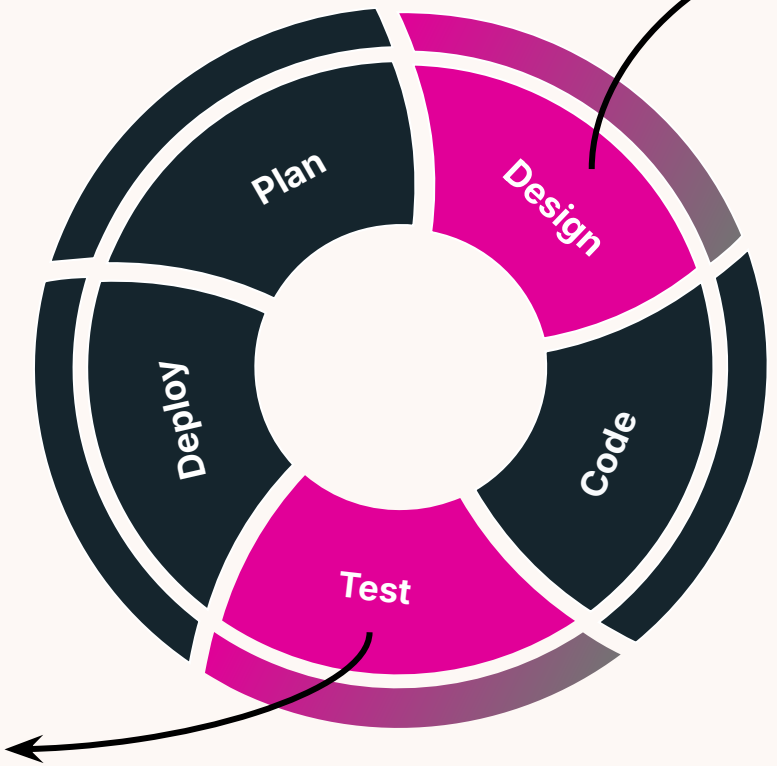
GraphQLConf
2026

hosted by GraphQL Foundation

Speed Without Sacrifice:
How Wayfair Transforms DevEx With AI

Maheswari Karlapudi, Muskan Kaur Sethi

Agenda



01. GraphQL Schema Reviews with GenAI

Instant feedback loops, faster design

02. GraphQL Response Mocking with GenAI

Faster testing & prototyping

Wayfair Supergraph

220+
SUBGRAPHS

5B+
DAILY REQUESTS

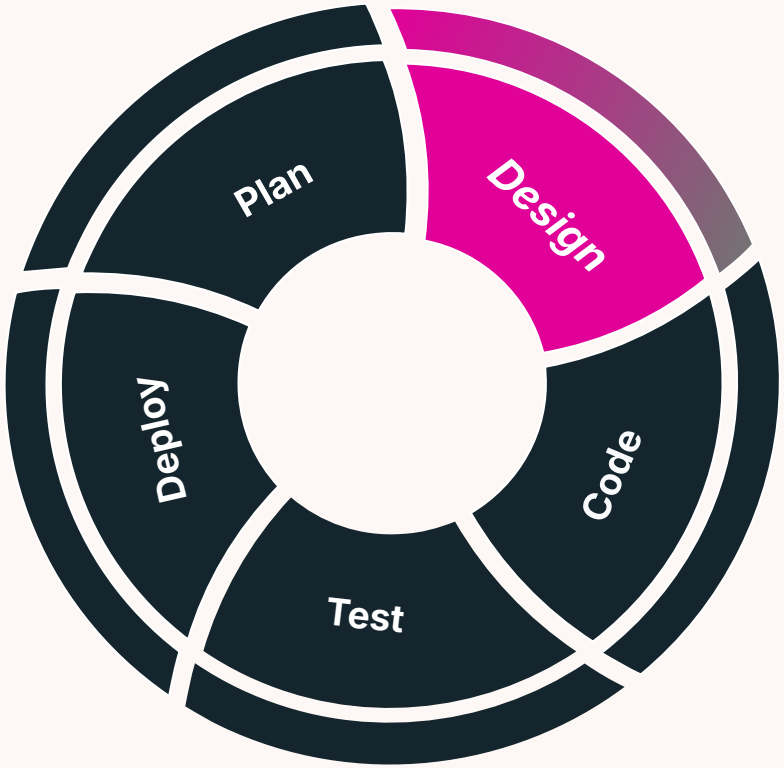
1200+
DEVELOPERS

2000+
GraphQL SCHEMA PUBLISHES

1

GraphQL Schema Reviews with GenAI

Instant feedback loops, faster design



Problem

Schema Reviews: **Slow, Painful and Costly**

⏱️ **2.5 days**

Time to first comment

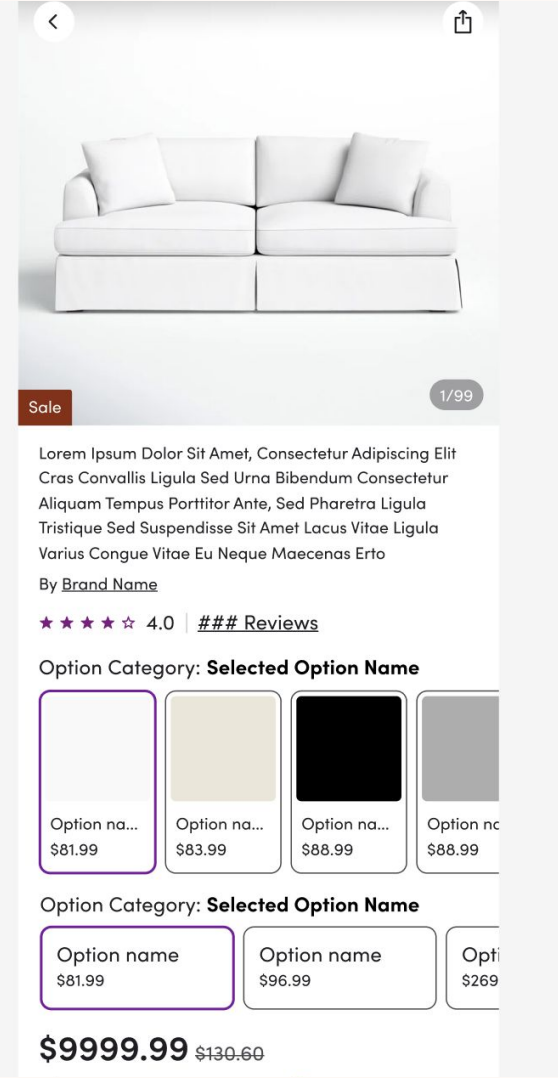
📅 **3 days**

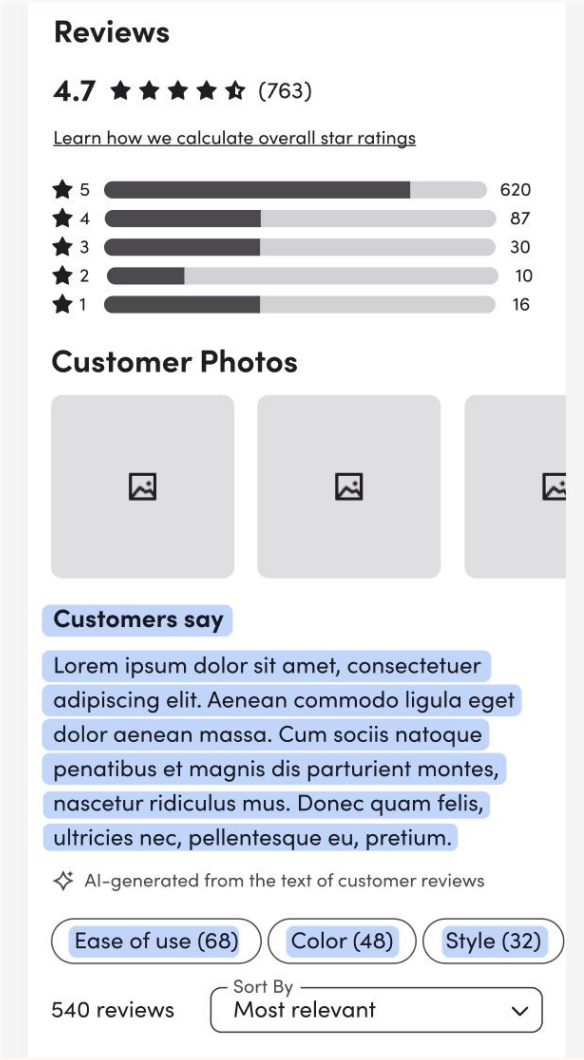
Time to approval

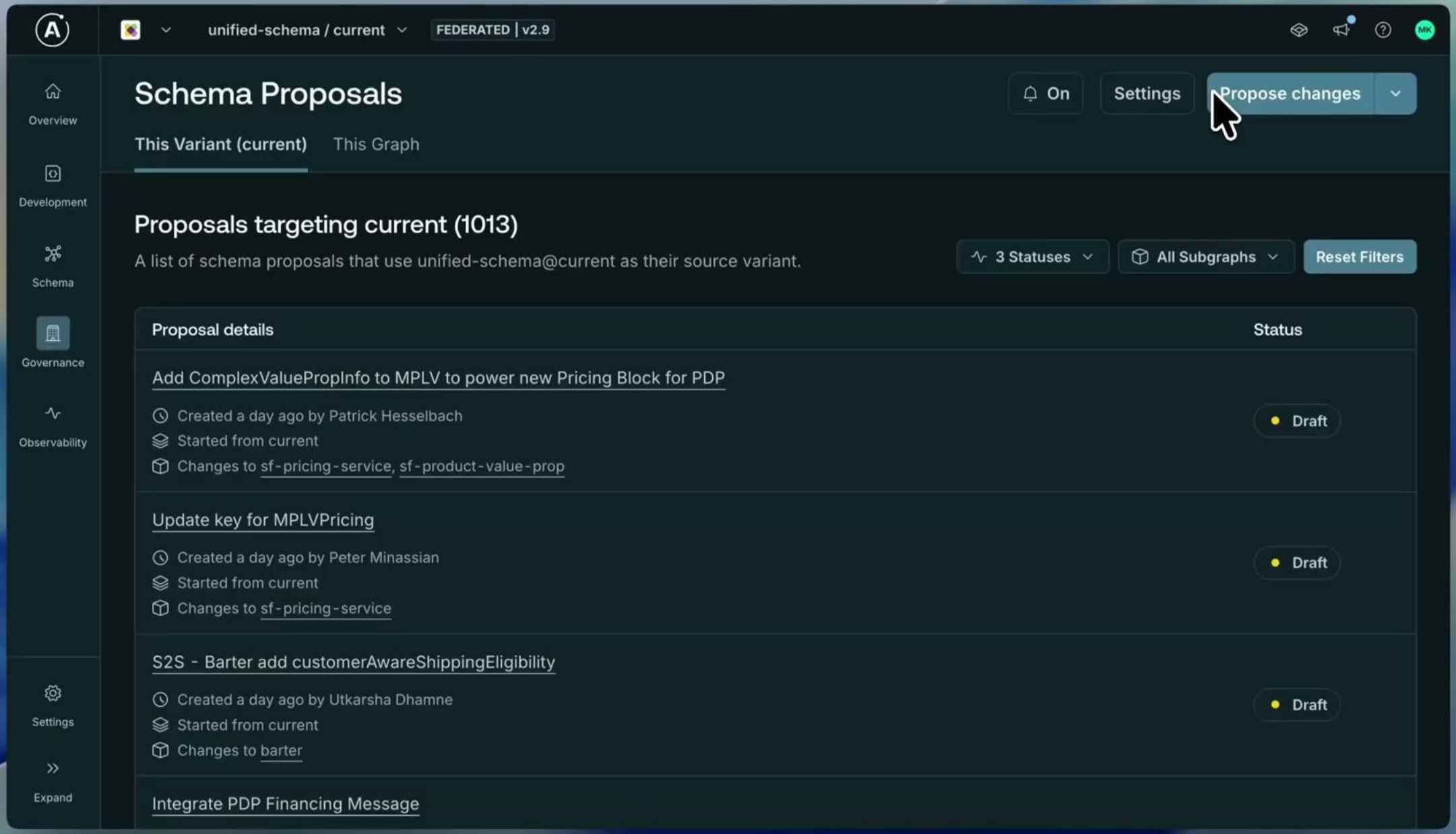
💸 **$350k**

Developer productivity loss

Solution: GraphQL Schema Copilot






unified-schema / current
FEDERATED | v2.9
Overview
Development
Schema
Governance
Observability
Settings
Expand
Schema Proposals
On
Settings
Propose changes
This Variant (current)
This Graph
Proposals targeting current (1013)
A list of schema proposals that use unified-schema@current as their source variant.
3 Statuses
All Subgraphs
Reset Filters
Proposal details
Status
Add ComplexValuePropInfo to MPLV to power new Pricing Block for PDP
Created a day ago by Patrick Hesselbach
Started from current
Changes to sf-pricing-service, sf-product-value-prop
Draft
Update key for MPLVPricing
Created a day ago by Peter Minassian
Started from current
Changes to sf-pricing-service
Draft
S2S - Barter add customerAwareShippingEligibility
Created a day ago by Utkarsha Dhamne
Started from current
Changes to barter
Draft
Integrate PDP Financing Message

Solution: Schema Copilot

Instant Architecture Review with AI

Real time feedback of GraphQL schema with no wait time

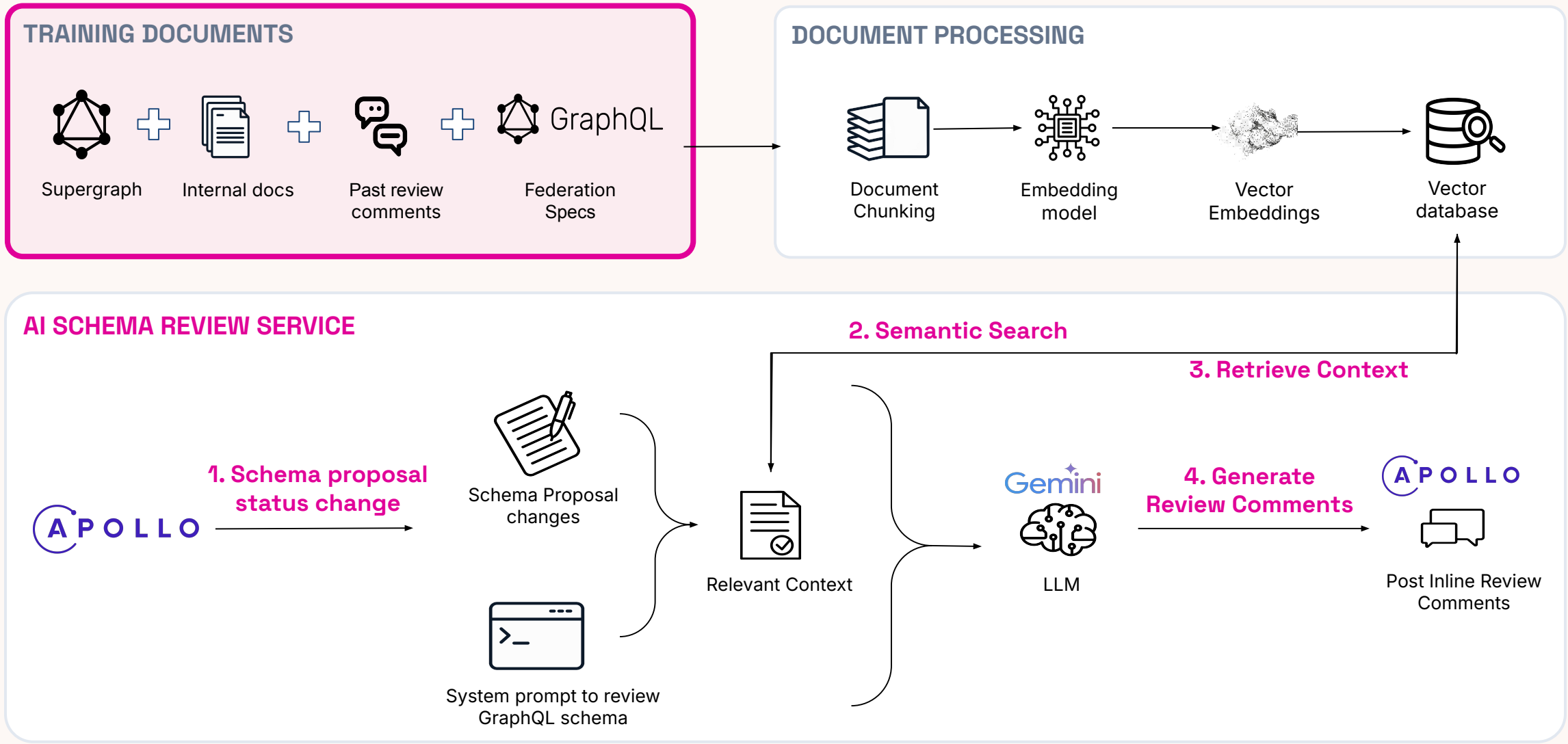
1-Click Auto-Fix

Agentic Copilot that can resolve common schema violations with a single click.

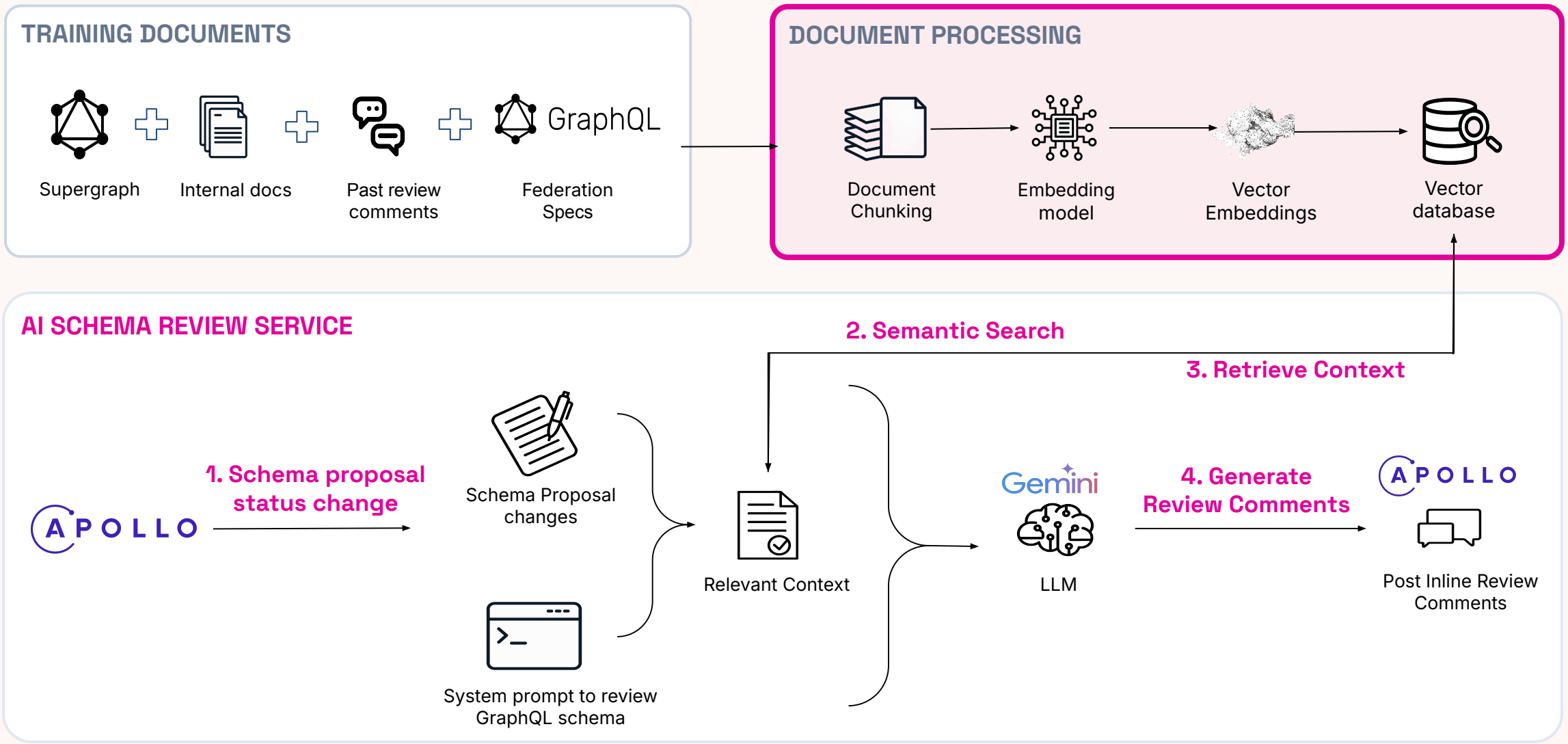
Corporate Memory Context

Leverages internal documentation to provide **contextually relevant feedback.**

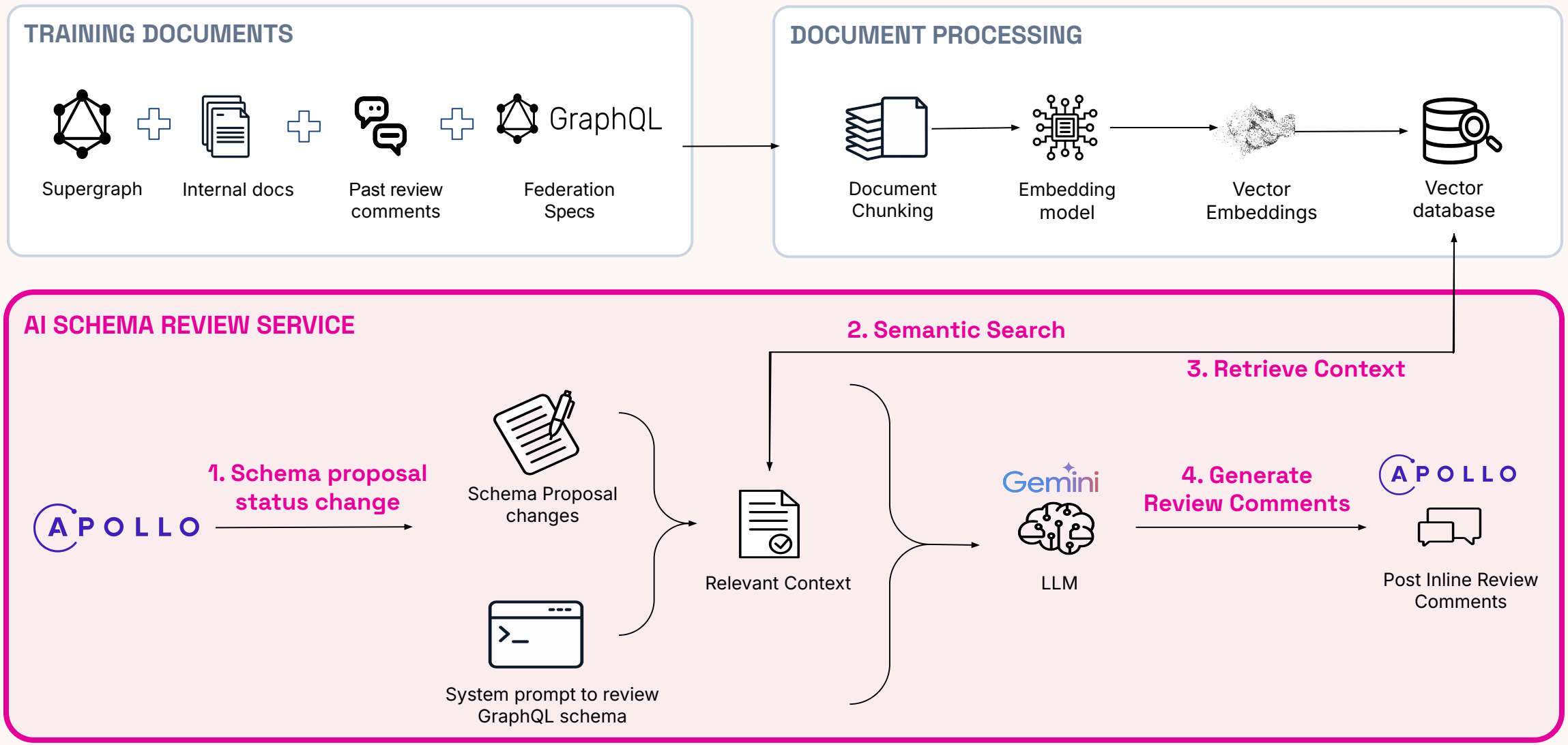
Architecture: RAG Pipeline



Architecture: Data Pre-processing



Architecture: RAG Pipeline



Infrastructure Cost

RAG: Each Review Cost

$0.05

- Embeddings for proposed changes
- LLM Review: $0.05
- 9k tokens

Indexing: One-Time Cost

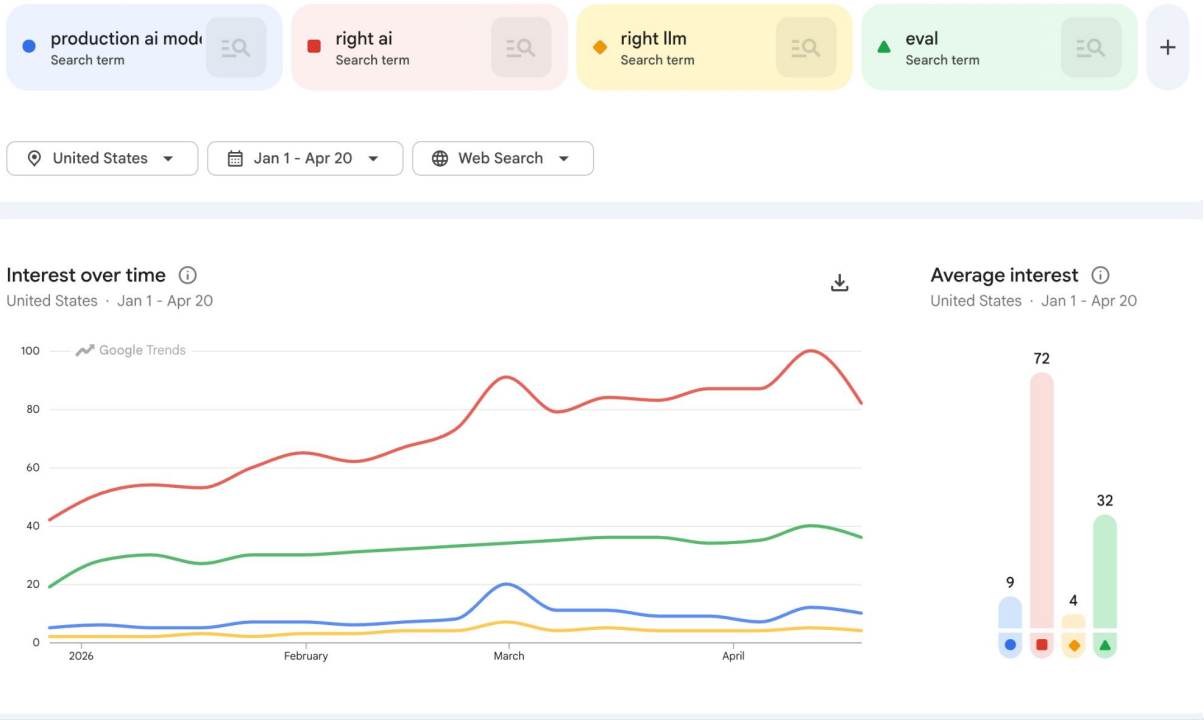
$0.45

- 12MB Knowledge Base
- text-embedding-005
- 3M tokens

Choosing the Right Brain

Comparison of Intentions

Normalized trending searches for “right” AI model on Google



Building Trust

Deterministic Metrics

Traditional NLP benchmarks like **BLEU** and **ROUGE** for quantitative assessment.

RAG Specific Metrics

The RAG Triad: **Faithfulness**, **Relevancy**, and **Contextual Relevancy** for end-to-end validation.

Human Centric Metrics

Tracking **manual edit distance** and **feedback surveys** to measure developer productivity.

LLM-as-Judge

Using high-capability models to evaluate **Semantic accuracy** and **rubrics**.

Building Trust

Deterministic Metrics

Traditional NLP benchmarks like **BLEU** and **ROUGE** for quantitative assessment.

RAG Specific Metrics

The RAG Triad: **Faithfulness**, **Relevancy**, and **Contextual Relevancy** for end-to-end validation.

Human Centric Metrics

Tracking **manual edit distance** and **feedback surveys** to measure developer productivity.

LLM-as-Judge

Using high-capability models to evaluate **Semantic accuracy** and **rubrics**.

Building Trust

Deterministic Metrics

Traditional NLP benchmarks like **BLEU** and **ROUGE** for quantitative assessment.

RAG Specific Metrics

The RAG Triad: **Faithfulness**, **Relevancy**, and **Contextual Relevancy** for end-to-end validation.

Human Centric Metrics

Tracking **manual edit distance** and **feedback surveys** to measure developer productivity.

LLM-as-Judge

Using high-capability models to evaluate **Semantic accuracy** and **rubrics**.

Building Trust: The Eval Engine

Deterministic Metrics

Traditional NLP benchmarks like **BLEU** and **ROUGE** for quantitative assessment.

RAG Specific Metrics

The RAG Triad: **Faithfulness**, **Relevancy**, and **Contextual Relevancy** for end-to-end validation.

Human Centric Metrics

Tracking **manual edit distance** and **feedback surveys** to measure developer productivity.

LLM-as-Judge

Using high-capability models to evaluate **Semantic accuracy** and **rubrics**.

Building Trust: LLM-as-Judge

Golden Dataset

Benchmarked against 100 historical schema reviews curated by super-users.

Format Requirements

- Valid JSON structures
- Correct subgraph keys
- Schema element validation

Review Criteria

- Naming convention checks
- Identify N+1 problems
- Lists vs. Paginations
- Security considerations

Constraint Adherence

- Exclude false positives
- Following instructions in the prompt

Building Trust: The Eval Engine

Model	Judge Score	Latency	Cost
Claude 4.5	80%	High	$$$$
Gemini 3.0 Pro	76%	Moderate	$$
Gemini 3.0 Flash	**83%**	**Ultra-Fast**	**$**

Agentic Copilot: 1-Click Apply Suggestion

01

LLM maps the exact AST coordinates of the schema.

02

Generates a structurally valid patch

03

Commits directly on single click after verifying new schema passes composition checks

Impact: Velocity Unlocked

2.5 days → 0

Time to first comment

3 days → 1.5

Time to approval

$150k

Developer productivity gains

Lessons Learned

📊 Measure

Identify your bottlenecks and where AI can help

🏃 Agile

Start small and iterate quickly

👩‍💻 Meet developers where they work

Integrating AI in applications where developers already work increases adoption and helps acceleration

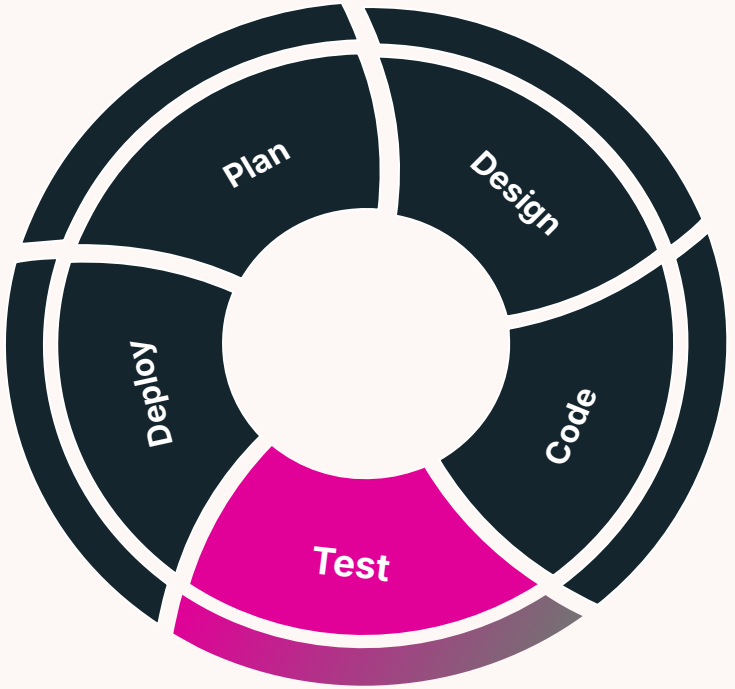
The Bottleneck Moves Downstream

Schema is approved. But the Frontend is still waiting.

2

GraphQL Response Mocking with GenAI

Faster testing & prototyping



Problem

Waiting. Reworking. Lost Velocity.

- **Backend Dependency**: Frontend teams blocked, waiting on backend schemas & mock data.
- **Disconnected Tools**: Mocks break as backend evolves → endless rework.
- **Low-Fidelity Data**: Unrealistic, static mocks don’t mirror production reality.

Solution

AI Schema mocks empower Frontend to prototype and test without backend reliance

Schema mocking with no backend strings attached!

Solution

AI Schema mocks empower Frontend to prototype and test without backend reliance

Rapid Prototyping

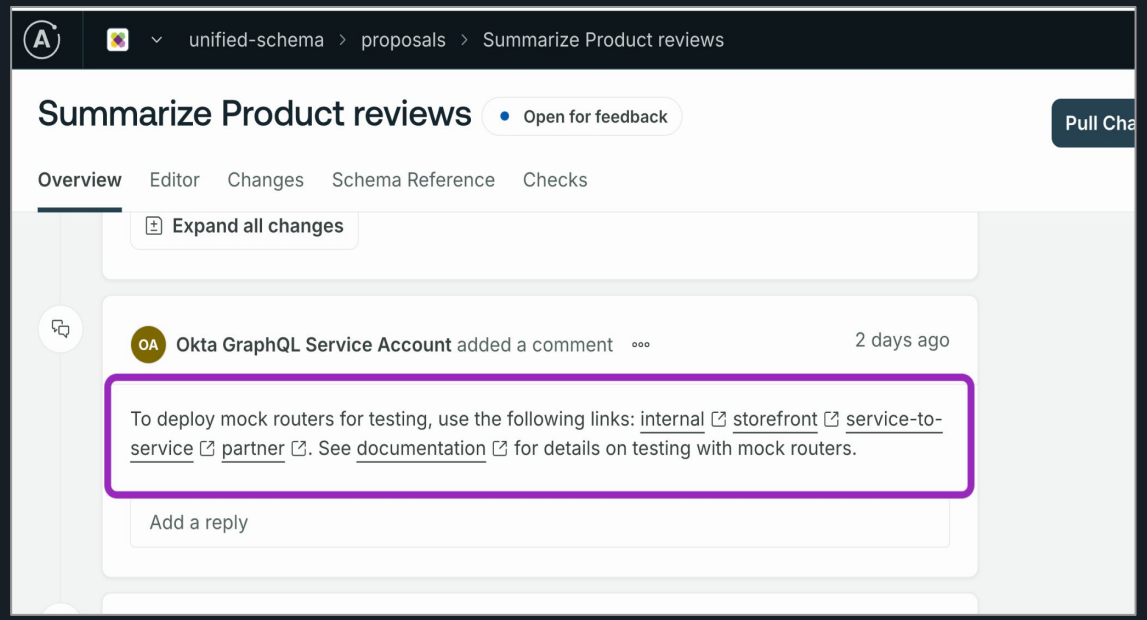
High-Fidelity Data

Decoupled Frontend

Step 1: Mock Server Deployment

Developers can deploy mock servers from schema proposals or via simple custom header.

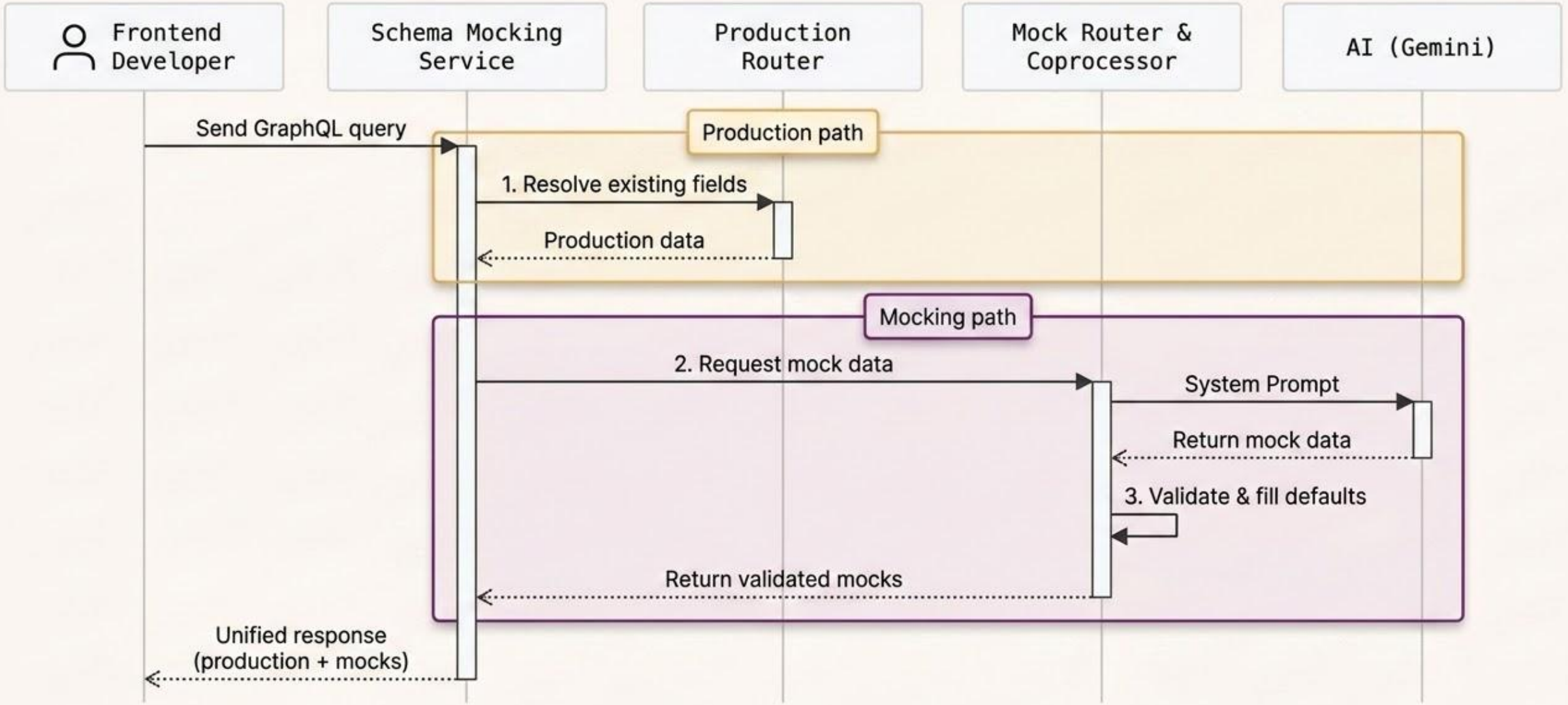
Option 1



Option 2

```
const link = createHttpLink({
    uri: 'https://www.wayfair-demo-summary-review.com/graphql',
    fetch,
    headers: {
      'x-clientname': 'summary-review-client',
      'x-client-version': 'client@1.0.0',
      'x-ai-mocking' : 'p-1099',
    },
});
```

Step 2: Schema Mocking



How Do We Split Query ?

AST Traversal

Client query strings are parsed into a structured Abstract Syntax Tree (AST) to allow programmatic traversal.

Schema Validation

We walk the tree, asking the production schema if specific field paths and argument names exist.

Nested Routing

Enforcing recursive validity: a field is only production-valid if every ancestor in its path is confirmed valid.

How Do We Split Query : An Example

Client Query

```
query ProductPage {
  product(id: "123") {
    title # old field
    label # new field
  }
}
```

Production Schema

```
type Query {
  product(id: ID!): Product
}
type Product {
  title: String!
}
```

Valid Production Query

```
query ProductPage {
  product(id: "123") {
    title
  }
}
```

New Mock Query

```
query ProductPage {
  product(id: "123") {
    label
  }
}
```

GraphQL Schema Mocking Service

Step 1

Supergraph + Query define the “data” shape

Step 2

Zod Schema

Encode “data” shape as typed, nested JSON contract.

Step 3

LLM

Natural language prompt + structural outline → generated JSON.

Step 4

Validate

Validate → patch defaults → return a complete data payload

GraphQL Schema Mocking Service

Step 1

Supergraph + Query define the “data” shape

Step 2

Zod Schema

Encode “data” shape as typed, nested JSON contract.

Step 3

LLM

Natural language prompt + structural outline → generated JSON.

Step 4

Validate

Validate → patch defaults → return a complete data payload

GraphQL Schema Mocking Service

Step 1

Supergraph + Query define the “data” shape

Step 2

Zod Schema

Encode “data” shape as typed, nested JSON contract.

Step 3

LLM

Natural language prompt + structural outline → generated JSON.

Step 4

Validate

Validate → patch defaults → return a complete data payload

GraphQL Schema Mocking Service

Step 1

Supergraph + Query define the “data” shape

Step 2

Zod Schema

Encode “data” shape as typed, nested JSON contract.

Step 3

LLM

Natural language prompt + structural outline → generated JSON.

Step 4

Validate

Validate → patch defaults → return a complete data payload

Impact: Less Waiting, More Shipping

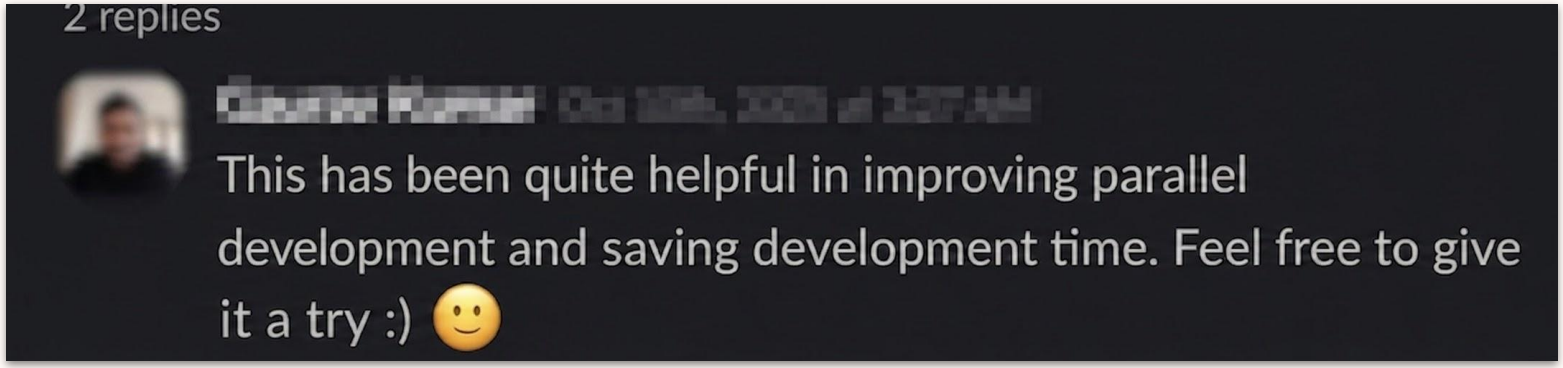
15%

Faster time to market

💵 **$100K***

Developer Productivity Gains

User Feedback :



* Developer Reported + Estimated

Lessons Learned

- **AI hallucinated** with large queries; Solved with post-processing validation
- **Developers wanted more control and deterministic outputs;** solved by query splitting

Key Takeaways

What if I start today?

STRATEGY

Identify your slowest stages and collaboration-heavy touchpoints.

Translate these bottlenecks into direct $ impact for the business.

IMPLEMENTATION

Choose the right AI approach for each specific use case.

RAG, prompt engineering, or another AI pattern

ADOPTION

Meet developers where they are to reduce friction.

Focus on intuitive workflows to drive long-term adoption and velocity.

Thank You!

Questions?