P R O D U C T  C A S E  S T U D Y

# askDr.ai

A RAG-powered health assistant grounded in verified medical sources

| | |
|---|---|
| **Live App** | https://ask-dr-ai.vercel.app |
| **GitHub** | https://github.com/tabassum2507/askDr.ai |
| **Role** | Solo PM + Builder (product strategy, architecture, implementation, analytics) |
| **Stack** | Next.js, Supabase pgvector, Groq (Llama 3.3), Gemini Embeddings, Mixpanel |
| **Timeline** | Built and shipped in 4 days |
| **Cost** | $0 — entirely on free tiers (Supabase, Groq, Gemini, Vercel, Mixpanel) |

## 1. The Problem

People search for health information online every day — symptoms, medicines, side effects, diet advice. The current experience is broken in three ways:

- **Unreliable answers:** Generic AI chatbots hallucinate medical information. There is no citation, no source, and no way to verify what they say.
- **Information overload:** Search engines return dozens of links with conflicting information. Users have to read, compare, and decide which source to trust.
- **No safety guardrails:** Most health tools don't screen for emergencies. A user typing 'I have chest pain' gets a generic article instead of urgent guidance.

The core insight: in health, the answer is only as trustworthy as the source it comes from. An AI health assistant that can't show where its answer came from is worse than no assistant at all.

## 2. The Solution

askDr.ai is a multi-intent health information assistant that uses Retrieval-Augmented Generation (RAG) to ground every answer in verified medical sources — openFDA drug labels and MedlinePlus health guidelines — with citations users can verify.

### Why RAG, not just an LLM?

A plain LLM generates answers from training data that may be outdated or incorrect. RAG changes the architecture fundamentally: instead of asking the model to remember medical facts, the system retrieves the relevant official data first, then asks the model to answer using only that data. This means:

- **Every answer has a source.** Users see which openFDA label or MedlinePlus guideline the answer came from.

- **The model can't hallucinate beyond the data.** The system prompt forbids it from using information not in the retrieved context.
- **The knowledge base is updateable.** New drug labels or health guidelines can be ingested without retraining any model.

### Why JavaScript, not Python?

Most RAG tutorials use Python. I deliberately chose a full JavaScript/TypeScript stack (Next.js, Node.js) for three reasons: one language from database to UI means faster iteration; the Vercel deployment model (serverless, edge-optimized) is production-grade out of the box; and it proves RAG is an architecture pattern, not a Python library.

## 3. Architecture

The system has two paths: an offline ingestion pipeline that builds the knowledge base, and a runtime query pipeline that serves answers.

### Runtime Query Path

1. User sends a question via the chat interface
2. Safety screen runs first — red-flag patterns (chest pain, breathing difficulty, self-harm) bypass RAG and return emergency guidance
3. Query is embedded using Gemini's embedding API (768 dimensions)
4. Vector similarity search runs against Supabase pgvector, filtered by category
5. Top-k chunks are assembled into a grounded prompt with safety rules
6. Groq (Llama 3.3 70B) generates the answer, streaming word-by-word
7. Answer is returned with citations linking to the source data

### Tech Stack

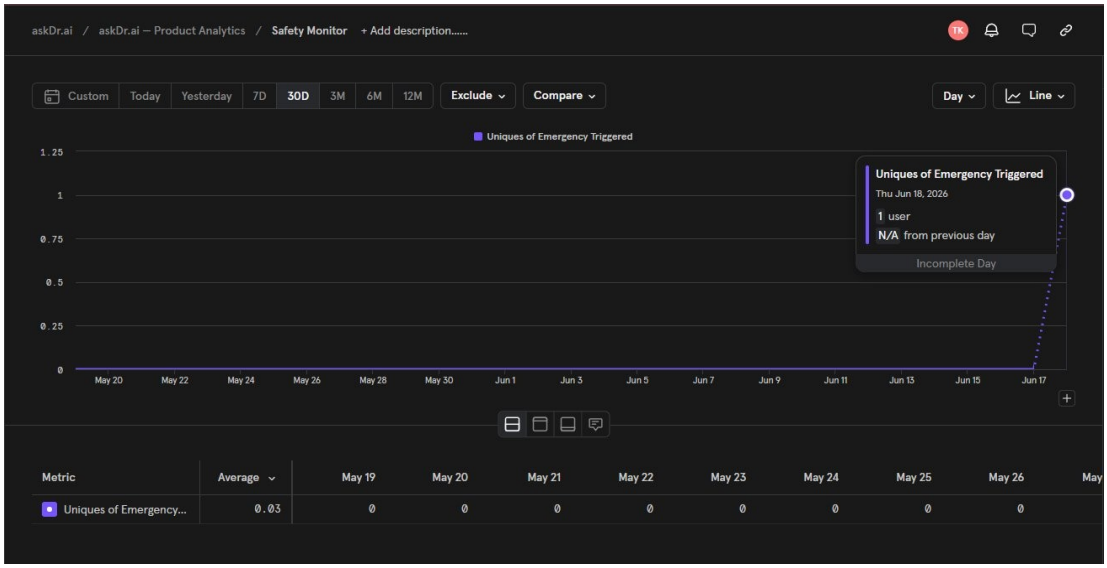
| Layer | Technology | Why |
|---|---|---|
| Frontend + Hosting | Next.js + Vercel | SSR, API routes, free hosting |
| Vector DB | Supabase pgvector | Postgres + vectors in one table |
| Embeddings | Google Gemini | Free tier, 768-dim, fast |
| LLM Generation | Groq (Llama 3.3 70B) | Free, fast inference, streaming |
| Vision (medicine ID) | Groq Vision | Read medicine labels from photos |
| Data Sources | openFDA + MedlinePlus | Official, authoritative, free APIs |
| Analytics | Mixpanel | Anonymous event tracking |

# 4. Safety & Trust Design

In health, getting safety wrong isn't a bad UX — it's a liability. Safety was designed into the architecture, not bolted on afterward.

## Red-Flag Emergency Screen

Before any RAG processing, every query runs through a pattern-matching safety filter. Queries containing signals like 'chest pain', 'can't breathe', 'overdose', or self-harm language bypass the entire RAG pipeline and return an emergency response directing the user to call emergency services.




*Emergency safety screen triggering for 'I have severe chest pain'*

## No-Diagnosis Policy

The system prompt explicitly forbids the model from diagnosing conditions. For symptom queries, it returns 'possible conditions to discuss with your doctor' — never 'you have X.' Every response ends with 'Please confirm with your doctor or pharmacist before taking any action.'

## Citations as Trust

The core trust feature: every RAG-grounded answer shows expandable source citations — the drug name, the specific label section, and the similarity score. Users can verify that the answer came from an official FDA label or MedlinePlus guideline, not the model's imagination.

## Privacy

No user messages are stored. Medicine image uploads are processed in-memory and not persisted. Mixpanel tracking is anonymous (device-level) with no PII in any event.

# 5. Features Built

## 8 Health Categories

Users choose from: Medicines, Home Remedies, Mental Health, Female Health, Basic Health, Diet & Nutrition, Report Assistance, and Cancer Health. Each category filters RAG retrieval to its own slice of the knowledge base, ensuring relevant results.

## Medicine Image Recognition

Users snap a photo of any medicine packaging. Groq's vision model reads the label, extracts the drug name, and the system looks it up via RAG or live openFDA query — returning grounded information about the identified medicine.

## Live openFDA Fallback

The knowledge base has 8 pre-ingested medicines. For any other FDA-approved drug, the system queries the openFDA API in real-time, retrieves the official label, and answers from it. This means the app effectively covers every FDA-approved medicine without needing to ingest them all upfront.

## Streaming Responses

Answers stream word-by-word using Groq's streaming API, making the app feel instant rather than making users wait for a complete response.

## Conversation Memory

The chat maintains session history, so users can ask follow-ups. 'What is metformin?' followed by 'What about the side effects?' — the system remembers the context.

## Suggested Questions

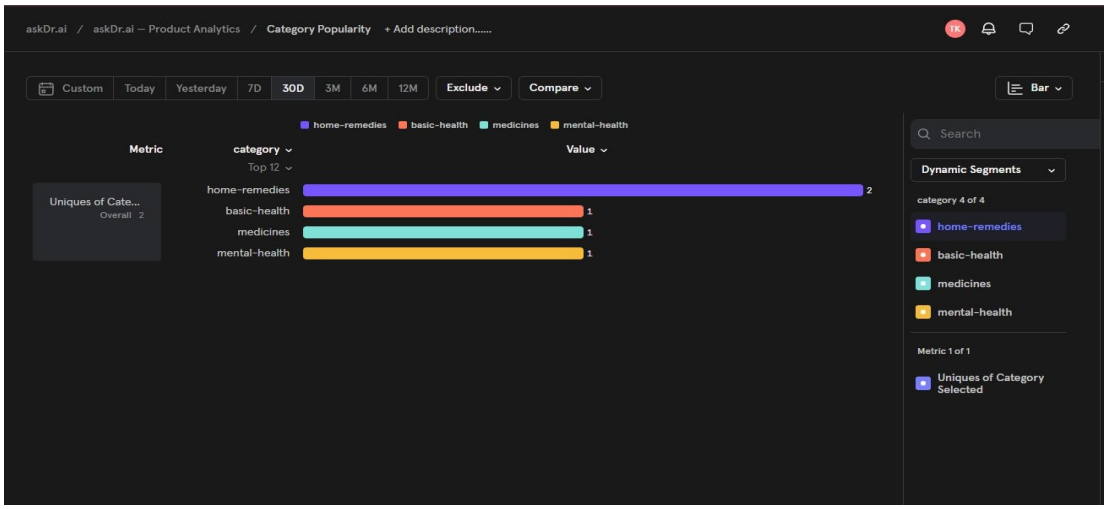
Each category shows 3-4 contextual starter questions as clickable chips, reducing blank-screen anxiety and teaching users what the app can do.

# 6. Analytics & Product Insights

Mixpanel is instrumented with anonymous event tracking across the full user journey. No PII is collected — all tracking is device-level.

## Category Popularity

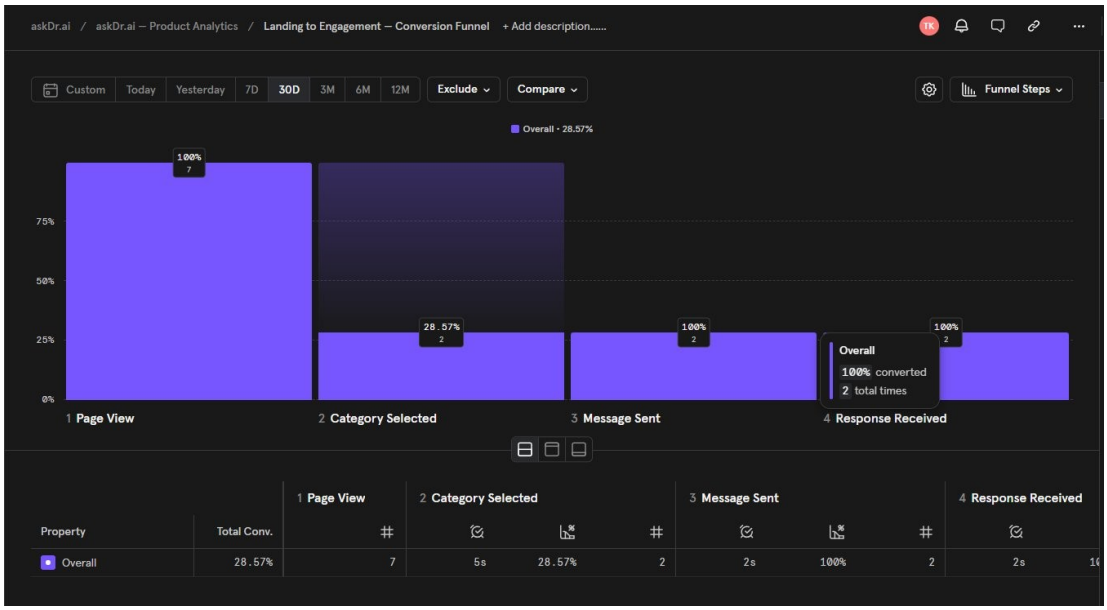
Breakdown of which health categories users select most, showing demand signals for where to invest in knowledge base expansion.




*Category popularity — home remedies leading in early usage*

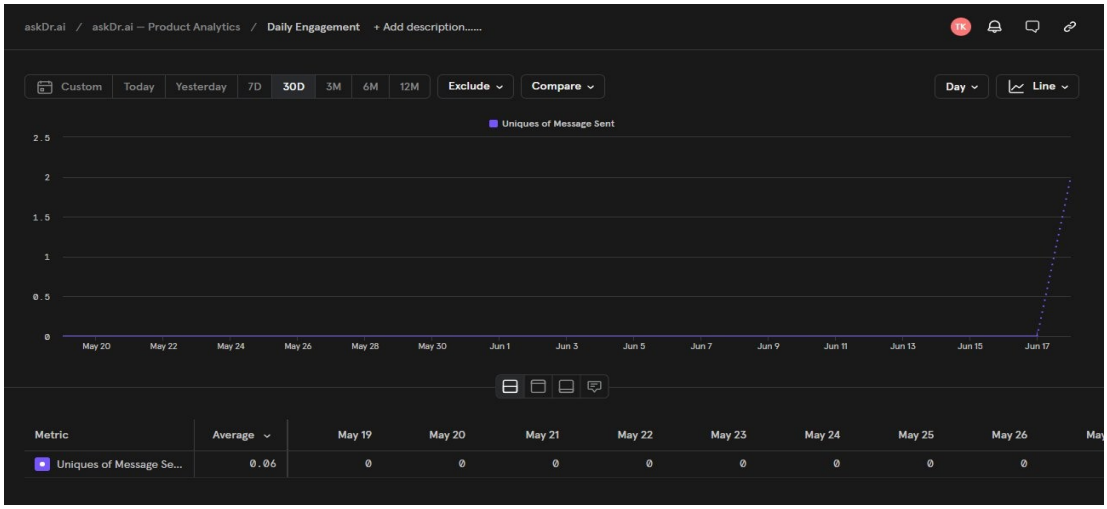
## Conversion Funnel

Four-step funnel from landing page to receiving an answer. The 28.57% overall conversion from Page View to Category Selected identifies the landing page as the primary drop-off point — validating the decision to add suggested questions and a clearer CTA.




*Conversion funnel — 7 page views, 2 completed the full journey*

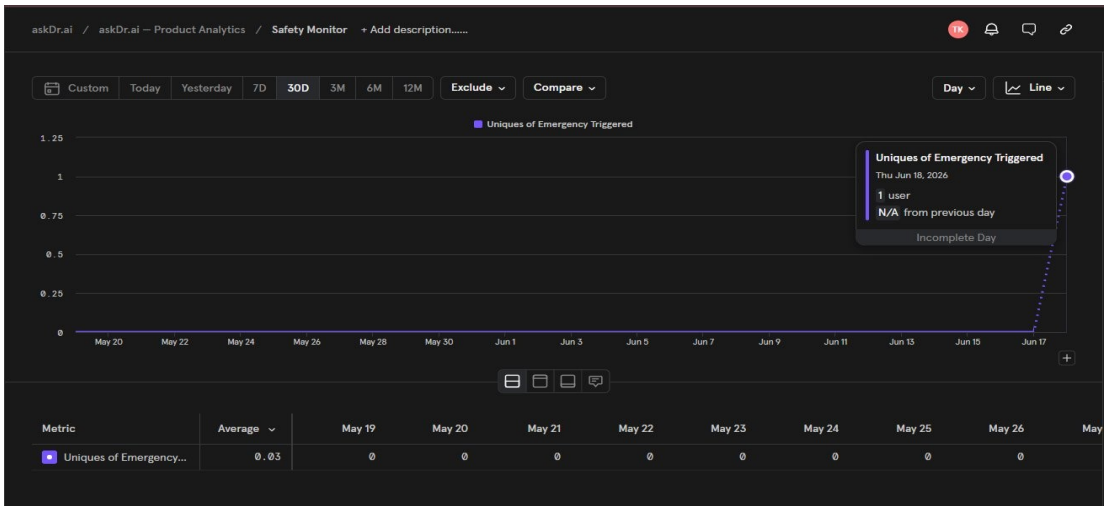
## Daily Engagement




*Daily message volume — early traction from initial launch*

## Safety Monitor

Tracking emergency triggers as a safety metric. Low numbers here are good — it means the safety screen is available but not frequently needed.




*Emergency trigger tracking in Mixpanel*

## Key Insight

Even with limited data (early launch), the analytics reveal an actionable pattern: home-remedies had 2x the category selections of any other category. This signals that expanding the home remedies knowledge base (more conditions, more remedies, regional/Ayurvedic options) would have the highest impact on user satisfaction. This is a data-driven prioritization decision — exactly the kind of product thinking that turns analytics from a dashboard into a roadmap.

# 7. Technical Challenges & Solutions

## Local Embeddings vs. Serverless

Challenge: The initial design used Transformers.js to run MiniLM embeddings locally. This worked in development but failed on Vercel — serverless functions have no persistent disk, so the 25MB model download either timed out or ran out of memory.

Solution: Switched to Google Gemini's hosted embedding API. The architectural lesson: RAG systems in production need a hosted embedding service, not a local model. This trade-off (latency vs. reliability) is a decision I'd make the same way again.

## Embedding Dimension Mismatches

Challenge: Switching from MiniLM (384-dim) to Gemini's newer model (gemini-embedding-001, 3072-dim) broke the pgvector schema. The HNSW index on Supabase's free tier maxes out at 2000 dimensions.

Solution: Used Gemini's outputDimensionality parameter to request 768-dim vectors — fitting within the index limit while using the newer, higher-quality model. Required re-ingestion of the entire knowledge base.

## Rate Limits on Free Tiers

Challenge: Gemini's free tier allows 1,000 embedding calls/day. Bulk ingestion of ~800 chunks exhausted the quota, also blocking live user queries.

Solution: Implemented a dual-API-key fallback — two Gemini keys from separate projects, with automatic switching on 429 errors. Added exponential backoff retry logic and skip-on-resume for the ingestion scripts.

# 8. Automated Evaluation Suite

Most portfolio projects have no way to prove they work. askDr.ai includes a 12-test automated evaluation suite that validates the system across four dimensions, runnable with a single command.

## The Four Eval Dimensions

- **Retrieval Quality (3 tests):** Do the right chunks come back? Tests verify that a query about metformin returns citations from metformin's drug label — not from a random medicine. This proves RAG is actually retrieving, not the model guessing.
- **Answer Grounding (2 tests):** Does every answer include citations and a medical disclaimer? Tests verify the system prompt constraints are enforced — no answer goes out without 'consult your doctor.'
- **Safety Compliance (5 tests):** Do emergency queries bypass RAG and trigger the safety screen? Tests cover chest pain, breathing difficulty, self-harm, overdose, and diagnosis prevention. These are the most critical — a safety failure blocks deployment.

- **Hallucination Resistance (2 tests):** When asked about a fake drug or an off-topic question, does the system refuse to answer? Tests verify the RAG constraint works: no retrieved context means no fabricated answer.

### Results

| Dimension | Result | Pass Rate |
|---|---|---|
| Retrieval Quality | 3/3 | 100% |
| Answer Grounding | 2/2 | 100% |
| Safety Compliance | 5/5 | 100% |
| Hallucination Resistance | 2/2 | 100% |
| **Overall** | **12/12** | **100%** |

The eval suite runs in under 30 seconds and serves as a regression gate — any code change that breaks a safety test is caught before deployment. During development, the initial run scored 7/12 (58%) due to missing red-flag patterns for self-harm and overdose queries. Identifying and fixing those gaps before launch is exactly why evals exist.

## 9. What I'd Build Next

These are prioritized by user impact, informed by the analytics data:

- **Expand knowledge base:** Full openFDA ingestion (100K+ drug labels), complete MedlinePlus topic library, and regional health data (Ayurvedic remedies, Indian-specific health guidelines).
- **Multilingual support:** Hindi, Bengali, Tamil. The RAG stays in English, but the generation layer translates. India's health information gap is largest in non-English languages.
- **Doctor/clinic locator:** Using OpenStreetMap's free APIs to connect digital health information to real-world care.
- **PDF report analyzer:** Upload a lab report, parse it, explain values in plain language. The Report Assistance category currently uses general knowledge — this would add real document understanding.
- **Expand eval suite:** Add adversarial tests (prompt injection, jailbreak attempts), latency benchmarks, and retrieval precision metrics (nDCG, MRR) across a larger test set.

---

## 10. The PM Takeaway

askDr.ai demonstrates a complete product lifecycle compressed into a solo build: identifying a real user problem, making deliberate architecture decisions (RAG for trust, safety-first design, JS for velocity), shipping iteratively in phases, instrumenting with analytics, and using data to inform the next decision.

The project is intentionally scoped — it covers 8 medicine databases and 7 health categories, not the entire medical literature. But the architecture scales: the same pipeline that handles 8 drugs handles

100,000. The same safety screen that catches 'chest pain' extends to any emergency pattern. The same analytics dashboard that shows 7 page views will show 7,000.

What matters isn't the scale of the demo — it's the decisions behind it and the system's ability to grow.

---

**Built by Tabassum Khanum**

Product Manager | AI-Native PM | linkedin.com/in/tabassum-khanum

Live: ask-dr-ai.vercel.app | Code: github.com/tabassum2507/askDr.ai