

Session 3: Tools and Technologies for Leading AI Initiatives

What AI leaders should know when guiding engineering teams and vendors

AI as an organizational capability: build, integrate, ground, orchestrate and evaluate

Leadership with AI – Rajendra M Sonar, IIT Bombay 

Platform-Led Substrates Accelerate AI Delivery by Eliminating Siloed Pitfalls

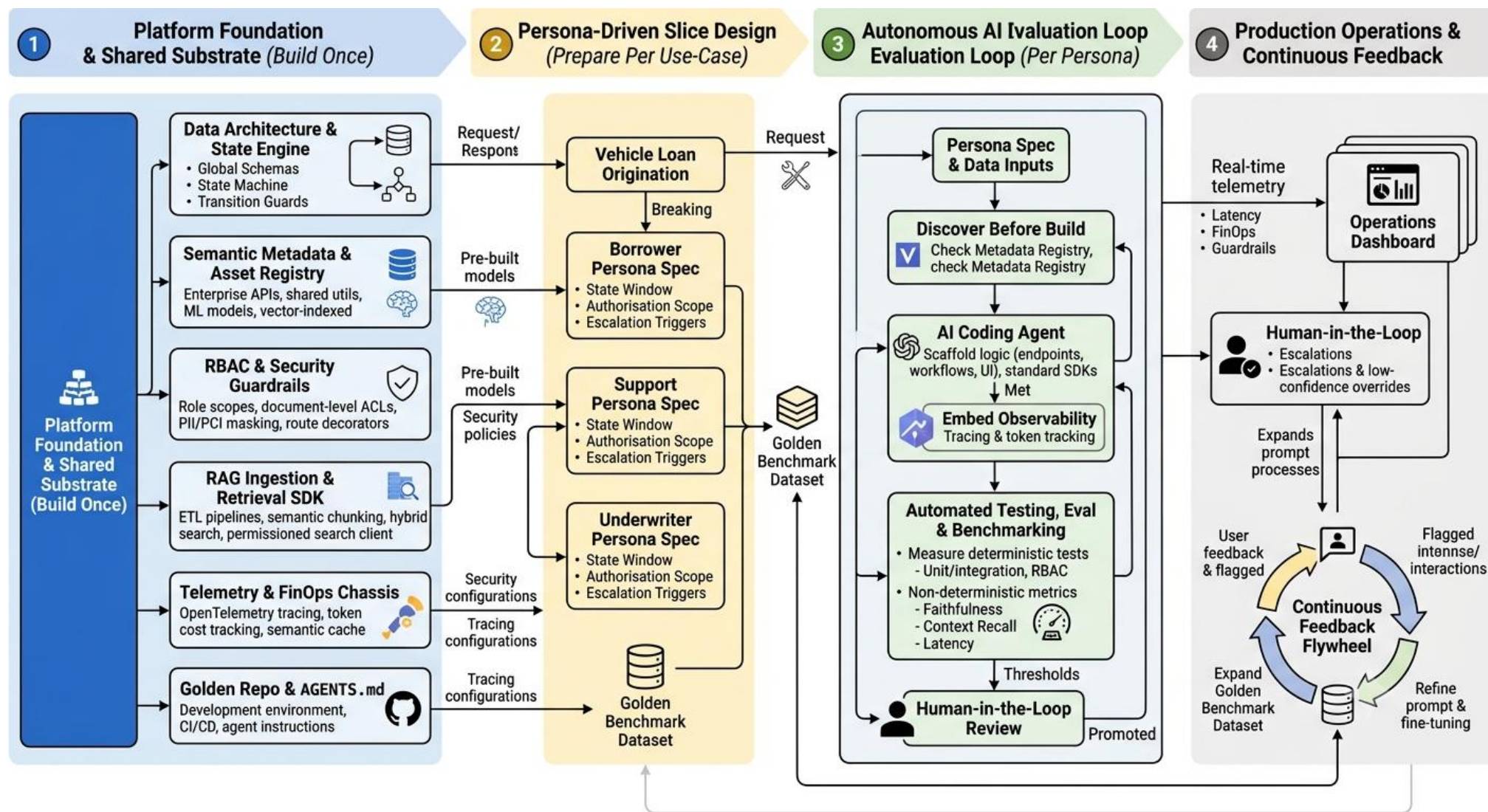
The Traditional Siloed Pitfall

- ✗ **Redundant Pipelines:** Custom-coding data connectors and pipelines separately for each new use-case.
- ✗ **Fragmented Schemas:** Inconsistent data structures causing sync failures and integration friction.
- ✗ **Duplicate Code:** Creating redundant third-party API and internal database calculators.
- ✗ **Observability Blindspots:** Operating without centralized distributed tracing, or real-time cost auditing.
- ✗ **Fragile Code Promotion:** Deploying agents directly to production without testing gates and baseline benchmarks.

The Platform-Led Standard

- ✓ **Unified Substrate:** Build-once core capabilities that act as a shared engine for all enterprise agents.
- ✓ **Strongly Typed Schemas:** Enforce centralized global entity definitions.
- ✓ **Semantic Asset Registry:** Index calculators and utilities in a vector registry to enforce tool reuse.
- ✓ **Embedded Guardrails:** Inject native I/O masking, prompt defense, and automated compliance.
- ✓ **Continuous Benchmarking:** Force agent code through automated non-deterministic evaluations.

Example Enterprise AI System Architecture & Delivery Lifecycle



Recap: Example: User prompts and AI working behind the scene

- 1 A user (e.g. employee) is looking for something something? E.g. *tell me what earned leave rules?*
- 2 This information is not available in public domain and very specific to my organization.
- 3 Where and how to **get** the relevant knowledge based on what the user is looking for? Which knowledge source e.g. document(s), which part(s) of document etc.
- 4 Ohk! I **got** the relevant text from HR leave document! What to do next? How can I put that in the context of what user is looking for and **generate** answer?
- 5 Response is given to the user *"you can avail 30 days of earned leave in a year, you can carry over unused earned leave next year, maximum you can accumulate 300 days overall ... "*
- 6 User now asking *"what about my earned leave status"*?
- 7 From where I can get data the user is asking for? What I need to do that? I checked user is logged in.
- 8 This information is available in my ERP system. Let me find out **which API/DB Code** would return me requested information. I got an API: *Get Leave Balance* to make a call.
- 9 API called, got response: 40. Let me create response to the user: *"you have 40 days earned leave in your leave account"*
- 10 Now user is asking *"I want to avail 10 days leave from 10.9.2026"*. Let me check whether such operations are allowed or not.
- 11 **Got an API/WF which does this operation.** Let me confirm with user whether the user wants to proceed. *Do you want to go ahead?* user says *yes!*
- 12 API is executed, it sent response." *Your 10 days leave from 10.9.2026 has been sent for approval to SK Raman Any thing else ... I can help you? "*
- 13 User responds: *"No thanks"*

Recap: Example: User prompts and AI working behind the scene

- 1 A user (e.g. employee) is looking for something something? E.g. *tell me what earned leave rules?*
- 2 This information is not available in public domain and very specific to my organization.
- 3 Where and how to **get** the relevant knowledge based on what the user is looking for? Which knowledge source e.g. document(s), which part(s) of document etc.
- 4 Ohk! I **got** the relevant text from HR leave document! What to do next? How can I put that in the context of what user is looking for and **generate** answer?
- 5 Response is given to the user *"you can avail 30 days of earned leave in a year, you can carry over unused earned leave next year, maximum you can accumulate 300 days overall ... "*
- 6 User now asking *"what about my earned leave status"*?
- 7 From where I can get data the user is asking for? What I need to do that? I checked user is logged in.
- 8 This information is available in my ERP system. Let me find out **which API/DB Code** would return me requested information. I got an API: *Get Leave Balance* to make a call.
- 9 API called, got response: 40. Let me create response to the user: *"you have 40 days earned leave in your leave account"*
- 10 Now user is asking *"I want to avail 10 days leave from 10.9.2026"*. Let me check whether such operations are allowed or not.
- 11 **Got an API/WF which does this operation.** Let me confirm with user whether the user wants to proceed. *Do you want to go ahead?* user says *yes!*
- 12 API is executed, it sent response." *Your 10 days leave from 10.9.2026 has been sent for approval to SK Raman Any thing else ... I can help you? "*
- 13 User responds: *"No thanks"*

Example: User prompts and AI working behind the scene

Codex Prompt:

Go through the screenshot depicting the steps in use-case scenario. This example pretty good involves technologies like RAG, APIs, SQL, databases, MCP, Agents, etc.

As part of my class on AI tools, frameworks, languages etc. in my leadership with AI program for practitioners. I am going to demonstrate how such use-cases can be implemented using various tools from scratch and explain the tech stack behind the scenes.

- 1> Consider HR document https://www.iitb.ac.in/sites/www.iitb.ac.in/files/Faculty_Handbook_2021.pdf
- 2> Create a database schema based on the document (e.g. mysql) with sample data
- 3> Create API layer to access the database (mostly CRUD kind of operations), create sample APIs for leave management
- 4> Create sample leave management app (using simple HTML/CSS, middle layer any popular programming language) can be based on APIs, one general API which executes SQL directly to demonstrate NLP to SQL conversion
- 5> Move this document into vector DB and use RAG to answer employee general queries using a chatbot.
- 6> Use the chatbot to answer queries requiring user context, invoke MCP to call appropriate API and return and format answers
- 7> Develop agents which allows employee to apply for leave etc.

Codex Session

Building an HR AI Copilot

From policy documents and SQL data to RAG, tools, MCP, agents, guardrails and evaluation

DEMO

One employee
conversation.
Three data
paths.
Ten technology
layers.

One conversation crosses three fundamentally different systems

01

Policy knowledge

“Tell me about earned leave”

Retrieve grounded guidance from the faculty handbook.

02

Personal status

“What is my balance?”

Use authenticated identity to query operational data.

03

Business action

“Apply from 10–19 Sep”

Validate, preview, confirm, transact and audit.

Choose the data path before generating the answer

POLICY

Document → chunks → embeddings → retrieval → cited answer

Narrative rules change slowly; provenance matters.

PERSONAL

Identity → authorization → API/tool → SQL → formatted answer

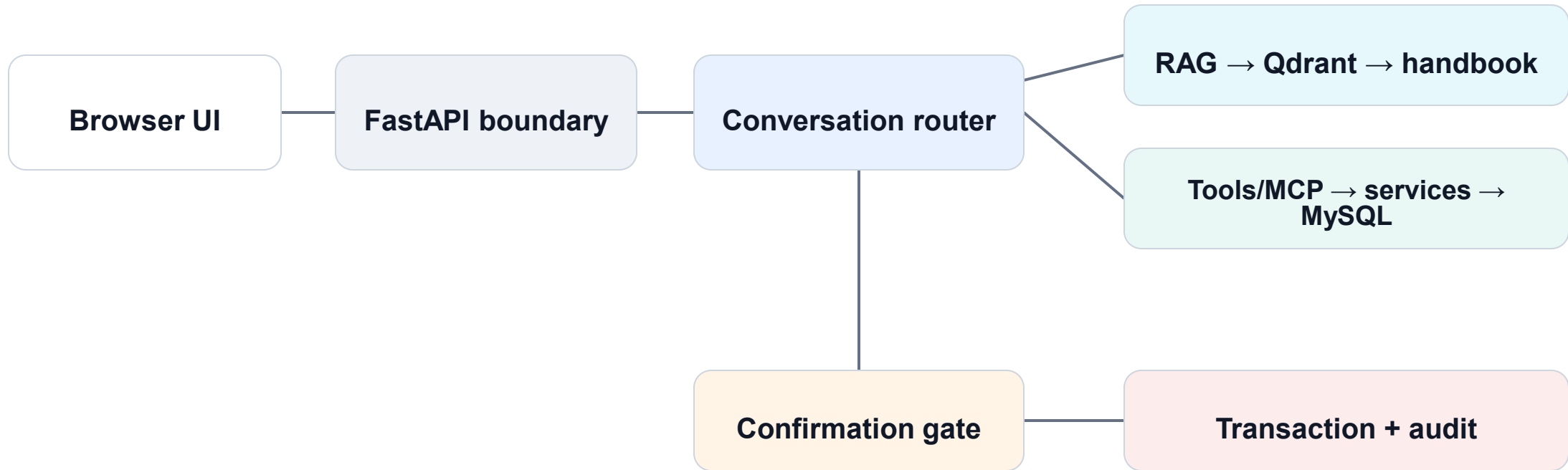
Balances are live, private and user-specific.

ACTION

Intent → validation → preview → confirmation → transaction + audit

Mutations require deterministic controls and consent.

The model orchestrates; deterministic services retain control



Same domain rules under REST and MCP

A prompt is not identity: context must be assembled deliberately

USER MESSAGE

“What is my earned leave balance?”

+

Authenticated identity

Authorization scope

Conversation state

Relevant policy

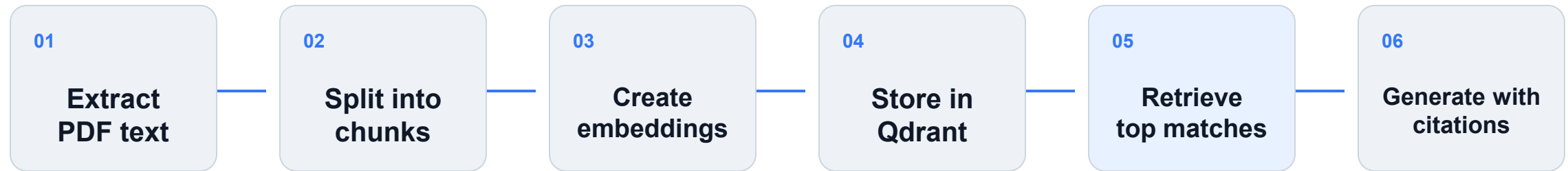
Permitted tools

=

SAFE
EXECUTION
CONTEXT

Signed out → generic policy only | Signed in → personal tools enabled

RAG grounds policy answers in the handbook—not model memory



Grounding contract Answer only from retrieved passages · cite page numbers · express uncertainty · remind users that the 2021 handbook is guidance

NLP-to-SQL needs a safety boundary after generation

1

Question

Show leave balances
for all employees

2

Generate

Structured SQL output
from known schema

3

Validate

SELECT only · one
statement
allowlisted tables

4

Execute

Read-only account
≤ 100 rows

DELETE · DROP · UPDATE · INSERT · multiple statements · unknown tables → BLOCK

APIs, MCP and agents solve different integration problems

API

**A transport contract
for applications**

MCP

**A discovery + invocation
contract for AI hosts**

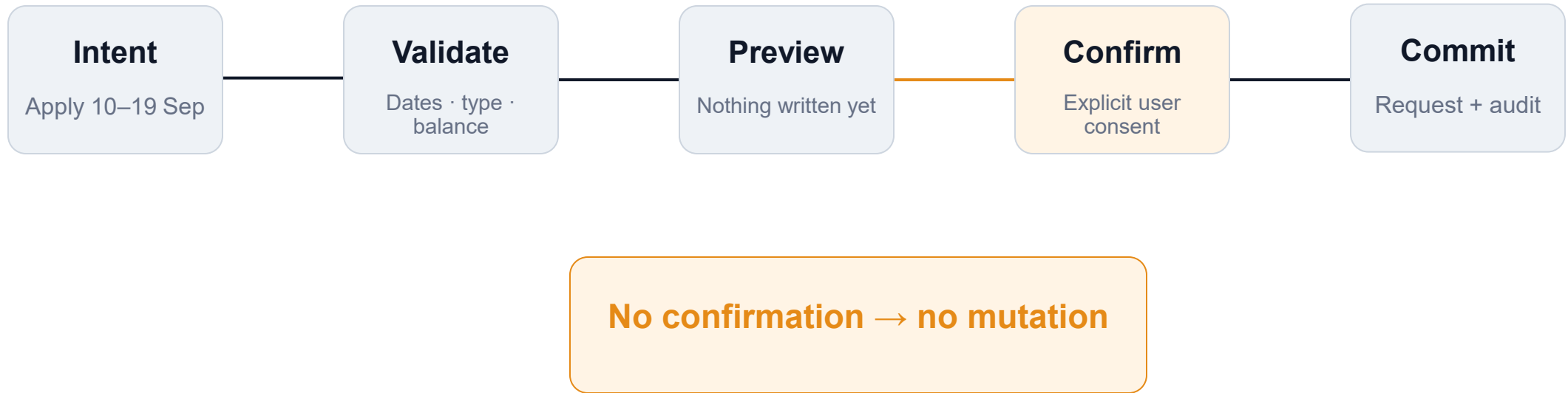
Agent

**An orchestrator for
intent, tools and state**

All three call the same deterministic leave services

Business rules do not belong inside prompts or protocol adapters.

State-changing actions pause at a human confirmation gate



Guardrails are hard gates across identity, data and action

- Anonymous users cannot read personal data
- Employee ID comes from trusted session context
- Retrieved documents are treated as data, not instructions
- Generated SQL is constrained and independently validated
- Leave requests require confirmation and create audit events

Block first. Explain clearly. Audit always.

Feedback discovers failures; curated evals prevent regressions



Guardrails

Pass/fail: auth, confirmation, SQL mutation, data isolation

Quality evaluations

Retrieval relevance, grounded claims, routing accuracy, task completion

A thumbs-up is a signal—not ground truth.

Now trace the same story through the live system

- 1 Signed out: personal request is blocked
- 2 Anonymous: policy question uses RAG
- 3 Asha vs Ravi: same prompt, different context
- 4 Apply: preview before confirmation
- 5 Confirm: request + audit record
- 6 Instructor: schema, NLP-to-SQL, guardrails, evals

TECHNOLOGY MAP

Fast API

MySQL / SQLite

Qdrant / local vectors

OpenAI Responses API

MCP tools

HTML · CSS ·

JavaScript

CI · Docker · tests

Goal: understand the boundaries—not memorize the libraries.