detailed look. My initial hypothesis leans towards Docker Compose, but the documentation might reveal specific nuances or alternative execution strategies, as well as necessary setup details. Understanding the available scripts within `package.json` is my next priority to streamline execution.

✓ Read README.md

✓ Read package.json


localhost:3000
Tabular Review
Untitled Project
Chat
Clear
Load Sample
Export
Wrap
Add Document
Gemini 3 Pro
Run Analysis
DOCUMENT
+