Not-Vibe Coding a GUI

Effectively Building a Web UI Using AI

Made with GAMMA

Core Thesis

You asked the AI to "build a simple dashboard," and twenty minutes later you were staring at a UI that *almost* worked — wrong flows, mismatched components, and fixes that kept breaking other screens.

You're not vibe coding.

You're leading a team. A team of fast, junior, very capable engineers.

LLMs don't replace product thinking or UX decisions — they amplify them *only if* you make them explicit.

Made with GAMMA

The Promise of Vibe Coding

The dream

- Describe the app in natural language
- Let the AI **figure out the details**
- Ship a UI in an afternoon

Made with GAMMA

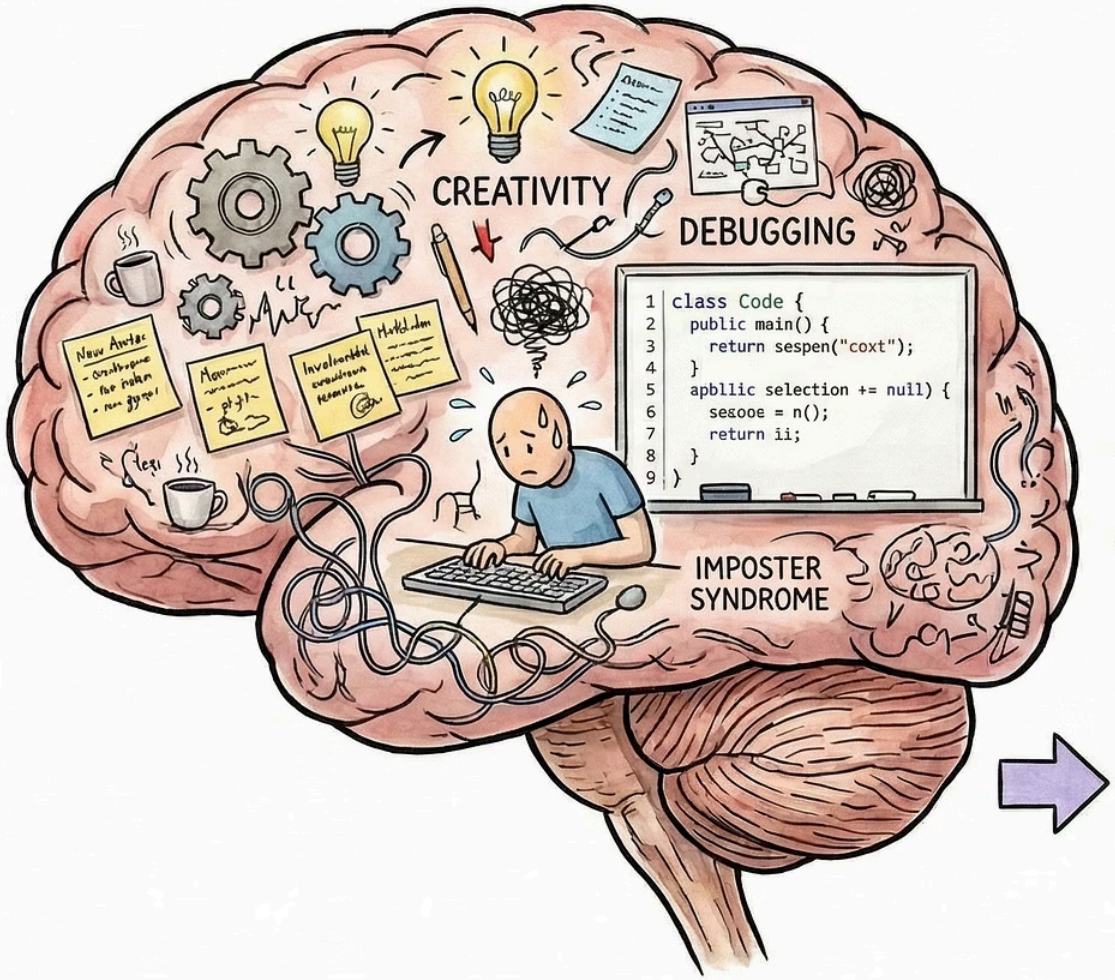
The Reality of Vibe Coding

The reality

- Inconsistent UI
- Broken or missing flows
- Conflicting components
- Endless re-prompting, trying to fix the wrongs.

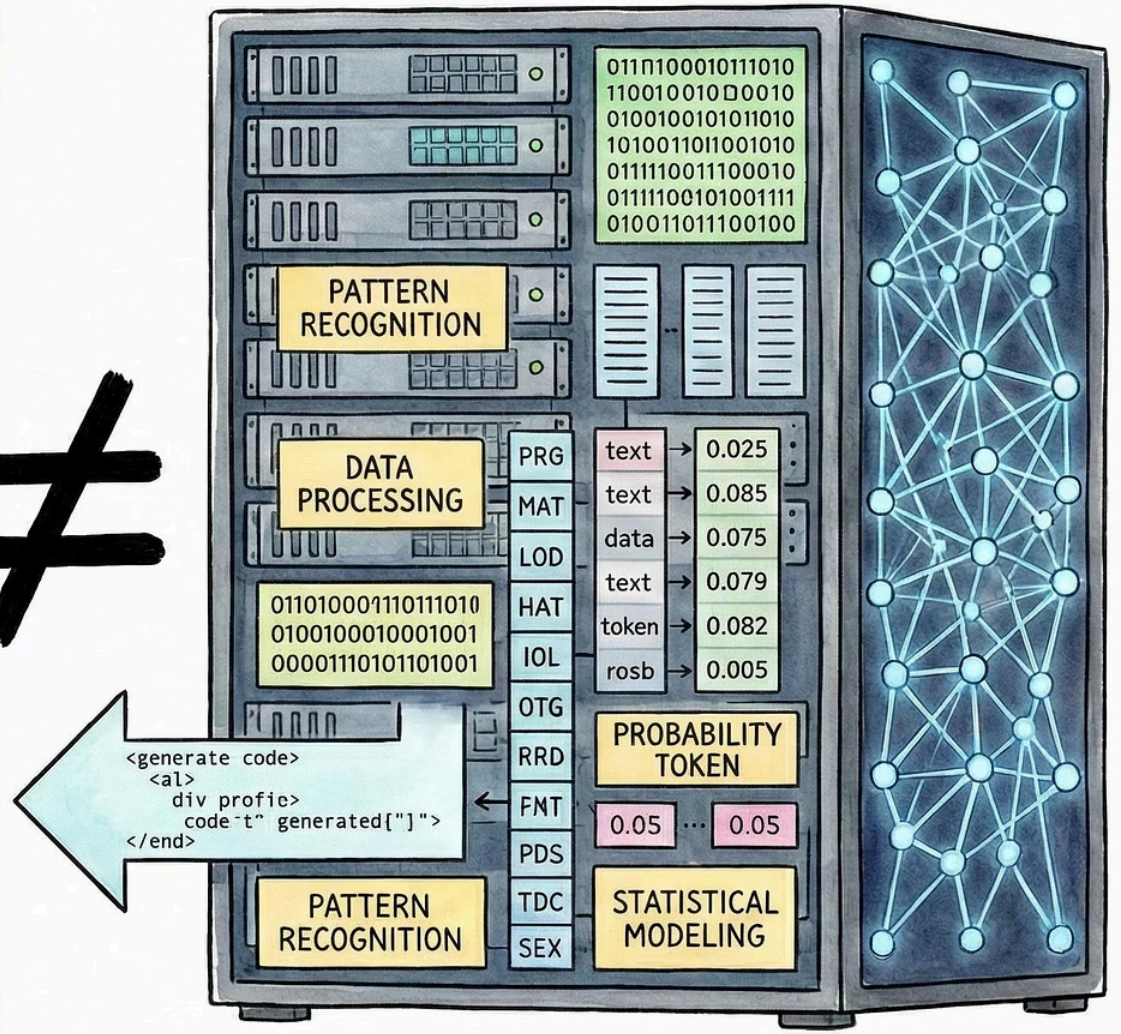
Made with GAMMA

DEVELOPER'S MIND



≠

LLM'S MIND



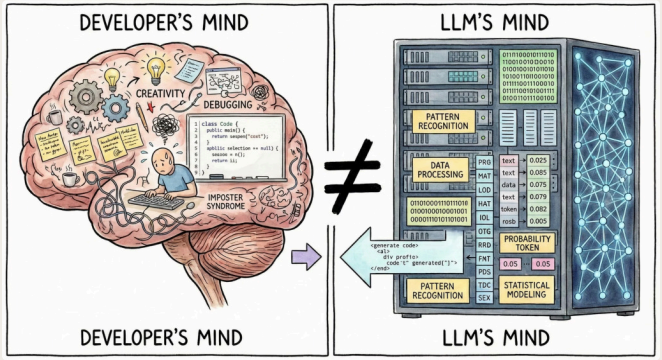
DEVELOPER'S MIND

LLM'S MIND

Made with GAMMA

The Problem: Wrong mental model

This gap between expectation and reality is the root cause of most AI-generated UI failures.



What we assume

- The model understands the product
- The model remembers earlier decisions
- The model infers your intent
- The model thinks about edge cases

What actually happens

- Generation is **stateless**
- Lost in ~~translation~~ verbalization - you think you understand what you ask for, but the model **interprets** it differently.
- No shared memory (unless you create it). You know the product, the model doesn't
- You imagine how it's going to look, but the AI "imagines" something different. There are **unspoken gaps**.

Made with GAMMA

The Correct Mental Model

You Are Leading a Team

Extremely fast

Extremely literal

But - Junior-level product intuition

And - Zero memory beyond what you document

Made with GAMMA

Prompts are not wishes - they are instructions.

Made with GAMMA

What "Leading" Means Here

Define *what* to build

Define *constraints*

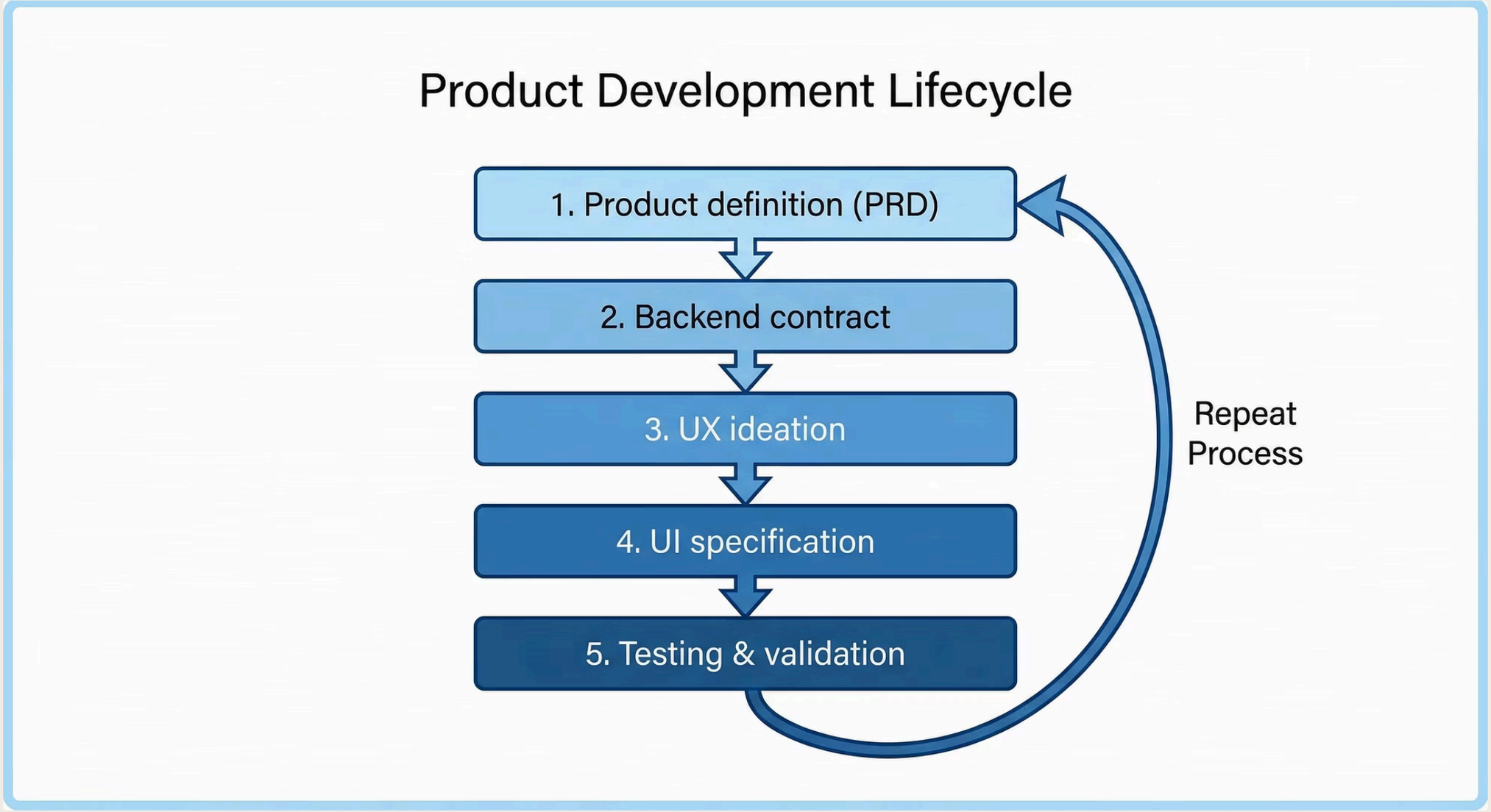
Define *interfaces*

Review output *critically*

This is engineering leadership, not magic.

The Process (Overview)

Each step deliberately reduces ambiguity - for you *and* for the model - turning vague intent into executable instructions.



Boring. Repeatable. Effective.

Made with GAMMA

4. User Stories / Personas

4.1. User Persona: RAG developer / PM

- **Role:** RAG developer / PM
- **Needs:** Need to find a tool for evaluating and improving their RAG system
- **Goals:** Assure the quality of their RAG system

4.1. User Persona: System administrator

- **Role:** AIIR admin
- **Needs:** Monitor health, troubleshoot, and assist users

4.2. User Stories

- As a user I want to:
  - Create a corpus
  - Generate a benchmark
  - Evaluate my RAG based on the benchmark
  - Experiment with DM using a single or 2 in-line docs
  - Use the demo corpus to experiment with DM
  - Check how much quota I have left
- As an admin I want to:
  - List all past generation requests
  - List all benchmark requests
  - Study usage patterns from benchmark definitions
  - List all ongoing requests
  - Monitor cost
  - Monitor failures (realtime) to provide support
  - List corpora
  - List errors and access logs
  - Purge old data: corpus, benchmark, evaluation results, users, logs

Step 1: Start with a Real PRD

1 Who is the user?

2 What are we solving?

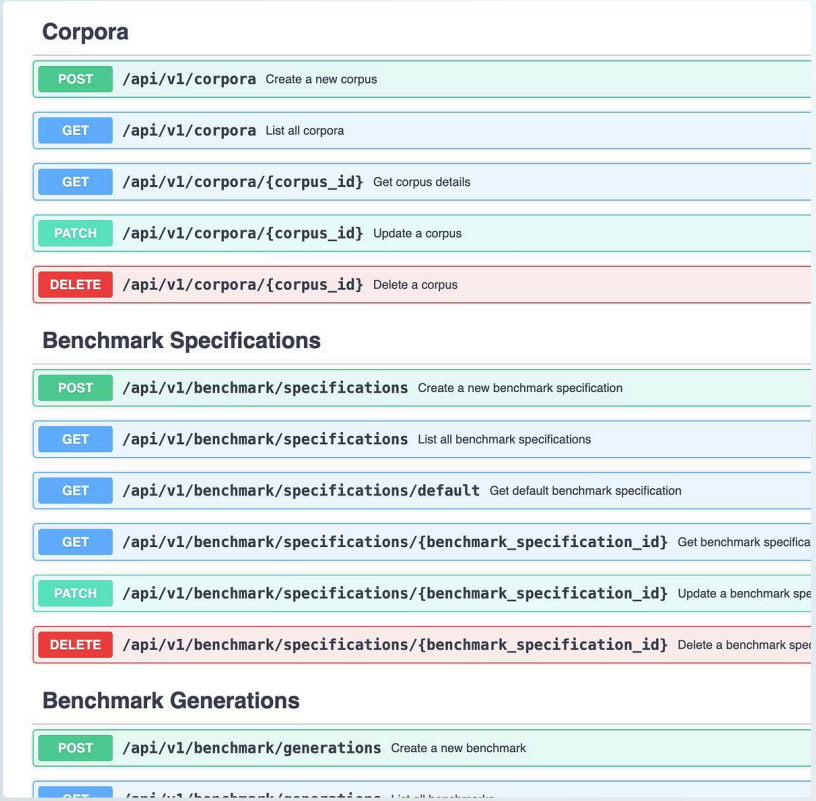
3 Use cases

4 Non-goals

5 Constraints

Iterate on the PRD with **Team members** and **AI**

Made with GAMMA

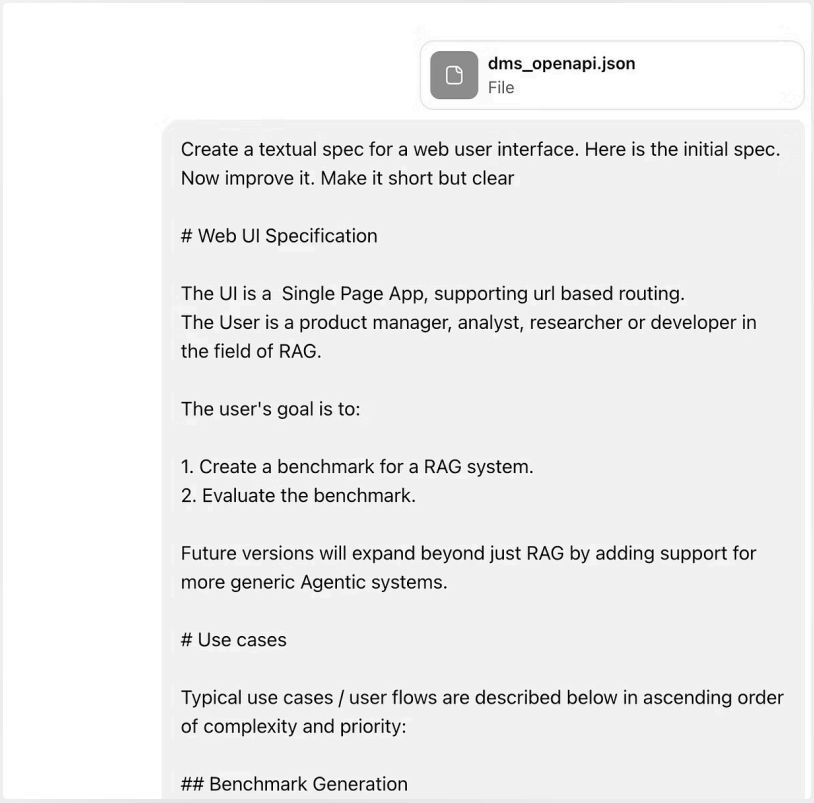


Step 2: Define the Backend Contract

- OpenAPI / Swagger
- Real endpoints
- Real schemas
- Error cases included

UIs cannot safely improvise APIs.

Made with GAMMA



Step 3: UX Ideation (with chat)

1. Explore flows conversationally
2. Ask "what happens next?"
3. Try alternatives
4. Capture decisions

Do not ship from this phase.

Made with GAMMA

Main goals:

1. Create benchmarks.
2. Evaluate benchmarks.

Future versions will extend to general agentic evaluations.

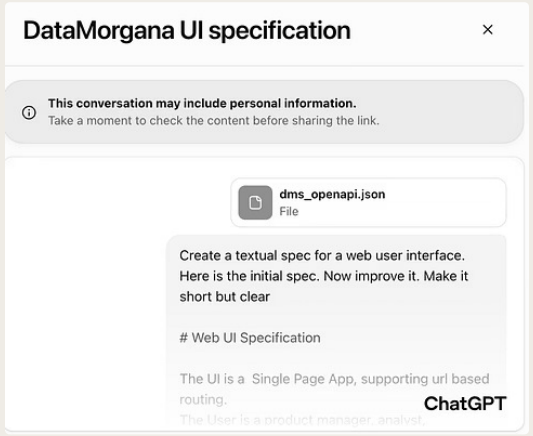
Core User Flows

1. Benchmark Generation

The UI should allow the user to generate a benchmark with increasing levels of customization:

1. **One-click default benchmark**
   - Use default corpus + default specification.
   - User only chooses number of conversations.
2. **Default benchmark with scenario override**
   - Provide a small "specification overrides" form (scenario, question/answer guidelines, max turns).
3. **Inline corpus benchmark**
   - User provides up to two inline text documents.
   - Uses default specification.
4. **Uploaded corpus benchmark**
   - Step 1: Upload corpus (UI manages multipart upload + status monitoring).

Made with GAMMA



Made with GAMMA

Skip to content Chat history

You said: dms_openapi.json File Create a textual spec for a web user interface. Here is the initial spec. Now improve it. Make it short but clear

Web UI Specification

The UI is a Single Page App, supporting url based routing.
The User is a product manager, analyst, researcher or developer in the field of RAG.

The user's goal is to:

1. Create a benchmark for a RAG system.
2. Evaluate the benchmark.

Future versions will expand beyond just RAG by adding support for more generic Agentic systems.

Use cases

Typical use cases / user flows are described below in ascending order of complexity and priority:

Benchmark Generation

1. Create a benchmark using the default corpus and default configuration ("single click" after login)
2. Create a benchmark using the default corpus, using the default configuration but overriding just the scenario description.
3. Create a benchmark using the inline-corpus, a list of up to two texts. This is useful for quick experimentation.
4. Create a benchmark using the user's uploaded corpus.
   a. First upload a corpus
   b. Then generate a benchmark using the corpus
5. Create a benchmark using the default corpus, using the user's own configuration (benchmark specification)
   a. First upload a specification
   b. Then generate a benchmark using the specification
6. Create a benchmark using the user's own corpus and own specification
   a. Upload corpus
   b. Upload specification
   c. Create a benchmark using the corpus and specification

Benchmark Evaluation

1. Evaluate a benchmark result using the default metrics
2. Evaluate a benchmark result using a selected subset of metrics

Content management

Corpus

1. List all my corpora (pending access control)
2. Get details about a corpus
   a. Be able to view its content (we're still not sure we want to allow that. Maybe just a list of files and dates, not full contents)
3. Create a corpus
   a. Monitor its indexing process
4. Delete a corpus

Specification

1. Get default specification (readonly)
2. Upload a specification
3. List all my specifications
4. Get a full specification I have read access to
5. Delete a specification (pending permissions)

Benchmarks

1. Create a benchmark
   a. Monitor its progress
2. List all my benchmarks
3. Download a ready benchmark
4. Delete a benchmark

Evaluations

1. Create an evaluation
   a. Monitor its progress
2. List all my evaluations
3. Download a ready evaluation result
4. Delete an evaluation

The product connects to a backend that supports the following API attached in the file dms_openapi.json ChatGPT said: Below is an improved, shorter, clearer textual UI specification. It keeps the structure but removes duplication, clarifies responsibilities, and aligns the UI with the backend API capabilities described in dms_openapi.json dms_openapi

.

Web UI Specification (Revised) Overview A single-page application with URL-based routing. Primary users: product managers, analysts, researchers, and developers working with RAG systems.

Main goals:

Create benchmarks.

Evaluate benchmarks.

Future versions will extend to general agentic evaluations.

Core User Flows

1. Benchmark Generation The UI should allow the user to generate a benchmark with increasing levels of customization:

One-click default benchmark

Use default corpus + default specification.

User only chooses number of conversations.

Default benchmark with scenario override

Provide a small "specification overrides" form (scenario, question/answer guidelines, max turns).

Inline corpus benchmark

User provides up to two inline text documents.

Uses default specification.

Uploaded corpus benchmark

Step 1: Upload corpus (UI manages multipart upload + status monitoring).

Step 2: Generate benchmark using that corpus.

Benchmark with custom specification

Step 1: Upload YAML specification.

Step 2: Generate benchmark using default corpus.

Fully custom benchmark

Upload corpus → upload spec → generate benchmark.

For all benchmarks:

Show progress and status (pending, generating, ready, failed).

Provide download link when ready.

1. Benchmark Evaluation Evaluate a benchmark using default metrics

Upload executed benchmark file (UI uses presigned URL).

Monitor progress until results are ready.

Evaluation with selected metrics

UI allows selecting metric subsets (if/when supported).

Otherwise relies on default evaluation.

Each evaluation shows: status, metrics summary (if any), and a download link.

Resource Management Views Corpus Management List corpora with status filters.

View corpus details: name, status, timestamps, ingestion report, and optionally raw file list (not full content).

Create corpus: upload via provided presigned S3 URL and track indexing progress (uploaded → processing → ready).

Delete corpus (if not currently processing).

Specification Management View default specification (read-only).

Upload specification (YAML validation before storing).

List user specifications.

View full specification including YAML.

Delete specification (permissions allowing).

Benchmark Management List benchmarks and filter by status.

View benchmark details: status info, overrides, corpus link, specification link, statistics.

Download completed benchmark.

Delete benchmark.

Evaluation Management List evaluations and filter by status.

View evaluation details: metrics, problems, statistics.

Download evaluation results.

Delete evaluation.

UI Structure (High-Level) Dashboard: Quick actions (“Generate Benchmark”, “Upload Corpus”, “Run Evaluation”).

Corpora: List → Details → Create.

Specifications: List → Details → Upload.

Benchmarks: List → Create → Details + Download.

Evaluations: List → Create → Details + Download.

Help / Docs: Link to specification examples and API reference.

If you’d like, I can also turn this into:

A wireframe-level spec,

A React component hierarchy,

A UI copywriting spec, or

A full UX flow diagram.

Sources You said: Create a wireframe-level spec ChatGPT said: Below is a clean, concise wireframe-level specification for the DataMorgana UI. It focuses on page layouts, key components, interaction points, and navigation—sufficient for designers or engineers to begin building screens.

Wireframe-Level UI Specification Global Layout Top Navigation Bar

Logo + product name

Links: Dashboard, Corpora, Specifications, Benchmarks, Evaluations

User menu (profile, logout)

Main Content Area

Each route loads into a single-page container

Right-hand side uses a collapsible panel for logs/progress when relevant

Global Elements

Toast notifications for success/failure

Status badges (pending, processing, ready, failed, deleted)

Pagination controls (bottom of listing pages)

1. Dashboard Purpose: Quick-start actions and system overview.

Layout

Hero block: “Create Benchmark” (primary button: Generate Default Benchmark)

Quick Actions Row:

Upload Corpus

Upload Specification

Create Inline Benchmark

Run Evaluation

Recent Items Panels (3 cards):

Recent Benchmarks (list 5)

Recent Evaluations (list 5)

Active Jobs (benchmarks/evaluations with progress bars)

1. Corpus Management 2.1 Corpus List Page Layout

Header: Corpora + button “Upload New Corpus”

Table columns:

Name

Status (badge)

Created at

Updated at

Actions → View / Delete

Filter dropdown: Status

Pagination bar

2.2 Corpus Detail Page Header: Corpus name + status badge + Delete button

Sections

Summary Card

Name

ID

Status

Created / Updated / Indexed

Upload Info

“Raw data download” link (if allowed)

Ingestion report summary (numbers of files, chunks, errors)

Show ingestion warnings/errors (accordion)

Progress Panel (right side)

Progress states: uploaded → processing → ready

Live status polling

Associated Benchmarks (list of benchmarks using this corpus)

Actions

Delete corpus (disabled if processing)

1. Specification Management 3.1 Specification List Page Layout

Header: Specifications + “Upload Specification”

Table columns:

Name

Status

Description (truncated)

Created at

Actions → View / Delete

3.2 Specification Detail Page Sections

Summary Card: Name, Description, Status

YAML Viewer Panel (read-only, collapsible)

Metadata (created/updated timestamps)

Benchmarks using this specification (list)

Actions

Delete specification

1. Benchmark Generation & Management 4.1 Create Benchmark Page (Wizard-Style) Step A — Select Corpus Options

Default corpus (radio)

Inline documents

Text area for up to 2 documents (add/remove buttons)

Uploaded corpus

Dropdown of user corpora + link “Upload new corpus”

Step B — Specification Options

Use default specification (radio)

Show optional “Override fields”:

Scenario description (textarea)

Question guidelines

Answer guidelines

Max turns

Use existing specification

Dropdown list

Upload specification

File picker

Step C — Benchmark Settings Number of conversations (input)

Dry run checkbox

Review summary panel

Footer

Back / Next

“Generate Benchmark” button (final step)

4.2 Benchmark List Page Columns:

ID (clickable)

Corpus (inline or named corpus)

Specification

Status

Created at

Actions → View / Delete

Status badge + small progress bar for active jobs.

4.3 Benchmark Detail Page Header: Benchmark ID + Status badge + “Delete Benchmark”

Sections

Summary Tile

Corpus used

Specification used

Inline docs indicator (if any)

Number of conversations

Created / Updated / Completed timestamps

Status & Progress Panel

Current state: pending → generating → ready

Error messages (accordion)

Statistics when ready (turn counts, problems)

Results Section

“Download Benchmark File” (enabled when ready)

Related Evaluations

List evaluations referencing this benchmark

Button “Run Evaluation”

1. Evaluation Management 5.1 Create Evaluation Page Form Layout

Select benchmark (dropdown or quick-pick from recent)

Optional name field

Upload executed benchmark file (via presigned URL)

If metric selection becomes available: metric checklist section

Submit button: “Start Evaluation”

Right-side panel shows upload progress bar.

5.2 Evaluation List Page Columns:

Name

Benchmark

Status

Created at

Actions → View / Delete

Active ones show progress bar.

5.3 Evaluation Detail Page Sections

Summary Card

Evaluation name

Benchmark ID

Status

Timestamps

Status Panel

Status updates

Errors / warnings

Metrics (table or expandable JSON)

Statistics (if available)

Results

“Download Evaluation Results” (when ready)

1. Supporting / Shared Components Status Badges pending (gray)

generating / processing (blue)

ready (green)

failed (red)

deleted (strikethrough gray)

Progress Indicators Inline progress bar in listings

Detailed progress view in right-hand panel

YAML Viewer Collapsible panel with syntax highlighting

Copy-to-clipboard button

Document Editor (inline corpus) Simple multi-text-box editor

Character / token counter

Confirmation Dialogs Delete corpus

Delete specification

Delete benchmark

Delete evaluation

If you'd like, I can also produce:

Visual mockups (ASCII or Figma-ready)

Component hierarchy for React / Vue / Svelte

User journey diagrams

Screen-by-screen copywriting

Just tell me the format you prefer.

You said: full ux flow diagram ChatGPT said: Below is a complete UX flow diagram in Mermaid format (easy to paste into Markdown, Notion, GitHub, or Figma Mermaid plugin). It captures all primary flows: corpora, specifications, benchmarks, evaluations, dashboard entry points, and cross-navigation.

If you'd like, I can also generate separate diagrams per flow, swimlane diagrams, or Figma-ready structured flow nodes.

Full UX Flow Diagram (Mermaid) flowchart TD

%% ================================ %% GLOBAL ENTRY %% ================================

A[User Logs In] --> B[Dashboard]

B --> BA[Quick Action: Generate Default Benchmark] B --> BB[Quick Action: Upload Corpus] B --> BC[Quick Action: Upload Specification] B --> BD[Quick Action: Create Inline Benchmark] B --> BE[Quick Action: Run Evaluation] B --> BF[View Recent Benchmarks] B --> BG[View Recent Evaluations]

%% ================================ %% CORPUS MANAGEMENT %% ================================

subgraph CORPUS[Corpus Management] C1[Corpora List] -->|Click 'View'| C2[Corpus Detail] C1 -->|Click 'Upload Corpus'| C3[Corpus Upload Form]

```
C3 --> C4[Multipart Upload → Status Polling]
C4 -->|processing → ready| C2
C4 -->|error| C2
```

```
C2 -->|Delete Corpus| C5{Confirm Delete?}
C5 -->|Yes| C6[Delete Corpus → Back to List]
C5 -->|No| C2
```

end

BB --> C3 BF --> C1

%% ============================== %% SPECIFICATION MANAGEMENT %% ==============================

subgraph SPEC[Specification Management] S1[Specification List] -->|View| S2[Specification Detail]

```
S1 -->|Upload Specification| S3[Upload Spec Form]
S3 --> S4[Validation → Ready]

S2 -->|Delete| S5{Confirm Delete?}
S5 -->|Yes| S6[Delete Specification → Back to List]
S5 -->|No| S2
```

end

BC --> S3

%% ============================== %% BENCHMARK GENERATION %% ==============================

subgraph BM[Benchmark Generation] BG1[Create Benchmark Wizard → Step 1: Select Corpus]

```
BG1 -->|Inline Documents| BG1A[Enter Up to 2 Docs]
BG1 -->|Use Default Corpus| BG1B[Default Corpus Selected]
BG1 -->|Choose Uploaded Corpus| C1

BG1 --> BG2[Step 2: Choose Specification]

BG2 -->|Default Spec| BG2A[Optional Overrides]
BG2 -->|Choose Existing Spec| S1
BG2 -->|Upload Spec| S3

BG2 --> BG3[Step 3: Settings → n_conversations, dry_run]

BG3 --> BG4[Step 4: Review Summary]
BG4 --> BG5[Submit: Generate Benchmark]

BG5 --> BG6[Benchmark Status Page]
BG6 -->|pending → generating → ready| BG7[Benchmark Ready]
BG6 -->|failed| BG8[Benchmark Error]

BG7 -->|Download| BG9[Benchmark File Download]
BG7 -->|Run Evaluation| E1
```

end

BA --> BG1 BD --> BG1 C2 --> BG1 S2 --> BG1 BF --> BG6

%% ============================== %% BENCHMARK LIST %% ==============================

subgraph BML[Benchmark Management] BL1[Benchmark List] BL1 -->|View| BL2[Benchmark Detail] BL2 -->|Delete| BL3{Confirm Delete?} BL3 -->|Yes| BL4[Delete → Back to List] BL3 -->|No| BL2 end

BF --> BL1

%% ============================== %% EVALUATION FLOW %% ==============================

subgraph EV[Evaluation Management]

```
%% Create Evaluation
E1[Create Evaluation Form → Select Benchmark]
E1 --> E1A[Upload Executed Benchmark File (Presigned URL)]
E1A --> E2[Evaluation Status Page]

%% Status Flow
E2 -->|pending → processing → ready| E3[Evaluation Ready]
E2 -->|failed| E4[Evaluation Error]

E3 -->|Download Results| E5[Results File]

%% Evaluation List
EL1[Evaluation List] -->|View| EL2[Evaluation Detail]
EL2 -->|Delete| EL3{Confirm Delete?}
EL3 -->|Yes| EL4[Delete → Back to List]
EL3 -->|No| EL2
```

end

BE --> E1 BG7 --> E1 BG --> EL1 BG --> EL1 What This Diagram Includes ✔️ End-to-end flows: login → dashboard → all resource operations ✔️ Corpus, specification, benchmark, and evaluation lifecycles ✔️ Cross-navigation paths (e.g., from Benchmark → Run Evaluation) ✔️ Error and success branches ✔️ Use-case entry points (Dashboard quick actions) ✔️ Wizard UI for benchmark creation ✔️ Realistic backend-aligned flows (upload → poll → ready)

If you want, I can also generate:

✅ A Figma flow structure (as a JSON or node list) ✅ Swimlane diagram (User / UI / Backend) ✅ Component hierarchy matching the wireframes ✅ Clickable prototype text script for designers

Just tell me what format you prefer next.

You said: separate please ChatGPT said: Sure — here are separate, clean UX flow diagrams, one per domain: Dashboard, Corpus, Specification, Benchmark Generation, Benchmark Management, and Evaluation.

All diagrams are Mermaid and self-contained.

1. Dashboard Flow flowchart TD

A[User Logs In] --> B[Dashboard]

B --> BA[Generate Default Benchmark] B --> BB[Upload Corpus] B --> BC[Upload Specification] B --> BD[Create Inline Benchmark] B --> BE[Run Evaluation]

B --> BF[Recent Benchmarks → Benchmark Detail] B --> BG[Recent Evaluations → Evaluation Detail] 2. Corpus Management Flow flowchart TD

C1[Corpora List] -->|View| C2[Corpus Detail] C1 -->|Upload Corpus| C3[Corpus Upload Form]

C3 --> C4[Multipart Upload + Status Polling] C4 -->|processing → ready| C2 C4 -->|failed| C2

C2 -->|Delete| C5{Confirm Delete?} C5 -->|Yes| C6[Delete Corpus → Back to List] C5 -->|No| C2

C2 -->|Use in Benchmark| BG1[Benchmark Wizard] 3. Specification Management Flow flowchart TD

S1[Specification List] -->|View| S2[Specification Detail] S1 -->|Upload Specification| S3[Upload Specification Form]

S3 --> S4[Validate YAML → Ready]

S2 -->|Delete| S5{Confirm Delete?} S5 -->|Yes| S6[Delete → Back to List] S5 -->|No| S2

S2 -->|Use in Benchmark| BG2[Benchmark Wizard] 4. Benchmark Generation Flow (Wizard) flowchart TD

BG1[Step 1: Select Corpus] --> BG1A[Inline Documents] BG1 --> BG1B[Default Corpus] BG1 -->|Select Uploaded Corpus| C1[Corpora List]

BG1 --> BG2[Step 2: Select Specification]

BG2 --> BG2A[Default Spec + Optional Overrides] BG2 -->|Choose Spec| S1[Spec List] BG2 -->|Upload Spec| S3[Upload Spec]

BG2 --> BG3[Step 3: Settings (n_conversations, dry_run)]

BG3 --> BG4[Step 4: Review Summary]

BG4 --> BG5[Submit Benchmark]

BG5 --> BG6[Benchmark Status Page]

BG6 -->|pending → generating → ready| BG7[Benchmark Ready] BG6 -->|failed| BG8[Benchmark Error]

BG7 -->|Download| BG9[Benchmark File] BG7 -->|Run Evaluation| E1[Create Evaluation] 5. Benchmark Management Flow (List + Detail) flowchart TD

BL1[Benchmark List] -->|View| BL2[Benchmark Detail]

BL2 -->|Download (if ready)| BL3[Benchmark File] BL2 -->|Run Evaluation| E1[Create Evaluation]

BL2 -->|Delete| BL4{Confirm Delete?} BL4 -->|Yes| BL5[Delete → Back to List] BL4 -->|No| BL2

BL1 -->|Create Benchmark| BG1[Benchmark Wizard] 6. Evaluation Management Flow flowchart TD

E1[Create Evaluation Form] --> E1A[Upload Executed Benchmark File (Presigned URL)] E1A --> E2[Evaluation Status Page]

E2 -->|pending → processing → ready| E3[Evaluation Ready] E2 -->|failed| E4[Evaluation Error]

E3 -->|Download Results| E5[Results File]

EL1[Evaluation List] -->|View| EL2[Evaluation Detail] EL2 -->|Delete| EL3{Confirm Delete?} EL3 -->|Yes| EL4[Delete → Back to List] EL3 -->|No| EL2

EL1 -->|New Evaluation| E1 Want higher fidelity versions? I can generate any of the following:

✅ Swimlane diagram (User vs UI vs Backend) ✅ Figma-ready design structure (nodes & edges JSON) ✅ Clickable-prototype flow script for designers ✅ Sequence diagrams aligned with the API ✅ Wireframe mockups (ASCII or image)

Tell me which format you'd like next.

You said: The (and) cause syntax errors. How to escape them? ChatGPT said: In Mermaid, regular parentheses do not need escaping inside node labels, but they do need escaping inside edge text or special syntax contexts.

To escape parentheses safely everywhere, use a backslash:

(and) Examples that avoid syntax errors:

1. Node labels (usually safe without escaping) A[Download (Ready)] --> B[Next] This normally works.
2. Edge labels or special characters (must escape) A -->|Download (Ready)| B
3. Text that Mermaid interprets as part of its grammar (escape) A["Upload Executed Benchmark File (Presigned URL)"] --> B
4. Using parentheses in conditions A --> B{Confirm Delete? (Yes/No)} Practical Rule of Thumb Situation Do you need to escape (and)? How Inside square-bracket node labels like A[Foo (Bar)] ❌ No — Inside quoted labels "Foo (Bar)" ❌ Rarely Only if Mermaid complains Inside edge labels like `A --> Foo (Bar) B` Inside decision nodes {Yes (X)} ✅ Yes Yes (X) If you want, I can rewrite all your diagrams with the parentheses escaped automatically. Just say: "Regenerate all diagrams with escaped parentheses".

You said: Is it possible to make two charts in the same file?

e.g. something like

flowchart TD

```
BG1[Step 1: Select Corpus] --> BG1A[Inline Documents]
BG1 --> BG1B[Default Corpus]
BG1 -->|Select Uploaded Corpus| C1[Corpora List]

BG1 --> BG2[Step 2: Select Specification]

BG2 --> BG2A[Default Spec + Optional Overrides]
BG2 -->|Choose Spec| S1[Spec List]
BG2 -->|Upload Spec| S3[Upload Spec]

BG2 --> BG3["Step 3: Settings (n_conversations, dry_run)"]

BG3 --> BG4[Step 4: Review Summary]

BG4 --> BG5[Generate Benchmark]

BG5 --> BG6[Benchmark Status Page]

BG6 -->|pending → generating → ready| BG7[Benchmark Ready]
BG6 -->|failed| BG8[Benchmark Error]

BG7 -->|Download| BG9[Benchmark File]
BG7 -->|Run Evaluation| E1[Create Evaluation]
```

flowchart TD

```
BL1[Benchmark List] -->|View| BL2[Benchmark Detail]

BL2 -->|Download (if ready)| BL3[Benchmark File]
BL2 -->|Run Evaluation| E1[Create Evaluation]

BL2 -->|Delete| BL4{Confirm Delete?}
BL4 -->|Yes| BL5[Delete → Back to List]
BL4 -->|No| BL2

BL1 -->|Create Benchmark| BG1[Benchmark Wizard]
```

ChatGPT said: Yes — you can put multiple Mermaid charts in the same file, but each chart must be in its own fenced code block.

You cannot put two flowchart TD declarations inside the same Mermaid block. You can put as many separate charts as you want as long as each one is wrapped like this:

```
flowchart TD
    A --> B
```

```
flowchart TD
    X --> Y
```

That's all — two (or more) charts per file are completely valid.

Your Example, Fixed (Two Separate Diagrams)

```
flowchart TD

    BG1[Step 1: Select Corpus] --> BG1A[Inline Documents]
    BG1 --> BG1B[Default Corpus]
    BG1 -->|Select Uploaded Corpus| C1[Corpora List]

    BG1 --> BG2[Step 2: Select Specification]

    BG2 --> BG2A[Default Spec + Optional Overrides]
    BG2 -->|Choose Spec| S1[Spec List]
    BG2 -->|Upload Spec| S3[Upload Spec]

    BG2 --> BG3["Step 3: Settings \(n_conversations, dry_run\)"]

    BG3 --> BG4[Step 4: Review Summary]

    BG4 --> BG5[Generate Benchmark]

    BG5 --> BG6[Benchmark Status Page]

    BG6 -->|pending → generating → ready| BG7[Benchmark Ready]
    BG6 -->|failed| BG8[Benchmark Error]

    BG7 -->|Download| BG9[Benchmark File]
    BG7 -->|Run Evaluation| E1[Create Evaluation]
```

```
flowchart TD

    BL1[Benchmark List] -->|View| BL2[Benchmark Detail]

    BL2 -->|Download \(if ready\)| BL3[Benchmark File]
    BL2 -->|Run Evaluation| E1[Create Evaluation]

    BL2 -->|Delete| BL4{Confirm Delete\?}
    BL4 -->|Yes| BL5[Delete → Back to List]
    BL4 -->|No| BL2

    BL1 -->|Create Benchmark| BG1[Benchmark Wizard]
```

Parentheses are escaped where needed.

If you'd like, I can produce a full multi-diagram document combining all flows in clean, consistent formatting.

You said: Create visual mockups ChatGPT said: Great — I can create visual mockups, but first I need to know what fidelity you want.

Below are the options — just tell me which style you want:

🎨 Choose Your Mockup Style Option A — ASCII Wireframes (Low-Fi, fast) Example:

+--+ | Create Benchmark | +--+ | Corpus: [Default ▼] | | Spec: [Default Spec ▼] | | Overrides: | | Scenario: ______________________ | | Q/A Guidelines: __________________ | +--+ | Conversations: [50] | | Dry run: [ ] | +--+ | (Generate Benchmark) | +--+ Works well in plain text / documents.

Option B — Box Layout Mockups (Mid-Fi) A more structured block layout resembling early product design:

┌──┐ │
Dashboard │
├──┤ │
Quick Actions │ │ ┌─────────────┐ ┌─────────────┐ ┌─────────────┐ │ │
│ New Bench... │ │ Upload Corpus │ │ Run Eval │ │ │ └─────────────┘ └─────────────┘
└─────────────┘ │ │ │ │ Recent Benchmarks │ │
┌──┐ │ │ │
#1234 Ready 50 conv Download │ │ │
└──┘ │
└──┘

Readable, structured, looks like early design spec.

Option C — High-Level UI Panels with Shading (Mid-fi +) Using text-art to approximate spacing, alignment, and layout.

Example:

╒══╕ ║ Benchmark
Detail ║ ╘══╛

Status: READY Created: Jan 12, 2026 Corpus: uploaded_corpus_v3 Spec: custom_spec.yaml

──────────── Result ──────────── [Download benchmark.jsonl]

──────────── Statistics ──────── • Conversations: 50 • Avg turns: 4.2 • Problems: none Better for communicating layout intentions to designers/engineers.

Option D — Figma-style Node Mockups (High-level UI definitions) Describe the UI as if building frames in Figma, e.g.:

Frame: Benchmark Detail

- Header (left: title, right: status badge)
- Section: Summary (grid 2x3)
- Section: Actions (Download, Evaluate)
- Tabs: Overview | Statistics | Raw YAML Useful for transferring into design tools.

👉 What I need from you Tell me which option(s) you want mockups for:

Dashboard

Corpora List

Corpus Detail

Specification List

Specification Detail

Benchmark Creation Wizard

Benchmark List

Benchmark Detail

Evaluation Creation

Evaluation List

Evaluation Detail

You can reply with something like:

"Give me Option B mockups for dashboard + benchmark creation wizard."

Or:

"Create Option C mockups for all screens."

I'll generate them immediately.

You said: Create Option C mockups for all screens. ChatGPT said: Below are Option C mid-fidelity text-visual mockups for ALL SCREENS in the DataMorgana UI. They are clean, structured, and suitable as a handoff to designers or frontend engineers.

If you'd like a single combined PDF or a style-guide version afterward, I can generate that too.

🎨 Option C Mockups — Full UI Set

1. Dashboard

║ DASHBOARD ║

Generate Benchmark │ │ Upload Corpus │ │ (default settings) │ │ │

Upload Specification │ │ Run Evaluation │ │ (YAML) │ │ │

Recent Benchmarks #1243 READY 50 conv [View] [Download] #1242 FAILED — [View] #1241 GENERATING 35%

Recent Evaluations Eval #882 READY [View Results] Eval #879 PROCESSING 62% 2. Corpora List ║ CORPORA ║ [Upload New Corpus]

Name Status Created at Actions

legal_docs_v1 READY Jan 11, 2026 [View] [Delete] inline_test_02 FAILED Jan 10, 2026 [View] product_specs PROCESSING Jan 09, 2026 [View]

Page 1 / 4 < Prev | Next > 3. Corpus Detail ║ CORPUS DETAIL ║

Name: legal_docs_v1 Status: READY Created: Jan 11, 2026 Updated: Jan 11, 2026 Corpus ID: c_239012

[Delete Corpus]

Corpus Summary Total files: 145 Successful: 145 Skipped: 0 Malformed: 0 Total chunks: 9,821

Raw Data: [Download]

Ingestion Report • No errors during ingestion. • Indexed in 00:02:14.

Related Benchmarks Benchmark #1243 (READY) [View] Benchmark #1199 (FAILED) [View] 4. Specification List ║ SPECIFICATIONS ║ [Upload Specification]

Name Status Created Actions

default_spec READY Jan 1, 2026 [View] qa_scenario_1 READY Jan 11, 2026 [View] [Delete] broken_spec FAILED Jan 10, 2026 [View]

5. Specification Detail ║ SPECIFICATION DETAIL ║

Name: qa_scenario_1 Status: READY Created: Jan 11, 2026 Description: "Customer support oriented Q/A"

[Delete Specification]

YAML Specification ▼ specification.yaml

scenario_description: | You are a customer support agent... questions_guidelines: ... answer_guidelines: ... max_turns: 6

Related Benchmarks Benchmark #1243 READY [View] Benchmark #1180 READY [View] 6. Benchmark Creation Wizard — Step 1: Select Corpus ║ CREATE BENCHMARK — STEP 1 ║

Choose Corpus:

(•) Default Corpus

( ) Inline Corpus
Document 1: [textarea ....] Document 2: [textarea ....]

( ) Uploaded Corpus Select corpus: [▼ legal_docs_v1] [Upload New Corpus]

```
    [ Next > ]
```

1. Benchmark Creation Wizard — Step 2: Select Specification ║ CREATE BENCHMARK — STEP 2 ║

Specification:

(•) Use Default Specification
▼ Optional Overrides ---
Scenario Description: [multiline text........]
Question Guidelines: [......................]
Answer Guidelines: [......................]
Max Turns: [6]

( ) Use Existing Specification
[▼ qa_scenario_1]

( ) Upload Specification
[Choose File]

```
    [ < Back ]    [ Next > ]
```

1. Benchmark Creation Wizard — Step 3: Settings ║ CREATE BENCHMARK — STEP 3 ║

Benchmark Settings:

Number of conversations: [50] Dry run: [ ]

```
    [ < Back ]    [ Next > ]
```

1. Benchmark Creation — Review & Submit ║ CREATE BENCHMARK — REVIEW ║

Summary:

Corpus: legal_docs_v1
Specification: qa_scenario_1
Overrides: none
Conversations: 50
Dry run: false

Confirm

[Generate Benchmark]

1. Benchmark List ║ BENCHMARKS ║

[Create Benchmark]

ID Spec Corpus Status Actions

#1243 qa_scenario_1 legal_docs_v1 READY [View] #1242 default_spec inline FAILED [View] #1241 qa_scenario_1 product_specs GENERATING [View]

11. Benchmark Detail

BENCHMARK DETAIL ║ ╚══════════════════════════════════════╝

Benchmark #1243 Status: READY Created: Jan 11, 2026 Updated: Jan 11, 2026

Corpus: legal_docs_v1 Spec: qa_scenario_1 Conversations: 50

[Delete Benchmark] [Run Evaluation] [Download Benchmark]

────────────── Statistics ────────────── • Total conversations: 50
• Avg turns per convo: 4.2
• Problems: none

────────────── Events Log ────────────── • Job started at 12:51
• Generation completed at 12:54
12. Evaluation Creation ╔══════════════════════════════════════╗ ║ CREATE EVALUATION ║ ╚══════════════════════════════════════╝

Select Benchmark: [▼ Benchmark #1243 — READY]

Evaluation Name: [text input]

Upload Executed Benchmark File: [Choose File] progress: 78% ████████░░

[Start Evaluation] 13. Evaluation List ╔══════════════════════════════════════╗ ║ EVALUATIONS ║ ╚══════════════════════════════════════╝

──

──── ID Benchmark Status Created Actions

──

──── #882 #1243 READY Jan 11, 2026 [View] #879 #1241 PROCESSING Jan 10, 2026 [View] #840 #1180 FAILED Jan 08, 2026 [View]

──

──── 14. Evaluation Detail ╔══════════════════════════════════════╗ ║ EVALUATION DETAIL ║ ╚══════════════════════════════════════╝

Evaluation #882 Status: READY Benchmark: #1243 Created: Jan 11, 2026 Completed: Jan 11, 2026

[Delete Evaluation] [Download Results]

────────────── Metrics ────────────── • answer_accuracy: 0.83
• retrieval_score: 0.92
• hallucination_rate: 0.04

────────────── Statistics ────────────── • No evaluation errors
• Processed 50 conversations Want More? I can also generate:

✨ High-resolution rendered mockups (image-based) ✨ Clickable prototype script ✨ A condensed UX spec PDF ✨ Design tokens + component structure for React/Next.js

Just tell me what you want next.

You said: how to use figma in google docs? ChatGPT said: You can use Figma inside Google Docs, but not by embedding the actual interactive Figma file. Google Docs does not support live embeds of Figma frames (unlike Notion, Confluence, or Slack).

However, there are three supported ways to integrate Figma content into Google Docs:

✅ Option 1 — Use the official "Figma for Google Workspace" add-on (recommended) This gives you a simple "Insert from Figma" button.

How to install: Open any Google Doc.

Go to Extensions → Add-ons → Get add-ons.

Search for "Figma for Google Workspace".

Click Install.

How to use it: Open your Google Doc.

Go to Extensions → Figma.

Click Insert Figma file.

Paste a Figma file link or frame link.

A static image preview of the Figma frame will be inserted.

✔️ Updates automatically if you refresh the image ✔️ Keeps the design always in sync with Figma ⍰ Not interactive ⍰ View-only, not editable inside Docs

✅ Option 2 — Copy + Paste as Image (quickest) In Figma, select a frame.

Press ⌘ + C (Mac) or Ctrl + C (Windows).

Paste directly into Google Docs.

Pros: ✔️ Very fast ✔️ Preserves high resolution

Cons: ⍰ Not auto-updating ⍰ Requires manual re-paste if design changes

✅ Option 3 — Export PNG / JPG / SVG and upload In Figma: select the frame → Export → choose PNG/JPG/SVG.

Upload the exported file into Google Docs.

Good for final snapshots or documentation with version control.

❌ What you cannot do in Google Docs ❌ Live interactive embeds

❌ Clickable prototypes

❌ Embedding Figma comments

❌ Real-time collaborative view

(Google Docs just doesn't support that API.)

Bonus Tip: If you need interactive Figma embeddings Use Google Slides, not Docs.

Google Slides supports:

Figma embeds using the "Add-on"

Automatic updating of linked Figma frames

Full-screen presentation of design screens

If you'd like I can also show you:

How to export multiple Figma frames into Docs automatically

How to create a "living design document" workflow

How to set up Figma → Docs auto-updatable screenshots

Just tell me!

You said: What would be a better fit for such app? React, Vue, Svelte, other? I want simplicity and not to write code, just generate it but make it easy for code generation tools to work on it ChatGPT said:

Thought for 16s Short answer: React + TypeScript (probably with Vite or Next.js) is your best bet.

Longer answer, but still human-sized 👇

What you care about You said:

You want simplicity

You don't really want to hand-write a lot of code

You want it to be easy for code generation tools (ChatGPT, Cursor, etc.) to work on it

Those constraints matter more than tiny DX differences between frameworks.

How frameworks compare for AI-generated apps React Pros

It's the default language of AI codegen right now.

Every model, example, template, and dev tool has React + TS as the first-class target.

Huge ecosystem of:

Component libraries (shadcn/ui, MUI, Mantine)

Table / chart / form libs

Auth & routing examples

Codegen tools "know" React idioms: hooks, context, forms, etc., so:

Fewer weird edge cases

Easier refactors ("convert to controlled component", "add a dialog", etc.)

Cons

Verbose compared to Svelte

You have to tolerate JSX and hook boilerplate

But since you're not writing most of it manually, the verbosity cost is smaller, and the ecosystem / codegen benefit wins.

Vue Vue 3 with

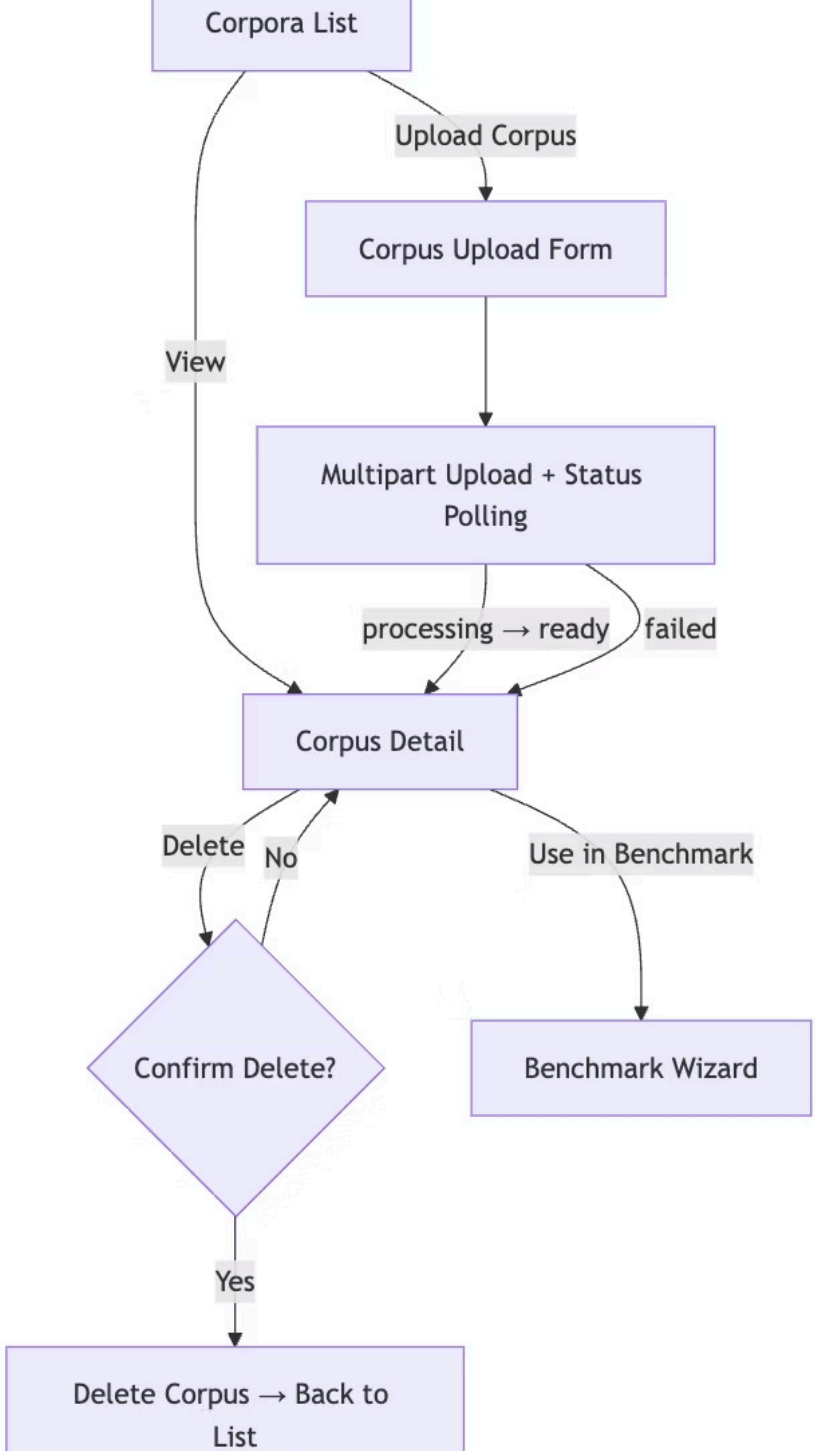
1

2

3

4

Made with GAMMA



Step 4: Freeze It into a UI Spec

Convert vibes into artifacts:

Markdown UI spec

Screen inventory

User flows

Component behavior

Edge cases

Attach wireframes and flow diagrams.

Made with GAMMA

🎨 Option C Mockups — Full UI Set

1. Dashboard

less

Copy code

```
╔═══════════════════════════════════════════════════════════════╗
║                          DASHBOARD                            ║
╚═══════════════════════════════════════════════════════════════╝

┌──────────────────────────┐   ┌──────────────────────────┐
│  Generate Benchmark      │   │     Upload Corpus        │
│  (default settings)      │   │                          │
└──────────────────────────┘   └──────────────────────────┘

┌──────────────────────────┐   ┌──────────────────────────┐
│ Upload Specification     │   │  Run Evaluation          │
│ (YAML)                   │   │                          │
└──────────────────────────┘   └──────────────────────────┘


──────────────── Recent Benchmarks ────────────────
#1243   READY      50 conv     [View] [Download]
#1242   FAILED     –           [View]
```

Made with GAMMA

2. Corpora List

pgsql

Copy code

```
╔═══════════════════════════════════════════════════════════╗
║                         CORPORA                           ║
╚═══════════════════════════════════════════════════════════╝

[ Upload New Corpus ]

──────────────────────────────────────────────────────────────
Name                 Status        Created at        Actions
──────────────────────────────────────────────────────────────
legal_docs_v1        READY         Jan 11, 2026      [View] [Delete]
inline_test_02       FAILED        Jan 10, 2026      [View]
product_specs        PROCESSING    Jan 09, 2026      [View]
──────────────────────────────────────────────────────────────

                    Page 1 / 4          < Prev | Next >
```

Made with GAMMA

3. Corpus Detail

yaml

Copy code

```
╔═══════════════════════════════════════╗
║              CORPUS DETAIL            ║
╚═══════════════════════════════════════╝

Name: legal_docs_v1
Status: READY
Created: Jan 11, 2026     Updated: Jan 11, 2026
Corpus ID: c_239012

[ Delete Corpus ]


──────────────── Corpus Summary ────────────────
Total files:      145
Successful:        145
Skipped:           0
Malformed:         0
Total chunks:      9,821

Raw Data:   [ Download ]

──────────────── Ingestion Report ──────────────
• No errors during ingestion.
• Indexed in 00:02:14.

──────────────── Related Benchmarks ────────────
```

Made with GAMMA

What Wireframes Are (and Aren’t)

Wireframes are:

- Structural
- Focused on layout
- About hierarchy and flow

Wireframes are not:

- Pixel-perfect
- Visual design

They exist to reduce ambiguity.

Think of them as drawing screens on a napkin.

Wireframe

2. Corpora List

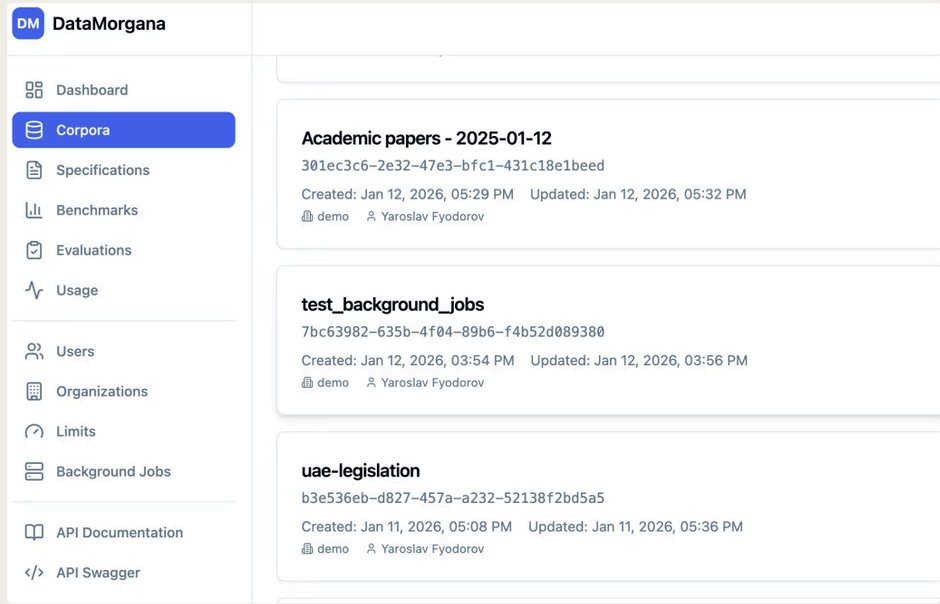
pgsql Copy code

```
╔═══════════════════════════════════════════════╗
║                    CORPORA                    ║
╚═══════════════════════════════════════════════╝

[ Upload New Corpus ]

──────────────────────────────────────────────────────────
Name              Status       Created at       Actions
──────────────────────────────────────────────────────────
legal_docs_v1     READY        Jan 11, 2026     [View] [Delete]
inline_test_02    FAILED       Jan 10, 2026     [View]
product_specs     PROCESSING   Jan 09, 2026     [View]
──────────────────────────────────────────────────────────
                  Page 1 / 4          < Prev | Next >
```

Reality



Made with GAMMA

Why Flow Diagrams Matter for AI

- Make state explicit
- Clarify transitions
- Act as shared memory
- Reduce hallucination

Mermaid diagrams work especially well.

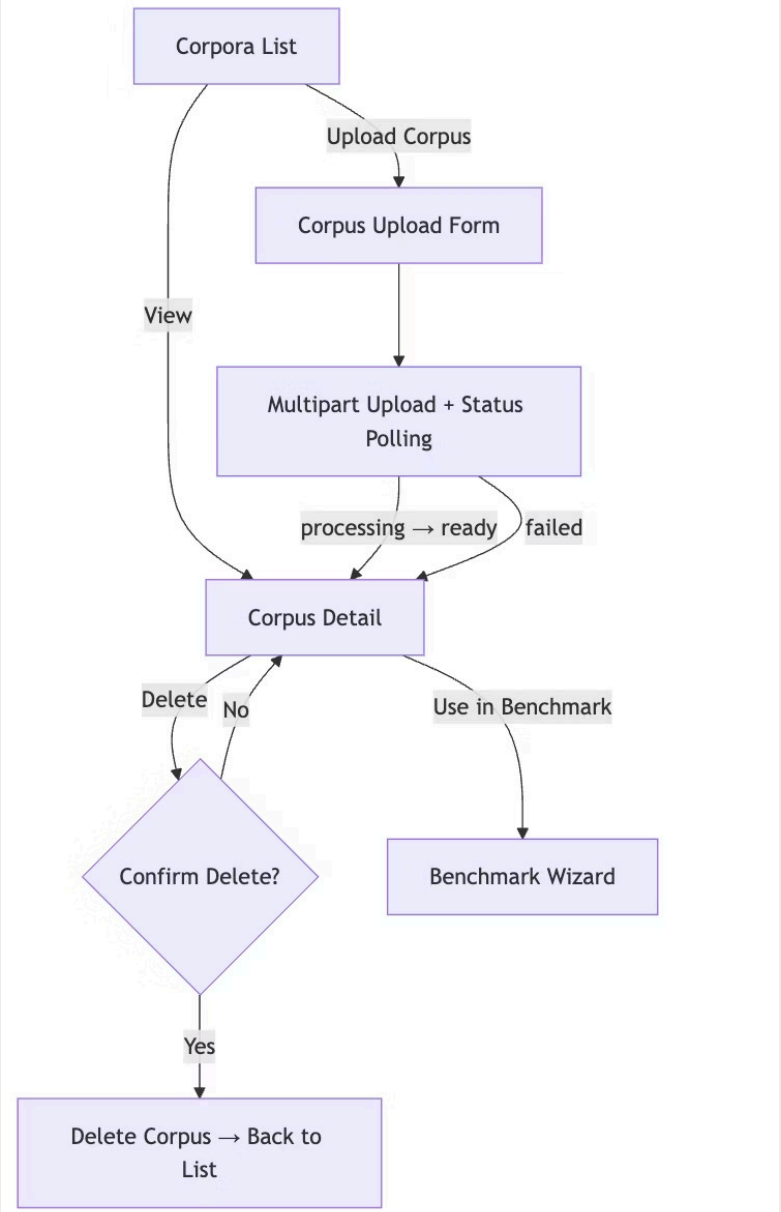
```
flowchart TD

C1[Corpora List] -->|View| C2[Corpus Detail]
C1 -->|Upload Corpus| C3[Corpus Upload Form]

C3 --> C4[Multipart Upload + Status Polling]
C4 -->|processing → ready| C2
C4 -->|failed| C2

C2 -->|Delete| C5{Confirm Delete?}
C5 -->|Yes| C6[Delete Corpus → Back to List]
C5 -->|No| C2

C2 -->|Use in Benchmark| BG1[Benchmark Wizard]
```



Made with GAMMA

Step 5: Choose Boring, Popular Tech

Typical choices:

- React + TypeScript
- TailwindCSS
- shadcn / MUI / Mantine

Not the best stack — the *most predictable* one.

Made with GAMMA

Ask not what your agent can do for you,

But what can you do for your agent

Made with GAMMA

Step 6: Scaffold First, Features Later

First prompt should generate:

- File structure
- Routing
- Layout
- Placeholder pages

No real business logic yet.

E.g. just build the infrastructure first. Filled with placeholders for later implementations.

Made with GAMMA

Create a web UI for this service. The requirements for the UI are described in the attached spec. The API is described in the file Data\ Morgana\ as\ a\ public\ service\ Oct\ 2025.md.
The web API spec is defined in dms_openapi.json

The stack needs to be:
React + TypeScript + Vite
TailwindCSS
Choose one of the component libraries MUI, Mantine, Chakra, shadcn/ui

Start with just laying out the basic file structure and building blocks that would allow the web UI to be used. Just a basic dashboard page without the rest of the app yet

Plan Auto

Made with GAMMA

Step 7: Implement One Feature at a Time

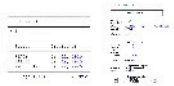
Example:

- `/corpora`
- `/corpus/:id`

Rules:

- Use only existing APIs. Explicitly ask it to refer to `opeapi.json`
- Follow established UI patterns

Made with GAMMA



Implement the Web UI for the Corpus Management section. See the reference docuement at Data Morgana as a public service...

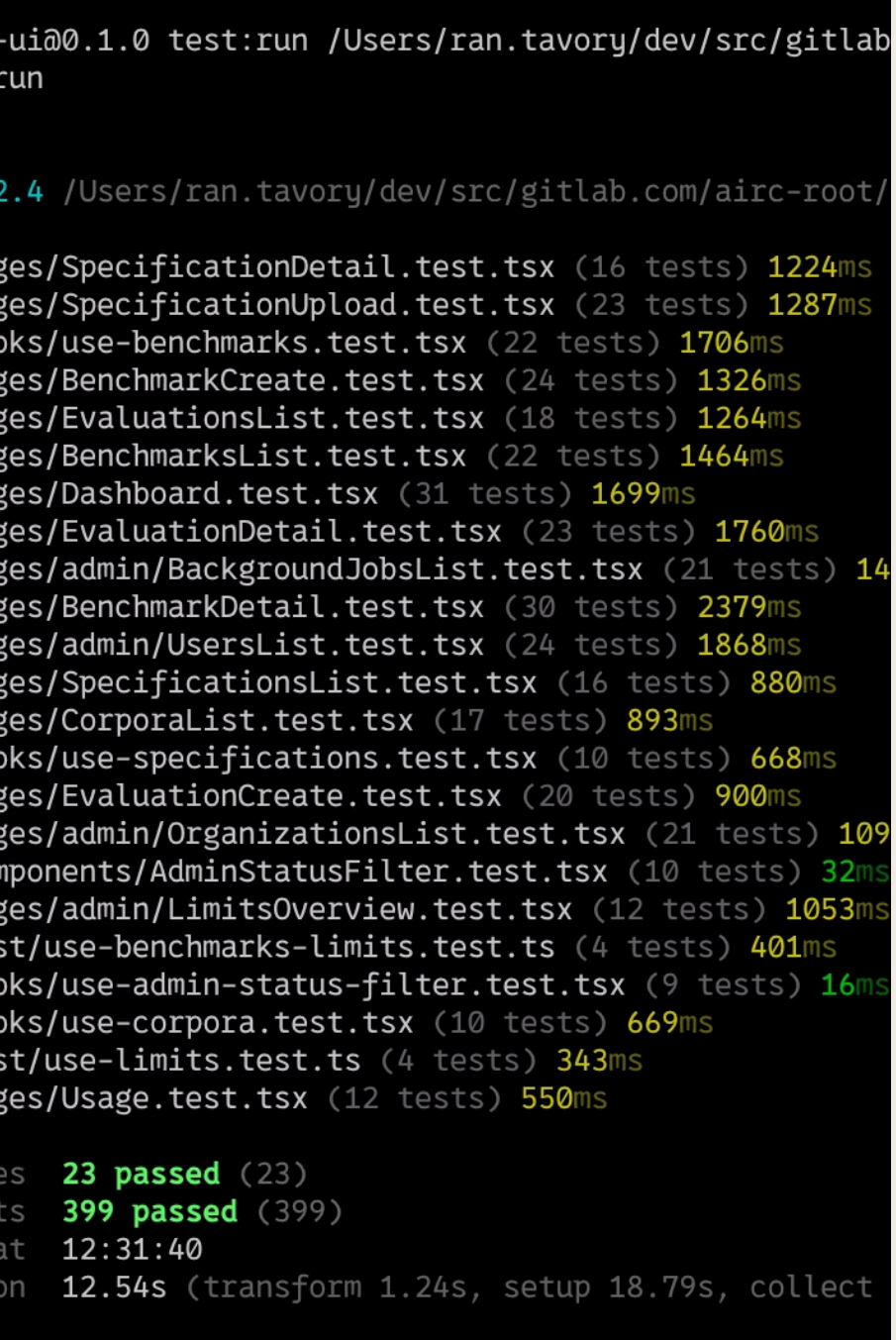
Some low fidelity Mockups found in list-corpora.jpg and in corpus-details.jpg

The available API is described at { } dms_openapi.json
Use only API that's available there, I want a working system. If some of the UI elements cannot be created then do not implement then and let me know.

Your goal is to hook into the exisign UI design under web-ui and use the same libraries and UI components there to create a consistent UI.
Focus your efforts now on `/corpora` and `/corpus/:id`

Plan Auto

Made with GAMMA



Step 8: Add Testing

- Discuss testing options first
- Add UI tests
- Manual sanity checks
- Lints

AI-generated code needs guardrails.

Made with GAMMA

What AI Coding agents need to succeed

- Clear, explicit **goals**
  - Product specs, wireframes, business rules
- Rich, concrete **context**
  - Technology stack, chosen libraries, existing codebase
- A reliable way to **verify** correctness
  - Automated tests are ideal

Made with GAMMA

What This Process Is Not

- Not fully automated
- Not zero thinking
- Not “prompt and pray”

Made with GAMMA

Conclusion

You’re not vibe coding.

You’re leading a team.

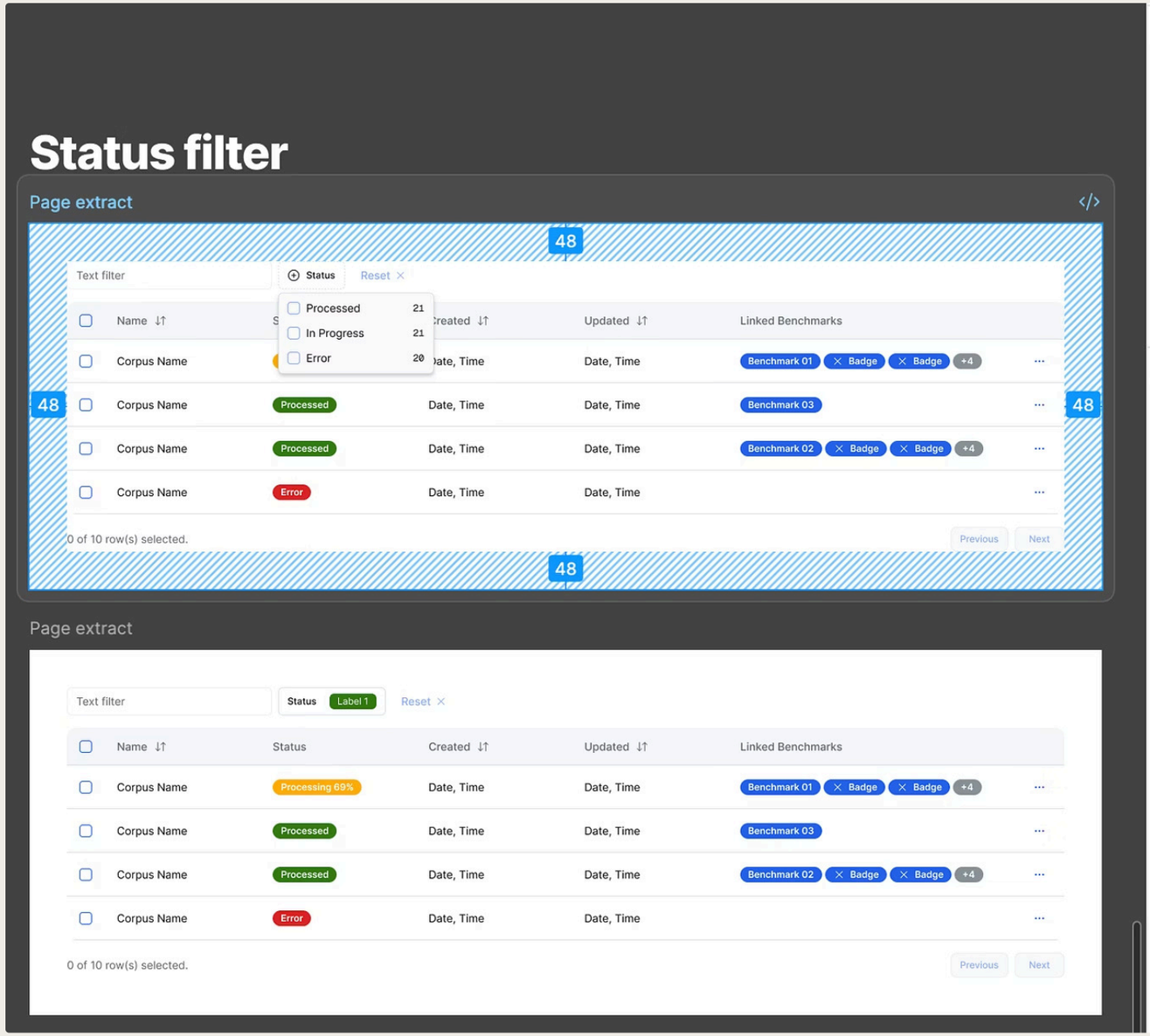
A team of junior, yet very capable coders.

If you instruct them clearly enough — they will succeed.

Actionable takeaway: write specs, not prompts. Specs are the guardrails that keep the model on track.

Made with GAMMA

Working with Figma



Made with GAMMA

What's the future like?

Ralph?

Made with GAMMA