Solution Design

Databricks Certified Generative AI Engineer Associate

KNOWLEDGE CHEATSHEET

CHOOSE THE APPLICATION PATTERN

- **Single call** — Transform
- **RAG** — Ground answers
- **Chain** — Defined steps
- **Agent** — Choose tools

01 Solution design and model selection

Start with the business task

- **Task contract:** define the input, desired output, permitted actions and success measure before choosing a model. Distinguish extraction, classification, summarization, generation and question answering.
- **Support automation** can retrieve similar resolved tickets, summarize relevant interactions and generate a personalized reply. Preserve root cause, resolution and conversation text rather than unrelated billing statistics.
- **Data fit:** use manuals and policies for document Q&A, structured tables for exact records and aggregates, and live APIs for changing external facts.
- **Architecture choice:** a single call suits a bounded transformation; a chain coordinates known steps; an agent chooses tools dynamically. Evaluate the simplest design that satisfies the task.

Select models by workload

- **Model cards** describe task, training data, languages, license and limitations. Benchmark candidates on representative domain examples; popularity and release recency do not establish suitability.
- **Generator versus encoder:** a generative language model writes responses; an embedding model such as BGE produces vectors for retrieval. Encoder models such as DistilBERT are suited to representation or classification tasks.
- **Specialized models:** CodeLlama illustrates code-oriented generation; Whisper performs speech-to-text; Llama Guard illustrates safety classification. Compare task-specific quality, speed and hosting requirements.
- **Context window** is model-specific. Compare the actual configuration of models such as MPT, DBRX or Llama instead of assuming a family name guarantees enough context.
- **Selection trade-offs:** parameter count, precision, memory, token throughput and task quality affect hosting cost. MTEB concerns embeddings; code benchmarks concern code tasks; MMLU measures broad knowledge.

Grounding, adaptation and token budgets

- **RAG** supplies retrieved evidence at inference time, making it useful for private or frequently updated knowledge. **Fine-tuning** changes model behavior or task performance using training examples.
- **Prompting** is the first adjustment for tone, format and instructions. Consider fine-tuning when repeatable behavior remains inadequate and suitable training data is available.
- **Context budget:** system instructions + conversation + retrieved passages + user input + reserved output must fit the generator window. Count tokens with the relevant tokenizer.

Solution Design / Data & Retrieval

Databricks Certified Generative AI Engineer Associate

KNOWLEDGE CHEATSHEET

GROUND A RESPONSE IN EVIDENCE

- **Parse + chunk** → **Embed + index** → **Filter + retrieve** → **Rerank + generate**

- **Overflow controls:** reduce retrieved passages, chunk size or conversation history; compress context when needed. Reducing the output allowance also frees budget, at the risk of truncating answers.
- **Confidential workloads** require approved data residency and hosting boundaries. A self-hosted open-weight generator may meet a no-third-party requirement; check every embedding, logging and tool dependency too.

02 Data preparation and vector retrieval

Prepare useful, traceable chunks

- **Ingestion:** parse text, clean noise, split documents, attach metadata, embed and index. At runtime, embed the query, retrieve context and generate an evidence-grounded answer.
- **Document parsers** such as unstructured extract text from supported files; HTML parsers such as Beautiful Soup isolate page content. Remove menus, advertisements and irrelevant material before indexing.
- **Chunk structure:** preserve paragraphs, section boundaries and meaningful headers. Overlap can retain continuity, but duplicates increase storage and prompt tokens.
- **Chunk enrichment** adds a section heading, parent-document summary or nearby context to explain an isolated passage. Keep the underlying evidence available; summaries can lose detail.
- **Temporal and hierarchical retrieval:** retain dates and parent-child links. Overlapping windows help continuity; hierarchical summaries can connect related concepts across long documents.
- **Delta preparation:** explode an array of chunks into one chunk per row. Assign a unique chunk ID and retain document ID, text, location and useful filter columns.

Embeddings and index lifecycle

- **Embedding fit:** evaluate domain vocabulary, supported languages, maximum input length, vector dimension, latency and cost. The embedding input limit is separate from the generator context window.
- **Vector compatibility:** documents and queries must use a compatible embedding model and dimension. Changing models normally requires re-embedding the corpus and rebuilding the affected index.
- **Delta Sync Index** follows a source Delta table. Managed embeddings compute vectors from text; self-managed embeddings supply an array of floating-point values with the configured dimension.
- **Direct Vector Access Index** requires application-managed writes and deletes. Creating an index, retrieving its handle and running similarity search are separate operations.

Data & Retrieval / Prompts & Agents

Databricks Certified Generative AI Engineer Associate

KNOWLEDGE CHEATSHEET

GROUND A RESPONSE IN EVIDENCE

Parse + chunk → **Embed + index** → **Filter + retrieve** → **Rerank + generate**

- **Change Data Feed (CDF)** supplies source changes for standard Delta Sync indexes. Validate the primary key, column types, permissions and endpoint configuration when index creation fails.
- **Sync mode:** continuous sync favors freshness; triggered sync refreshes on demand. Storage-optimized endpoints use triggered sync and suit large, less frequently refreshed corpora; validate dimension constraints.

Filter, retrieve and rerank

- **Metadata filters** enforce exact constraints such as product, model year, date range, topic or employee availability. Combine them with semantic matching rather than relying on embeddings for exact conditions.
- **Hybrid search** combines lexical and semantic signals. Keyword matching helps exact identifiers; dense vectors help paraphrases. Choose supported search features for the endpoint type.
- **Reranking** rescores an initial candidate set before context selection. Retrieve more candidates than the final prompt needs, then keep the most relevant within the token budget.
- **HNSW** uses a navigable graph; **LSH** groups nearby vectors through hashing. Compare approximate search against labeled relevance or exact-search neighbors, measuring recall, latency and resource cost.
- **Diagnose retrieval first** when responses discuss the wrong product or omit evidence. Inspect retrieved text, filters, ranking and chunk boundaries before replacing the generator.
- **Chunk experiments** hold the evaluation set stable while varying boundaries, overlap and size. Use retrieval metrics and downstream answer quality; a larger chunk is not automatically better.

03 Prompting and agent orchestration

Reliable prompts and structured outputs

- **Prompt contract:** state the task, relevant context, allowed labels, output schema and fallback for missing information. Keep instructions distinct from quoted user or retrieved content.
- **Few-shot examples** demonstrate the exact tone and format wanted. For classification, use consistent label spelling and request only the label; for extraction, demonstrate the required JSON fields.
- **Semantic precision:** define labels using the actual business rule. Product availability, customer satisfaction and successful call resolution are different classification targets.

Prompts & Agents

Databricks Certified Generative AI Engineer Associate

KNOWLEDGE CHEATSHEET

AN AGENT USES BOUNDED TOOLS

User intent → **Model decision** → **Authorized tool** → **Observation + reply**

- **Structured extraction:** specify keys, types and missing-value behavior, then parse and validate the response. A valid JSON object can still contain wrong or unsupported values.
- **LangChain PromptTemplate** needs template text and matching input variables. Supply every variable when invoking the chain; use a model component separately and escape literal braces in JSON examples.

Chains, routing and conversation

- **Prompt chaining** breaks a complex task into steps whose outputs feed later steps. It can improve control and diagnosis, but additional calls can increase cost and latency.
- **Intent routing** sends document questions to retrieval, tabular analysis to SQL tools and transactions to the appropriate business API. Route by the data and action needed.
- **Sentiment-aware support** can classify the initial message, then select response instructions that de-escalate frustration while addressing the actual request.
- **Conversation memory** retains relevant history for follow-up questions. Bound or summarize it to fit context; keep sessions isolated and avoid carrying stale facts forward.
- **Operational state** differs from chat history and semantic retrieval. Lakebase provides managed PostgreSQL for durable, transactional agent state; design user keys, access controls and retention explicitly.

Agents and live data tools

- **Tool contract:** expose a clear name, purpose, argument schema and return format. Give the model the available tool definitions and validate calls before execution.
- **ReAct** alternates model decisions, tool actions and observations until completion. LangGraph can represent branching workflows, tool execution and state transitions; set iteration and timeout limits.
- **Live facts** should come from authoritative APIs or tables. Feature Serving exposes low-latency keyed lookups from online data; SQL tools handle filters and aggregations over structured records.
- **Genie Conversational API** lets an agent interact programmatically with a Genie space for structured analytics. Preserve permissions and inspect query results before using them as answer evidence.
- **Agent Bricks:** Knowledge Assistant handles knowledge Q&A; Information Extraction produces structured fields; Multi-Agent Supervisor interprets intent and delegates to specialized agents or tools.

Prompts & Agents / Evaluation

Databricks Certified Generative AI Engineer Associate

KNOWLEDGE CHEATSHEET

EVALUATE EACH QUALITY LAYER

- **Retrieval** — Evidence
- **Generation** — Answer
- **Safety** — Policy
- **System** — Cost + latency

MCP and application interfaces

- **Model Context Protocol (MCP)** standardizes discovery and invocation of tools. Select managed Databricks servers for supported platform capabilities, external servers for third-party services and custom servers for proprietary logic.
- **MCP authentication** has two boundaries: access to the Databricks-facing service and credentials used with the underlying provider. Configure both and grant only required operations.
- **Custom computation** belongs in an executable tool, such as a proprietary risk-scoring function. A prompt describing an algorithm does not replace tested code execution.
- **User interface fit:** a Slack bot suits teams already working in Slack; Databricks Apps supports custom portals; REST endpoints support application integration. Preserve caller identity through downstream authorization.

04 Evaluation and quality

Build a representative benchmark

- **Offline evaluation** compares application variants on a curated dataset before release. Include typical questions, difficult cases, missing evidence, refusals, multilingual inputs and security cases.
- **Separate retrieval and generation:** measure whether relevant evidence was found, then whether the final answer is accurate, complete, relevant and grounded in that evidence.
- **SME rubrics** define correctness, completeness and relevance with examples and scoring anchors. Resolve disagreements through calibration rather than discarding every disputed example.
- **Controlled experiments** compare one meaningful change at a time and track data, prompt, model and retrieval configuration. Use task results alongside latency and cost to select a candidate.

Know what each metric measures

- **Recall@k** measures the fraction of relevant items retrieved; **precision@k** measures the relevant fraction among retrieved items. **NDCG** rewards placing more relevant items near the top.
- **Cosine similarity** compares vector direction; it is a retrieval signal, not proof that a generated answer is factually correct or complete.

Evaluation / Deployment & Operations

Databricks Certified Generative AI Engineer Associate

KNOWLEDGE CHEATSHEET

EVALUATE EACH QUALITY LAYER

- **Retrieval** — Evidence
- **Generation** — Answer
- **Safety** — Policy
- **System** — Cost + latency

- **BLEU** measures reference n-gram precision with a brevity penalty and is commonly used for translation. **ROUGE** measures reference overlap, often for summarization; neither fully measures meaning or safety.
- **Correctness** compares responses with expected answers or facts. **Groundedness** checks support in retrieved context. A grounded response can still be incomplete or based on inaccurate source material.
- **Relevance** asks whether the response or chunk addresses the query; **context sufficiency** asks whether retrieved evidence can support the expected answer. Correctness and sufficiency need suitable reference expectations.

Judges, custom scorers and tracing

- **LLM-as-a-judge** uses explicit criteria to score outputs and explain assessments. Calibrate against human labels and inspect bias, disagreement and repeatability.
- **MLflow GenAI evaluation** accepts datasets and scorers through **mlflow.genai.evaluate()**. Record expected answers or facts where required; legacy evaluation field names differ.
- **Custom scorer:** import **scorer** from **mlflow.genai.scorers**, decorate a function and return a numeric, boolean or Feedback result. Pass it through the evaluation call's **scorers** parameter.
- **Reusable scoring** can check prohibited phrases, schemas or business rules. Register supported scorer objects for reuse; code-based phrase checks do not replace semantic safety review.
- **MLflow tracing** exposes nested model calls, retrieval, tools, inputs, outputs, timings and errors. Inspect the failing span to diagnose a multi-step agent; an aggregate evaluation score cannot identify every runtime failure.

05 Deployment and operations

Package and serve the application

- **MLflow packaging** captures the model or application, dependencies, artifacts and interface. A custom **PyFunc** can perform preprocessing, invoke models or tools and postprocess responses.
- **Deployment path:** log the model, register it in Unity Catalog and deploy the chosen model version to Model Serving. Test its signature, dependency access and response contract.
- **Cross-workspace reuse** can use the same Unity Catalog registered model through a shared metastore and appropriate catalog, schema, model and workspace permissions. Retraining a duplicate is usually unnecessary.

Deployment & Operations

Databricks Certified Generative AI Engineer Associate

KNOWLEDGE CHEATSHEET

RELEASE AND OBSERVE

Package + register → **Evaluate + approve** → **Serve + monitor** → **Feedback + rollback**

- **Serving secrets** should be resolved through supported secret references and environment configuration. Do not hardcode credentials into prompts, notebooks or model artifacts.
- **Foundation Model API clients** include the Databricks SDK, MLflow Deployments SDK, OpenAI-compatible clients and REST. Raw requests need explicit authentication; configure host and credentials for the client actually used.

Balance throughput, latency and cost

- **Pay-per-token** suits supported models with variable or low usage; **provisioned throughput** reserves serving capacity for sustained demand. Compare utilization, concurrency, quotas and total cost.
- **Capacity planning** measures input/output token lengths, request rate and tail latency under realistic load. Larger models or longer context do not guarantee better application economics.
- **ai_query()** invokes a serving endpoint from SQL for tasks such as extracting ticket fields into table columns. Design the output schema, handle failures and persist results for downstream analytics.
- **Bulk processing** should compare endpoint capacity and pricing against job size and completion targets. Estimate both input and output tokens; avoid repeating completed inference after retries.
- **Rate limits** constrain request or token volume and help prevent runaway usage. Add bounded retries and backoff for transient failures; cache only where freshness and access isolation permit.

Observe production and control releases

- **Inference tables** capture supported request/response payloads and metadata in Delta tables for analysis. AI Gateway logging can support audits and troubleshooting; verify configured capture coverage and caller fields.
- **Operational metrics** include request volume, throughput, errors and latency. **Quality metrics** require assessments, labels or scorers; deployment alone does not establish response correctness.
- **Business metrics** connect performance to outcomes such as resolution rate, escalation rate, user engagement and retention. Monitor diversity or repetition when those affect the experience.
- **Production monitoring** analyzes live telemetry over time for regressions, drift and SLA violations. Reuse evaluation criteria and feed representative production failures into the benchmark.
- **Prompt Registry** stores prompt versions and metadata. Evaluate a candidate, promote it through a governed release and move a production alias to the approved version; retain the prior version for rollback.
- **Release discipline:** version prompts, chains and dependencies, record evaluation results and deploy through CI/CD. Tags can mark deprecated artifacts; an alias is a mutable pointer, not a copy or an approval mechanism.

Security & Governance

Databricks Certified Generative AI Engineer Associate

KNOWLEDGE CHEATSHEET

PROTECT THE ENTIRE DATA PATH

- **Caller identity** → **Authorized data** → **Bounded tools** → **Checked output**

06 Security and governance

Identity, permissions and data boundaries

- **Production authentication:** use an application service principal with supported OAuth machine-to-machine authentication. Keep credentials out of code and assign least-privilege endpoint and data permissions.
- **Authorization before retrieval:** enforce user and tenant access in the data path. A system prompt cannot stop confidential chunks from being retrieved or sent to an unauthorized model.
- **Unity Catalog** governs supported data, functions and registered models. Cross-workspace sharing still requires the right grants, workspace access and downstream resource permissions.
- **Model inversion and extraction risks** concern sensitive training information and model behavior. Combine access restrictions, request controls, output inspection and testing; no single ACL guarantees protection.

Apply layered guardrails

- **Safety controls** address harmful or toxic content; **security controls** address malicious instructions, unauthorized access and leakage; **scope and policy controls** keep responses within permitted business topics.
- **Input checks** detect disallowed requests and suspicious instruction attempts. Treat retrieved documents and tool results as untrusted data; do not let them redefine system policy.
- **Output checks** inspect toxicity, sensitive information, policy compliance and required format before delivery. Regex helps with known patterns; semantic classifiers or judges handle broader meaning.
- **Source curation** removes irrelevant or unsafe material upstream. Approved sources, retrieval filtering, system instructions and output checks complement each other.
- **Safe fallback:** request clarification when facts are missing, abstain when evidence is insufficient and route urgent or high-risk cases through the approved escalation policy. Do not invent answers to force completion.

Responsible data and response quality

- **Data rights:** document provenance, license terms and permitted use before ingestion or training. Public availability alone does not establish permission; obtain authorization where required.
- **Privacy:** minimize personal data in prompts, memory, logs and evaluation sets. Apply retention, redaction and access controls to inference tables and traces as well as source documents.
- **Safety evaluation** checks harmful content, sensitive-data disclosure, misleading claims and adherence to the use-case policy. Translation accuracy and response speed alone do not establish safety.
- **Human review** is useful for sensitive content, uncertain cases and rubric calibration. Reviewers need the evidence and task criteria; user feedback alone is not a preventive guardrail.