

Study Delight Publications

From
Ideas
to
Real-World
Solutions

AI ENGINEERING & PROJECTS

Simplified

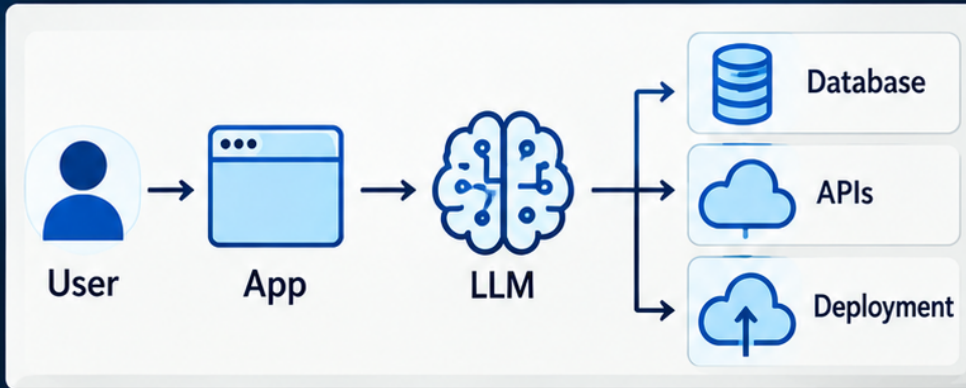
Learn
Apply
Build
Scale

PART 5

Build • Deploy • Scale • Real-World Applications


Build
Applications


Cloud
& APIs




MLOps
& Deployment


Real-World
Projects


End-to-End
Guidance


Code Examples


Deployment
Made Simple


Best Practices


Career Ready

pythonforaiml.com

PART 5 • AI ENGINEERING & PROJECTS

Turn models and LLMs into reliable, testable, observable AI products.

30-PAGE ROADMAP



01 AI Engineering: Model ↔ Product

16 Data & Model Drift

02 Notebook ↔ Production

17 Logging & Observability

03 AI System Architecture

18 Testing AI Systems

04 Data Pipelines

19 Prompt & Version Control

05 Data Validation

20 RAG Architecture

06 Feature Engineering

21 Vector Search

07 Experiment Tracking

22 RAG Evaluation

08 Model Packaging

23 LLM Evaluation

09 APIs with FastAPI

24 Guardrails

10 Inference Pipeline

25 Prompt Injection

11 Docker Basics

26 Security & Privacy

12 Model Serving

27 Cost Optimization

13 Batch vs Real-Time Inference

28 Caching

14 Latency & Throughput

29 CI/CD for AI

15 Monitoring

30 End-to-End AI Project & Final Map

LEARN • PRACTICE • BUILD • GROW

Use this index as your quick navigation map while studying.

From model ✖ reliable AI product

THEORY

AI engineering is the discipline of turning an experiment into a dependable system. The model is only one component. A real product also needs data pipelines, APIs, testing, monitoring, security, evaluation and deployment.



KEY IDEA

Think like an engineer: accuracy is important, but reliability, latency, cost and safety decide whether users can trust the system.

ENGINEER'S NOTE

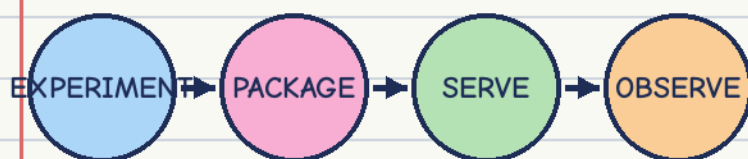
Make failures visible and actionable.

REMEMBER ✖ build systems that can be measured, tested and improved.

Notebook → production

THEORY

A notebook proves that an idea can work. Production engineering proves that it keeps working when users, traffic, new data and failures arrive.



WHY IT MATTERS

Typical path: explore → train → evaluate → package → serve → observe → improve. Every arrow introduces engineering decisions.

KEY IDEA

Typical path: explore → train → evaluate → package → serve → observe → improve. Every arrow introduces engineering decisions.

ENGINEER'S NOTE

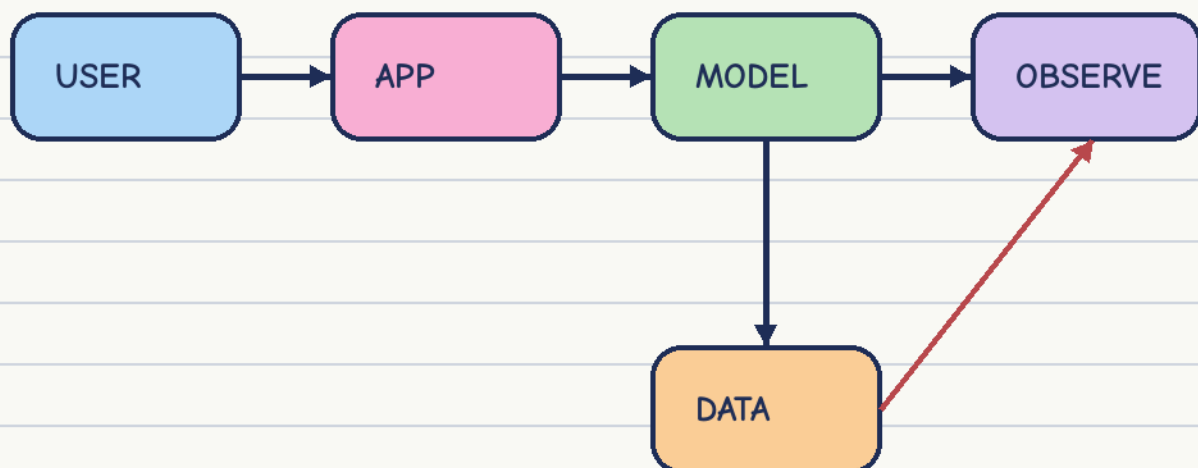
Keep trusted instructions separate from external data.

REMEMBER → build systems that can be measured, tested and improved.

See the whole AI system

THEORY

A production AI system usually has a user-facing layer, application logic, model or LLM layer, data stores and an observability layer. Separating these responsibilities makes systems easier to test and change.



boundaries = easier testing + safer change

KEY IDEA

Key idea: design the boundaries before writing a large amount of code.

ENGINEER'S NOTE

Reproducibility is a feature, not paperwork.

REMEMBER ☒ build systems that can be measured, tested and improved.

Garbage in ☒ unreliable predictions

THEORY

A data pipeline moves information from sources to a form that a model can safely consume. Steps often include ingestion, cleaning, validation, transformation, storage and feature creation.



WHY IT MATTERS

A pipeline should be repeatable. If training data cannot be reproduced, debugging a model becomes guesswork.

KEY IDEA

A pipeline should be repeatable. If training data cannot be reproduced, debugging a model becomes guesswork.

ENGINEER'S NOTE

Use small tests continuously instead of one giant test at the end.

REMEMBER ☒ build systems that can be measured, tested and improved.

Trust the input before trusting the model

THEORY

Validation checks whether incoming data has the expected schema, ranges, types and quality. It can catch missing columns, impossible values, duplicated records and unexpected categories before they reach inference.



KEY IDEA

Example: if a model expects age between 0 and 120, an age of 700 should be rejected or investigated—not silently accepted.

ENGINEER'S NOTE

A simple baseline gives every later improvement a reference point.

REMEMBER ☒ build systems that can be measured, tested and improved.

Turn raw signals into useful evidence

THEORY

Features are the measurable signals a model uses. Good feature engineering captures information that is relevant, stable and available at prediction time.



WHY IT MATTERS

Watch for leakage: a feature must not contain information that would only be known after the prediction event.

KEY IDEA

Watch for leakage: a feature must not contain information that would only be known after the prediction event.

ENGINEER'S NOTE

Measure it before optimizing it.

REMEMBER ☒ build systems that can be measured, tested and improved.

Remember what actually worked

THEORY

Machine learning involves many trials: data versions, features, model families and hyperparameters. Experiment tracking records these choices together with metrics and artifacts.



KEY IDEA

Without tracking, teams often reproduce the worst experiment: the one they remember incorrectly.

ENGINEER'S NOTE

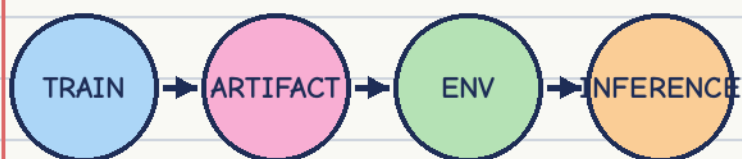
Make failures visible and actionable.

REMEMBER ☒ build systems that can be measured, tested and improved.

Make the model portable

THEORY

A trained model needs its learned parameters plus the preprocessing assumptions required at inference. Packaging should capture versions and configuration so the same artifact behaves consistently across environments.



WHY IT MATTERS

The goal is not simply 'save the model'. The goal is 'save everything needed to use the model correctly'.

KEY IDEA

The goal is not simply 'save the model'. The goal is 'save everything needed to use the model correctly'.

ENGINEER'S NOTE

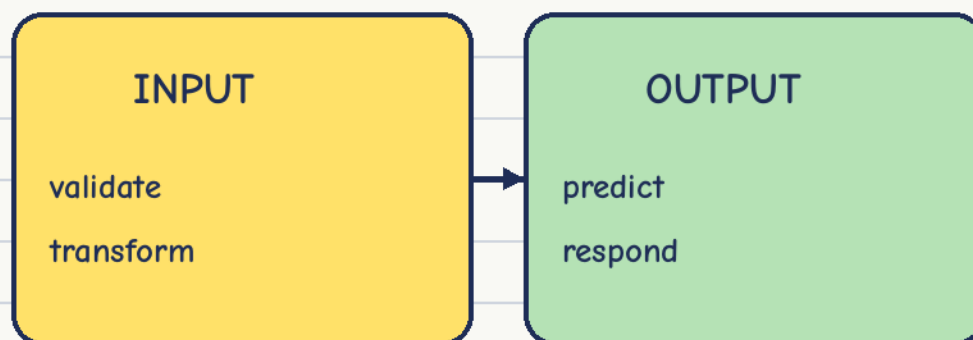
Keep trusted instructions separate from external data.

REMEMBER ☒ build systems that can be measured, tested and improved.

Give the model a doorway

THEORY

An API exposes model functionality through a predictable interface. A request contains input data; the service validates it, runs preprocessing and inference, then returns a structured response.



KEY IDEA

Keep API contracts explicit: required fields, types, error behavior and response format.

ENGINEER'S NOTE

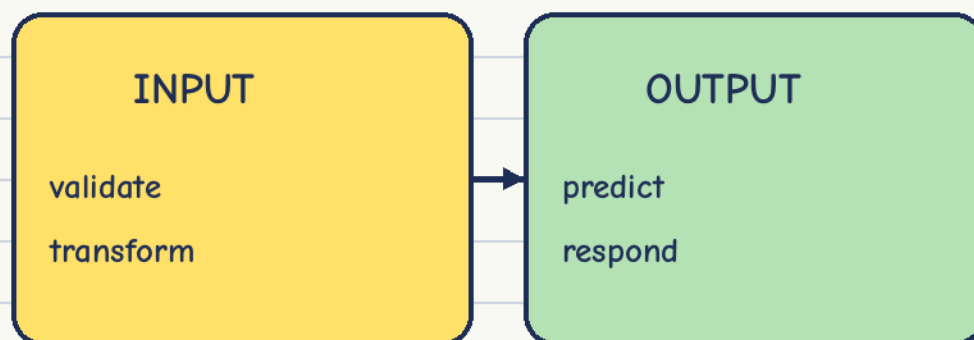
Reproducibility is a feature, not paperwork.

REMEMBER ☒ build systems that can be measured, tested and improved.

Prediction is a sequence, not a button

THEORY

Inference usually means: receive → validate → preprocess → predict → postprocess → respond. The same transformations used during training must be reproduced correctly.



KEY IDEA

A common production bug is training with one preprocessing rule and serving with another.

ENGINEER'S NOTE

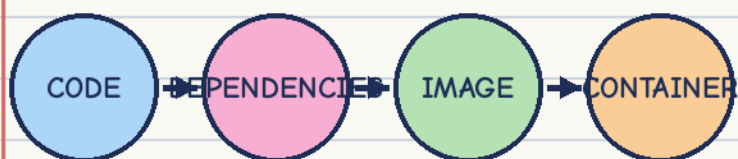
Use small tests continuously instead of one giant test at the end.

REMEMBER → build systems that can be measured, tested and improved.

Same environment, fewer surprises

THEORY

Containers package an application with its runtime dependencies. For AI systems this helps keep library versions, system packages and startup commands consistent between development and deployment.



KEY IDEA

A useful mental model: the container is a reproducible box around the service—not a replacement for good architecture.

ENGINEER'S NOTE

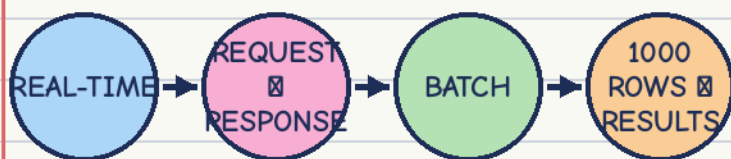
A simple baseline gives every later improvement a reference point.

REMEMBER ☒ build systems that can be measured, tested and improved.

Batch or real-time?

THEORY

Real-time inference responds to an individual request and is useful when a user is waiting. Batch inference processes many records together and is often cheaper for scheduled workloads.



WHY IT MATTERS

Choose based on the business requirement, not because one pattern sounds more advanced.

KEY IDEA

Choose based on the business requirement, not because one pattern sounds more advanced.

ENGINEER'S NOTE

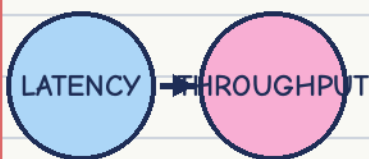
Measure it before optimizing it.

REMEMBER ☒ build systems that can be measured, tested and improved.

Fast is not the same as scalable

THEORY

Latency is how long one request takes. Throughput is how much work the system handles over time. A system can have low latency but poor throughput, or high throughput with slower individual responses.



KEY IDEA

Measure p50 and p95 latency, not just the average. Tail latency often determines the user experience.

ENGINEER'S NOTE

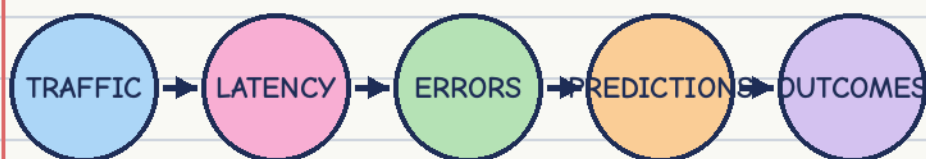
Make failures visible and actionable.

REMEMBER ☒ build systems that can be measured, tested and improved.

The model can fail quietly

THEORY

Monitoring watches the system after deployment. Useful signals include request volume, latency, error rate, prediction distributions, data quality and business outcomes.



WHY IT MATTERS

A dashboard without useful alerts is just decoration. Define thresholds and actions before production traffic arrives.

KEY IDEA

A dashboard without useful alerts is just decoration. Define thresholds and actions before production traffic arrives.

ENGINEER'S NOTE

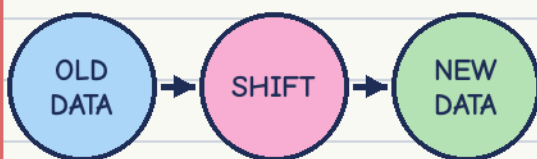
Keep trusted instructions separate from external data.

REMEMBER ☒ build systems that can be measured, tested and improved.

The world changes

THEORY

Data drift means the input distribution changes. Concept drift means the relationship between inputs and the target changes. Either can reduce model performance over time.



KEY IDEA

Example: customer behavior can change after a pricing policy changes. A model trained before the policy may become stale.

ENGINEER'S NOTE

Reproducibility is a feature, not paperwork.

REMEMBER ☒ build systems that can be measured, tested and improved.

Leave a trail for debugging

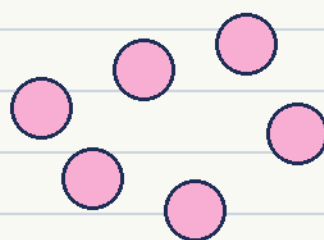
THEORY

Logs record important events such as request IDs, model version, latency, validation failures and controlled error information. Good logs make incidents explainable.

OLD



NEW



distribution changed

KEY IDEA

Never log secrets or unnecessary personal data. Observability must be designed with privacy in mind.

ENGINEER'S NOTE

Use small tests continuously instead of one giant test at the end.

REMEMBER ☒ build systems that can be measured, tested and improved.

Test more than the code

THEORY

Traditional unit tests remain useful, but AI systems also need data tests, model behavior tests and evaluation tests. A service can return HTTP 200 while producing poor predictions.



KEY IDEA

Think in layers: code correctness + data quality + model quality + system behavior.

ENGINEER'S NOTE

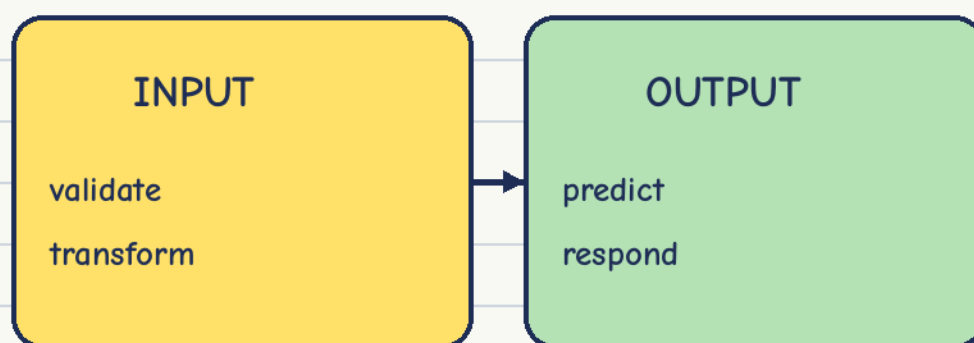
A simple baseline gives every later improvement a reference point.

REMEMBER ☒ build systems that can be measured, tested and improved.

Prompts are part of the system

THEORY

For LLM applications, prompts, system instructions, retrieval settings and model versions can change behavior. Treat them as versioned artifacts rather than invisible text copied between notebooks.



KEY IDEA

Record which prompt and model produced an output so regressions can be investigated.

ENGINEER'S NOTE

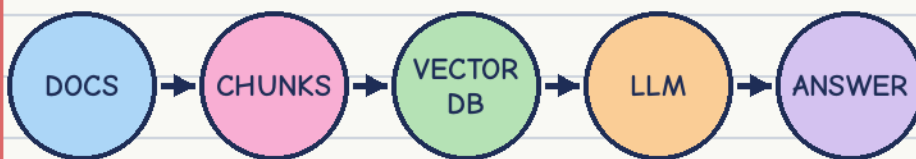
Measure it before optimizing it.

REMEMBER ☒ build systems that can be measured, tested and improved.

Give an LLM useful context

THEORY

Retrieval-Augmented Generation combines search with generation. Documents are chunked, embedded and indexed. At query time, relevant chunks are retrieved and placed into the model context.



KEY IDEA

RAG reduces dependence on model memory for private or changing information, but retrieval quality becomes a new engineering problem.

ENGINEER'S NOTE

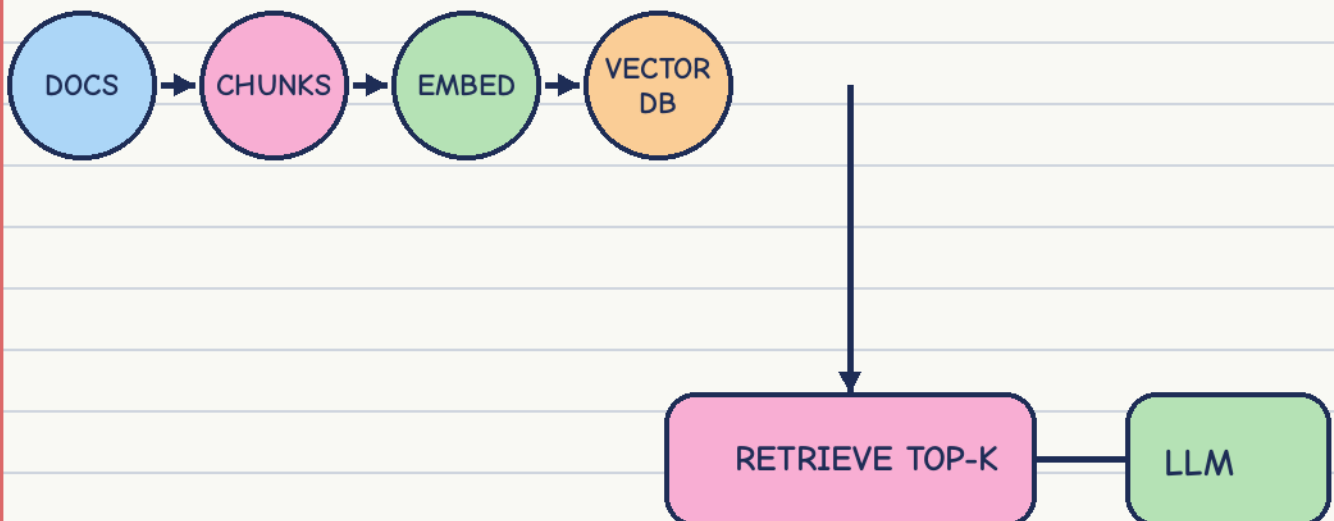
Make failures visible and actionable.

REMEMBER ☒ build systems that can be measured, tested and improved.

Similarity is the bridge

THEORY

Embeddings represent text as vectors. Similar meanings tend to be closer in vector space. A vector database can retrieve candidate passages using similarity search.



retrieval supplies evidence; generation turns evidence into an answer

KEY IDEA

Metadata such as source, date, access level and document ID is essential for filtering and traceability.

ENGINEER'S NOTE

Keep trusted instructions separate from external data.

REMEMBER ☒ build systems that can be measured, tested and improved.

Did retrieval actually help?

THEORY

A RAG system has at least two questions: did we retrieve useful evidence, and did the generator use it correctly? Evaluate retrieval and answer quality separately.



KEY IDEA

Useful dimensions include relevance, faithfulness, completeness and citation correctness.

ENGINEER'S NOTE

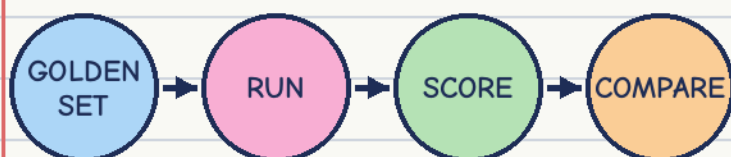
Reproducibility is a feature, not paperwork.

REMEMBER ☒ build systems that can be measured, tested and improved.

One score is not enough

THEORY

Generative systems can produce multiple acceptable answers. Evaluation therefore combines automated metrics, structured rubrics and human review where necessary.



WHY IT MATTERS

Create a small golden dataset of representative questions. Compare versions against the same set instead of relying on random examples.

KEY IDEA

Create a small golden dataset of representative questions. Compare versions against the same set instead of relying on random examples.

ENGINEER'S NOTE

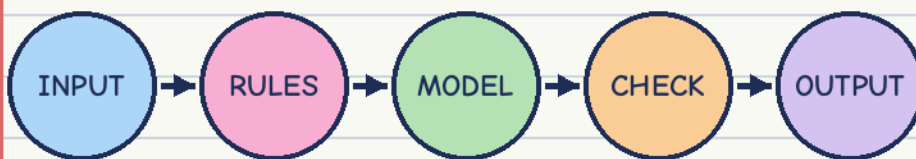
Use small tests continuously instead of one giant test at the end.

REMEMBER ☒ build systems that can be measured, tested and improved.

Useful freedom, bounded behavior

THEORY

Guardrails constrain unsafe, invalid or out-of-scope behavior. They can operate before generation, during tool use or after generation.



KEY IDEA

Examples include input validation, allowed-tool lists, output schemas, content checks and refusal policies.

ENGINEER'S NOTE

A simple baseline gives every later improvement a reference point.

REMEMBER ☒ build systems that can be measured, tested and improved.

Treat external text as untrusted

THEORY

Prompt injection attempts to manipulate an AI system through instructions embedded in user input or retrieved content. The core defense is to separate trusted instructions from untrusted data and restrict tool permissions.



WHY IT MATTERS

Never assume retrieved text is authoritative just because the retriever returned it.

KEY IDEA

Never assume retrieved text is authoritative just because the retriever returned it.

ENGINEER'S NOTE

Measure it before optimizing it.

REMEMBER ☒ build systems that can be measured, tested and improved.

AI systems handle valuable data

THEORY

Security covers authentication, authorization, secrets, dependency management and network controls. Privacy adds data minimization, retention rules and careful handling of personal information.



KEY IDEA

Principle: give each component only the permissions it needs. Reduce blast radius before an incident happens.

ENGINEER'S NOTE

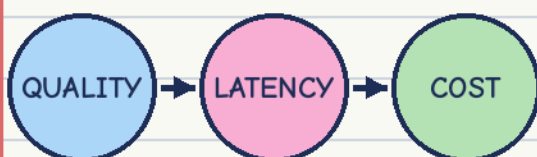
Make failures visible and actionable.

REMEMBER ☒ build systems that can be measured, tested and improved.

Accuracy has a budget

THEORY

AI systems consume compute, storage and model-token resources. Cost optimization can come from smaller models, caching, batching, better retrieval, shorter prompts and avoiding unnecessary calls.



WHY IT MATTERS

Optimize the whole workflow. A slightly cheaper model can still be expensive if poor retrieval causes repeated retries.

KEY IDEA

Optimize the whole workflow. A slightly cheaper model can still be expensive if poor retrieval causes repeated retries.

ENGINEER'S NOTE

Keep trusted instructions separate from external data.

REMEMBER ☒ build systems that can be measured, tested and improved.

Do not pay twice for the same work

THEORY

Caching stores reusable results. In AI applications, useful cache targets include embeddings, retrieval results and deterministic responses.



optimize the whole workflow

KEY IDEA

Cache keys must include the inputs that affect correctness, such as model version or prompt version. Otherwise stale results can appear correct.

ENGINEER'S NOTE

Reproducibility is a feature, not paperwork.

REMEMBER ☒ build systems that can be measured, tested and improved.

Ship changes safely

THEORY

Continuous integration checks changes automatically. Continuous delivery/deployment moves validated artifacts toward production. AI pipelines add data and evaluation gates to the usual software workflow.



WHY IT MATTERS

A practical gate: tests pass + evaluation does not regress + security checks pass + artifact is reproducible.

KEY IDEA

A practical gate: tests pass + evaluation does not regress + security checks pass + artifact is reproducible.

ENGINEER'S NOTE

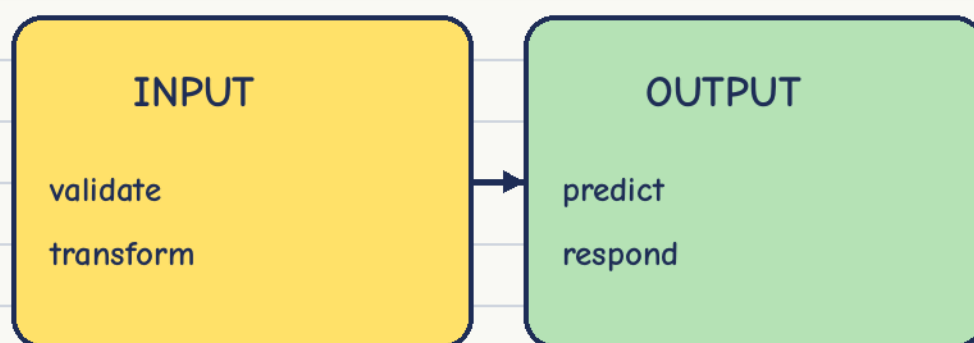
Use small tests continuously instead of one giant test at the end.

REMEMBER ☒ build systems that can be measured, tested and improved.

Build a production-minded AI app

THEORY

A complete project connects the pieces: data $\boxtimes$ validation $\boxtimes$ model/LLM $\boxtimes$ API $\boxtimes$ deployment $\boxtimes$ monitoring $\boxtimes$ evaluation $\boxtimes$ iteration.



KEY IDEA

Project blueprint: define the problem, create a measurable baseline, package the system, add tests and observability, then improve based on evidence.

ENGINEER'S NOTE

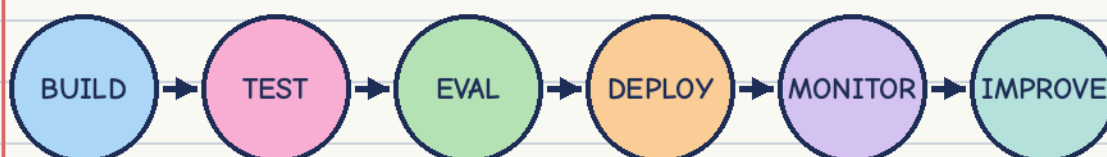
A simple baseline gives every later improvement a reference point.

REMEMBER $\boxtimes$ build systems that can be measured, tested and improved.

Your AI engineering checklist

THEORY

A strong AI engineer thinks across the entire lifecycle. Ask: Is the data valid? Is the model evaluated? Is the artifact reproducible? Is the API predictable? Is production observable? Is the system secure and cost-aware?



WHY IT MATTERS

Master loop: BUILD ✕ TEST ✕ EVALUATE ✕ DEPLOY ✕ MONITOR ✕ LEARN ✕ IMPROVE.

KEY IDEA

Master loop: BUILD ✕ TEST ✕ EVALUATE ✕ DEPLOY ✕ MONITOR ✕ LEARN ✕ IMPROVE.

ENGINEER'S NOTE

Measure it before optimizing it.

REMEMBER ✕ build systems that can be measured, tested and improved.