Surveying the AI Landscape

A technical taxonomy for faster business decisions

Lee Farretta and Matthew Farretta • LMM Labs

Prepared in collaboration with ChatGPT

Research cutoff October 1, 2026 • Published October 17, 2026



AI-generated conceptual artwork. An illustration of the landscape, not a technical diagram.

Abstract

An AI application may combine a model, company information, business software, and controls over how work proceeds. This article organizes those capabilities into five working areas and connects them to product choices. It develops the functional model introduced in Making Sense of AI and the Acceleration Opportunity into a practical taxonomy. Figure 1 maps capabilities to product examples; Table 1 supplies ten implementation categories, fifty candidates, and selection criteria. Figure 2 explains how to use both to define requirements, shortlist components, and test their fit. Table 2 applies that process to ten business functions, followed by a worked customer-service example that recommends an architecture and compares two implementation options. These are design analyses, not measured deployment results. A glossary clarifies the classification and technical terms. The purpose is to shorten investigation without bypassing technical and business judgment.

Introduction

An AI application often combines a model with company information and business software. The design challenge is deciding which parts are needed, how they connect, and who controls the result. A useful taxonomy classifies components by function so a team can compare products that perform the same job.

Making Sense of AI and the Acceleration Opportunity introduced five working areas for understanding those functions (Farretta & Farretta, 2026). This article connects them to named products and a method for choosing among them. The aim is to reduce repeated investigation and move sooner into useful development. After the selection flowchart, ten business applications and a worked customer-service example demonstrate how to use the taxonomy. A glossary follows the references.

Reading the taxonomy

Figure 1 answers “What does this component do?” Its branches run from the five working areas to capabilities and product examples. A product can serve several functions, so its position identifies a useful role rather than an exclusive category. Managed platforms appear in a shared band because they span several areas.

Table 1 answers “Which products should we consider for that role?” It groups the options into ten implementation categories, each with five candidates and selection criteria. Read across the row that matches a required capability. Figure 2 then shows how to turn that classification into a shortlist and a set of selection checks. There is no need to choose a product from every branch.

Learning and generating

Predictive models classify items, estimate quantities, or forecast future values. Libraries such as scikit-learn provide methods for developing and comparing these models (scikit-learn developers, n.d.). Generative models produce language, code, images, or other content. Model services expose these capabilities through application programming interfaces (APIs), allowing a team to use a model without training its own.

Training and fine-tuning change a model’s learned parameters. Instructions and retrieved documents influence a response without, by themselves, changing those parameters. This matters when company information changes: updating access to documents is different from adapting the model itself. Table 1 separates predictive development from generative APIs so the shortlist begins with the kind of task the application must perform.

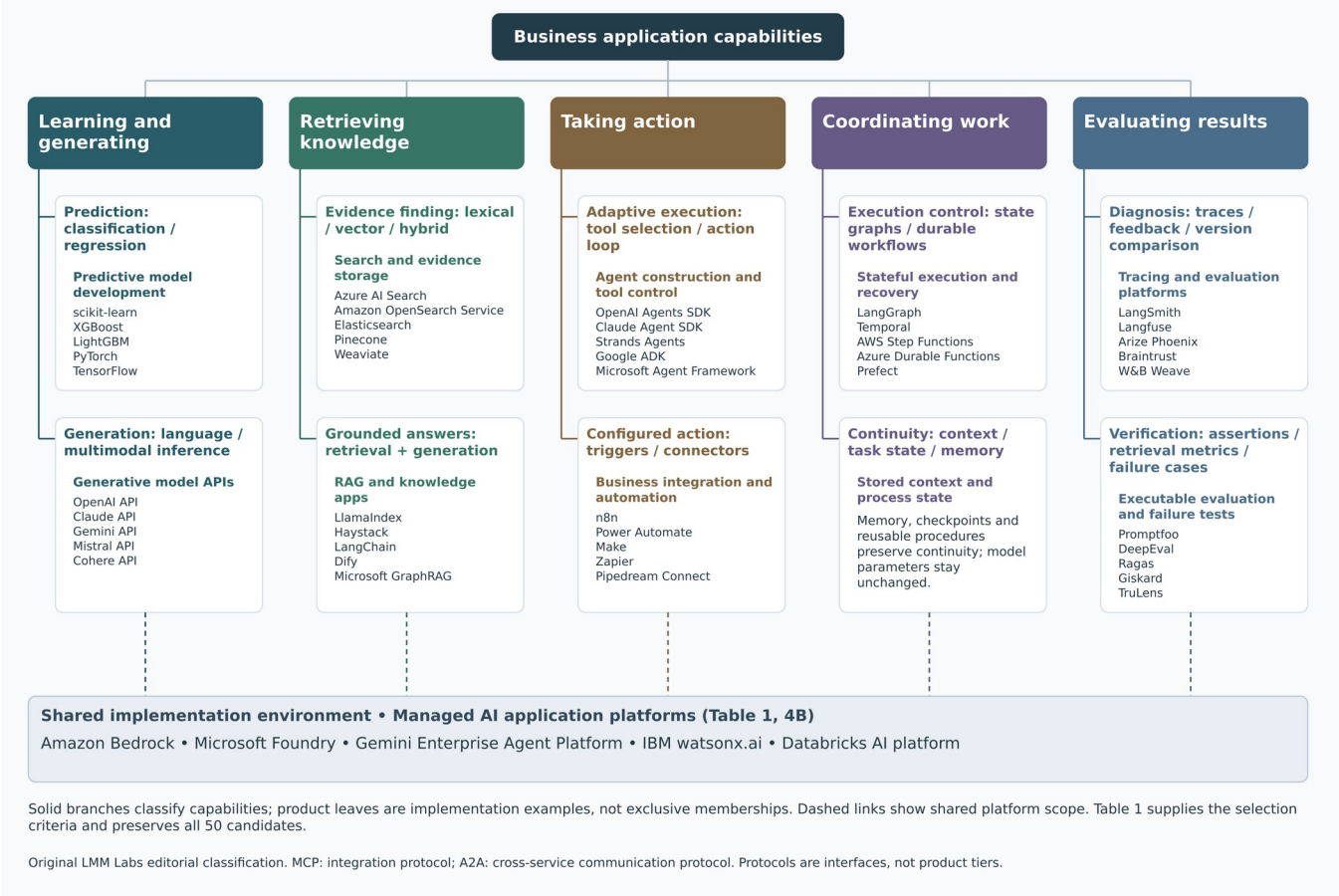
Retrieving knowledge

Retrieval finds information outside the model. Keyword search is useful for exact terms and identifiers. Vector search compares numerical representations, called embeddings, to find related meaning. Hybrid search combines the two (Microsoft, 2026). The application must also check permissions and document revisions; similarity alone does not establish that information is authorized or applicable.

Retrieval-augmented generation (RAG) gives retrieved material to a model as evidence for its response (Lewis et al., 2020). Table 1 separates search and storage products from tools for building RAG applications. The latter help connect document processing, retrieval, and generation, but the design still needs source attribution and answer checks. GraphRAG adds relationships and summaries for questions spanning a collection (Edge et al., 2025). Consider that extra work when the questions require it.

Figure 1

Functional taxonomy from working areas to product examples



Note. Original LMM Labs taxonomy. Follow each branch from a working area to its capabilities and product examples. Dashed links show the shared scope of managed platforms. Product placement identifies a role, not exclusive membership; Table 1 provides the complete shortlist and selection criteria.

Taking action

Tools connect AI to software operations, such as reading a database or submitting a business record. The model can propose an operation and its arguments; application code validates and executes the request. The system must enforce permissions and any required human approval.

A workflow follows predefined steps and branches. An agent selects its next actions from intermediate results, a pattern explored in ReAct (Yao et al., 2023). A workflow can contain a bounded agent stage where adaptation is useful. Compare that flexibility with its added cost and delay (Anthropic, 2024). Table 1 distinguishes agent construction from business integration. Model Context Protocol (MCP) standardizes connections to tools and context; it does not set the application's authority or approval policy (Model Context Protocol, n.d.).

Coordinating work

Coordination manages steps, waits, and recovery. Context is the information supplied to a model call. Task state records progress and pending decisions. Retained memory makes selected information available to later work. Saving any of these does not, by itself, change the model's learned parameters.

LangGraph supports flows combining models and code; Temporal supports durable workflows across failures and long waits (LangChain, n.d.-a; Temporal Technologies, n.d.). Recovery must also account for external actions. Repeating a request can duplicate a completed write if the receiving system lacks suitable protection. Agent2Agent (A2A) supports exchanges of tasks and results between agent services (A2A Project, n.d.). Use these components where the process needs them, rather than adding coordination layers by default.

Evaluating results

Evaluation checks whether the system meets its requirements. Observability records what happened so a result can be investigated. A trace links the sources, model calls, tool results, and outcome of an execution (LangChain, n.d.-b). Table 1 separates platforms for these records and feedback from tools that run checks, while recognizing that their features overlap.

The checks must match the failure. A formatted answer can still use the wrong evidence. RAGAs examines retrieval and answer quality; τ-bench examines tool use, final state, and consistency across repeated trials (Es et al., 2024; Yao et al., 2024). These methods inform testing, but an application needs evidence from its own data and integrations. Include correction, human review, elapsed time, and total cost when judging whether it accelerates accepted work.

Using the product map

Use the categories to compare products that address the same requirement. Table 1 identifies differences worth testing, including exact-identifier search, permission filters, approval waits, recovery, and records that explain a failed case. A managed platform may cover several roles, but its services must still fit the application’s data, deployment, and integration needs.

Check product lifecycle as well as features. Pipedream Connect, for example, is distinct from the Workflows product scheduled to close in March 2027 (Pipedream, n.d.). Keep the shortlist tied to the required capability and the work that remains. Figure 2 turns these comparisons into a selection process.

Applying the taxonomy

Figure 2 shows how to use the taxonomy and product table together. Begin with a specific business outcome and decide what would count as success. Use Figure 1 to identify the capabilities needed, then Table 1 to shortlist products for those roles.

Check whether the candidates can work together within the data, permission, deployment, and cost limits. Test them with the same representative inputs and acceptance criteria. If a candidate fails, return to the shortlist. If none fits, make any scope change an explicit business decision. Record the selected components and remaining integration work for the system blueprint and make-or-buy analysis.

Table 1

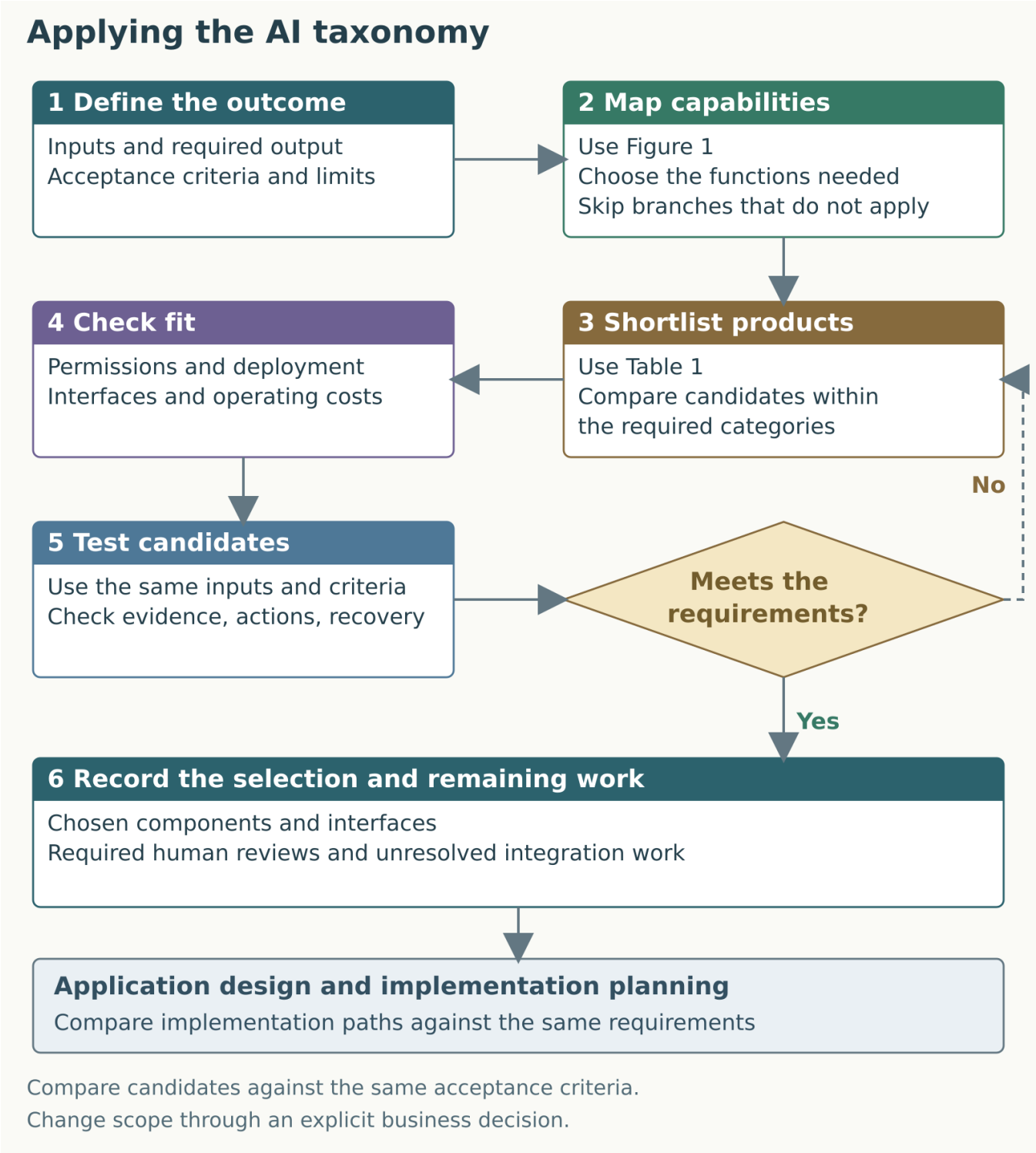
Five working areas and ten implementation categories

Working area	Implementation category A	Implementation category B
Learning and generating	**1A Predictive model development** scikit-learn · XGBoost · LightGBM · PyTorch · TensorFlow **Selection criteria.** Start with scikit-learn baselines and compare XGBoost/LightGBM on tabular data. Use PyTorch/TensorFlow when a neural model is justified. Test leakage, calibration and costly errors—not accuracy alone.	**1B Generative model APIs** OpenAI API · Claude API · Gemini API · Mistral API · Cohere API **Selection criteria.** Compare the same document, extraction and tool-call cases across APIs. Require valid output schemas, adequate modality support, acceptable response time and cost, and suitable data-retention terms for the chosen model.
Retrieving knowledge	**2A Search and evidence storage** Azure AI Search · Amazon OpenSearch Service · Elasticsearch · Pinecone · Weaviate **Selection criteria.** Test exact equipment IDs alongside semantic queries. Require revision/date filters and document access controls before evidence reaches the model. Compare indexing freshness, ranking quality and the burden of operating the search service.	**2B RAG and knowledge apps** LlamaIndex · Haystack · LangChain · Dify · Microsoft GraphRAG **Selection criteria.** Use conventional RAG for passage-level facts; consider GraphRAG for relationships and collection-wide synthesis. Compare coded pipelines with visual assembly. Check parsing, source provenance and whether added indexing work improves the required answers.
Taking action	**3A Agent construction and tool control** OpenAI Agents SDK · Claude Agent SDK · Strands Agents · Google ADK · Microsoft Agent Framework **Selection criteria.** Test approval before writes, saved-state resume, tool-argument validation and failure handling. Compare built-in execution tools with custom tools. Choose an SDK whose process, sandbox and model-provider requirements fit the application.	**3B Business integration and automation** n8n · Power Automate · Make · Zapier · Pipedream Connect **Selection criteria.** Verify connector operations, authentication scopes and trigger delivery. Test retries and duplicate events before writes. Compare visual configuration with coded API steps. Use Pipedream Connect for developer integrations; Pipedream Workflows retires in March 2027.
Coordinating work	**4A Stateful execution and recovery** LangGraph · Temporal · AWS Step Functions · Azure Durable Functions · Prefect **Selection criteria.** LangGraph controls mixed model/code paths; Temporal supports durable business processes; AWS/Azure runtimes fit their cloud estates. Test recovery during an approval wait and after a write. Repeated delivery must not repeat the business action.	**4B Managed AI application platforms** Amazon Bedrock · Microsoft Foundry · Gemini Enterprise Agent Platform · IBM watsonx.ai · Databricks AI platform **Selection criteria.** Compare model availability, identity/data integration, deployment boundaries and required feature maturity. Count custom integration work and operating responsibility. Existing cloud alignment helps only if the required services and controls are actually available.
Evaluating results	**5A Tracing and evaluation platforms** LangSmith · Langfuse · Arize Phoenix · Braintrust · W&B Weave **Selection criteria.** Require one trace linking retrieved sources, model calls, tool results and the final outcome. Compare feedback-to-dataset workflows, version comparisons, sensitive-data handling and export. Trace visibility must support diagnosing a specific failed case.	**5B Executable evaluation and failure tests** Promptfoo · DeepEval · Ragas · Giskard · TruLens **Selection criteria.** Match the test to the failure: Ragas for retrieval/answer metrics, Promptfoo for output assertions, DeepEval for agent trajectories. Require known failure cases and executable outcome checks; calibrate model judges against expert decisions.

Note. Managed AI platforms (4B) span generation, retrieval, action, and evaluation as well as coordination. Primary placement is a navigation aid, not exclusive membership. The ten categories preserve all 50 product entries and their full original selection criteria. Product information retains the October 1, 2026 research cutoff; the companion source register supplies the official links. No performance ranking is implied.

Figure 2

Using Figure 1 and Table 1 to select components



Note. Original LMM Labs selection guide. These are component-selection checks, not proof that a complete application is ready for use. A failed candidate returns to the shortlist; a scope change requires an explicit business decision.

Applying the taxonomy across business functions

Table 2 records the preliminary design choices reached by applying the six steps in Figure 2 to ten representative business functions. Each row maps an application to the relevant taxonomy categories, an architecture, candidate products, and the checks that would decide the choice. These are worked design mappings; product performance has not been measured.

Table 2

Business applications mapped to architecture and candidate components

Business function and application	Architecture and Table 1 categories	Candidate components	Reason for the choice and decisive check
Finance and accounting **Invoice exceptions**	Extract fields, match records using coded rules, and route exceptions. Categories 1B, 3B	OpenAI API or Claude API for extraction; Power Automate for routing.	Rules govern matching. Verify totals and duplicate IDs; staff approve posting and payment.
Sales **Account and proposal brief**	Retrieve authorized account evidence and approved collateral; draft a cited brief. Categories 1B, 2A, 2B	Azure AI Search + OpenAI API; alternatively Amazon OpenSearch Service + Amazon Bedrock.	Current evidence matters more than autonomy. Test customer isolation, document dates, and unsupported pricing claims.
Marketing **Campaign content**	Generate from an approved brief, review, then schedule release. Categories 1B, 3B	Gemini API or Cohere API for drafting; Make for configured handoffs.	Repeatable approvals favor a workflow. Review factual claims and brand compliance before publication.
Customer service **Response drafting**	Retrieve authorized policies, draft with citations, and require representative review. Categories 1B, 2A, 4B	Azure AI Search + OpenAI API; alternatively Amazon Bedrock Knowledge Bases with a configured vector store.	Evidence and review suffice initially. Test obsolete policies, missing evidence, and access boundaries.
Human resources **Policy guidance**	Retrieve authorized policies; cite answers and escalate personal cases. Categories 1B, 2A, 2B	LlamaIndex + Claude API; alternatively Dify + OpenAI API, with an appropriate search store.	Guidance needs controlled evidence. Test jurisdiction and effective dates; exclude restricted personnel records and employee ranking.
Operations **Incident investigation**	Run bounded, read-only diagnostic tools inside a controlled agent workflow. Categories 1B, 2A, 3A, 4A	LangGraph + OpenAI Agents SDK; alternatively Strands Agents + Amazon Bedrock.	Findings change the diagnostic plan. Limit tool calls; require evidence and approval before operational changes.
Supply chain and logistics **Demand and replenishment**	Schedule predictive models; enforce coded inventory limits and planner review. Categories 1A, 4A	scikit-learn baseline versus XGBoost; Prefect schedules the selected pipeline.	Forecasts need numerical validation. Test against a simple baseline using time-based splits and shortage costs; planners approve orders.
Procurement **Bid comparison**	Extract bid evidence, calculate comparable terms, and request buyer approval. Categories 1B, 2A, 2B, 3B	Haystack + Claude API or LlamaIndex + OpenAI API; Power Automate for approval.	Evidence and arithmetic drive comparison. Check units, currencies, exclusions, and missing terms; a buyer selects the supplier.
Information technology **Ticket diagnosis**	Retrieve runbooks, run bounded diagnostics, and separately approve changes. Categories 1B, 2A, 3A, 4A	LangGraph + OpenAI Agents SDK; alternatively Strands Agents + Amazon Bedrock.	Adaptive diagnosis needs narrow authority. Test injected instructions, retries, and rollback in a sandbox.
Legal and compliance **Contract review**	Extract clauses, retrieve a review playbook, and cite deviations. Categories 1B, 2A, 2B	Haystack + Claude API; alternatively LlamaIndex + OpenAI API, with an appropriate search store.	Clause traceability matters. Test omissions, amendments, and jurisdiction; qualified reviewers retain legal decisions.

Note. Category IDs refer to Table 1. The mappings apply the flowchart to defined business tasks; they are not a ranking. Listed components are not complete stacks. Amazon Bedrock Knowledge Bases supports retrieval and cited generation (Amazon Web Services, n.d.). Verify product editions and interfaces, and specify hosting, storage, permissions, and integration work before implementation.

Worked example in customer service

Consider a mid-market equipment distributor whose representatives search product manuals and warranty policies before replying to customers. The following fictional case works through Figure 2; it is not a customer deployment or a product benchmark.

1 Define the outcome. Produce a draft reply with the applicable policy passage and a source link. Assume an Azure-hosted application, authenticated staff, and a maintained document collection. The first release cannot send replies, authorize refunds, or promise replacements. Success means less total handling time, including review and corrections, without weakening evidence or access controls.

2 Map the capabilities. Figure 1 points to generation, retrieval, and evaluation. Table 1 narrows the shortlist to 1B and 2A, with 5A and 5B for testing and trace review. A fixed retrieval-and-drafting workflow is sufficient; a free-form agent would add planning work without a demonstrated need (Anthropic, 2024). Ticketing remains the system of record.

3 Shortlist the products. Start with Azure AI Search for hybrid retrieval (Microsoft, 2026) and the OpenAI API for drafting. Application code passes retrieved passages to the model. An AWS alternative is Amazon Bedrock Knowledge Bases, with its model and vector-store configuration, including an Amazon OpenSearch option (Amazon Web Services, n.d.). The stated Azure environment favors the first pair; it does not establish superior answer quality.

4 Check the fit. The application authenticates the representative, filters documents by authorization, product, and effective date, retrieves evidence, drafts a response, and displays it for review. It stores the question, source revisions, draft, and disposition. Validate structured output and citations in code: schema compliance does not establish factual correctness (OpenAI, n.d.). Confirm retention terms, region, indexing updates, and total operating cost before committing.

5 Test a concrete case. Use a synthetic request: “A ZX-40 failed after 18 months. Can I promise a replacement?” Supply a current test policy stating 24 months of coverage subject to inspection, plus an obsolete 12-month policy. The expected draft cites the current policy and explains that inspection is required; it must not promise replacement. Repeat with the current policy missing and with access denied: escalate in the first case and exclude unauthorized evidence in the second. These are specified tests and expected outcomes, not observed product results.

6 Record the recommendation. Recommend the Azure AI Search and OpenAI API pair for an initial read-only pilot under the stated assumptions. Retain Bedrock Knowledge Bases as the AWS alternative. Compare both on the same reviewed cases, including conflicting documents and instructions embedded in retrieved text. Reject unsafe behavior; require an agreed improvement in total handling time and acceptable cost. Record model versions, configuration, integration work, and test results. Add an agent only if measured retrieval failures justify adaptive searches.

Conclusion

The taxonomy is useful when it narrows a real decision. It should leave a team with a component shortlist, reasons for each choice, and specific questions to resolve in design. Figure 2 provides that path without requiring every technology in the map. Human goals, architecture, and review remain central to LMM Labs’ development methodology (LMM Labs, n.d.). Use the selection record to guide the application’s design and implementation planning.

Research and preparation note

Technical explanatory synthesis with an original editorial taxonomy and selection flowchart. Research cutoff and product information: October 1, 2026. Editorial revision: October 17, 2026. Sources and product roles remain documented in the companion register. The flowchart is an editorial guide, not a validated selection model. No comparative experiment, new performance measurement, or market-share ranking is reported. Prepared in collaboration with ChatGPT; Lee Farretta is executive editor. © 2026 LMM Labs. Third-party rights are retained.

References

A2A Project. (n.d.). *Core concepts*. https://a2a-protocol.org/latest/topics/key-concepts/

Amazon Web Services. (n.d.). *Retrieve data and generate AI responses with Amazon Bedrock Knowledge Bases*. https://docs.aws.amazon.com/bedrock/latest/userguide/knowledge-base.html

Anthropic. (2024, December 19). *Building effective agents*. https://www.anthropic.com/engineering/building-effective-agents

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., Metropolitansky, D., Ness, R. O., & Larson, J. (2025). *From local to global: A GraphRAG approach to query-focused summarization* [Preprint]. arXiv. https://arxiv.org/abs/2404.16130v2

Es, S., James, J., Espinosa Anke, L., & Schockaert, S. (2024). RAGAs: Automated evaluation of retrieval augmented generation. In N. Aletras & O. De Clercq (Eds.), *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations* (pp. 150–158). Association for Computational Linguistics. https://aclanthology.org/2024.eacl-demo.16/

Farretta, L., & Farretta, M. (2026, September 30). *Making sense of AI and the acceleration opportunity*. LMM Labs. https://lmmlabs.com/articles/making-sense-of-ai-and-the-acceleration-opportunity/

LangChain. (n.d.-a). *LangGraph overview*. https://docs.langchain.com/oss/python/langgraph/overview

LangChain. (n.d.-b). *LangSmith observability*. https://docs.langchain.com/langsmith/observability

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, *33*, *9459–9474*. https://papers.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

LMM Labs. (n.d.). *Development approach—Methodology*. https://lmmlabs.com/development-methodology/

Microsoft. (2026, August 31). *Hybrid search using vectors and full-text search in Azure AI Search*. https://learn.microsoft.com/en-us/azure/search/hybrid-search-overview

Model Context Protocol. (n.d.). *Architecture overview*. https://modelcontextprotocol.io/docs/2026-07-28/learn/architecture

OpenAI. (n.d.). *Structured model outputs*. https://developers.openai.com/api/docs/guides/structured-outputs

Pipedream. (n.d.). *Workflows and String are shutting down on March 31, 2027*. https://pipedream.com/docs/workflows

scikit-learn developers. (n.d.). *User guide*. https://scikit-learn.org/stable/user_guide.html

Temporal Technologies. (n.d.). *Temporal Workflow Execution overview*. https://docs.temporal.io/workflow-execution

Yao, S., Shinn, N., Razavi, P., & Narasimhan, K. (2024). *τ-bench: A benchmark for tool-agent-user interaction in real-world domains* [Preprint]. arXiv. https://arxiv.org/abs/2406.12045v1

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. *International Conference on Learning Representations*. https://arxiv.org/abs/2210.03629v3

Appendix

Glossary

Terms describe their use in this article. Business functions identify where work occurs; working areas and implementation categories organize the technology used to support it.

Acceptance criteria. Agreed conditions used to accept an application or compare candidates, including quality, safety, cost, and response time.

Agent. A model-driven execution pattern that selects actions or tools using intermediate results, within application-defined limits.

API and SDK. An application programming interface defines how software requests a service. A software development kit supplies code and tools for using interfaces or building applications.

Architecture. The arrangement of application components, their interfaces, data flows, execution control, and deployment boundaries.

Business function. An organizational responsibility, such as finance or customer service. It describes where work occurs, not a technical layer.

Capability. A function a component can perform, such as forecasting, finding evidence, or executing an approved action.

Chatbot. A conversational interface. Its underlying execution can use a simple model call, a workflow, an agent, or a combination.

Context and task state. Context is information supplied to a model call. Task state records execution progress and pending decisions between steps.

Embedding and vector search. An embedding represents content numerically. Vector search compares these representations to retrieve related content.

Evaluation and observability. Evaluation tests whether requirements are met. Observability records execution so behavior and failures can be investigated.

Fine-tuning. Additional training that changes model parameters. Supplying instructions or retrieved documents does not itself fine-tune a model.

Hybrid search. Retrieval combining keyword and vector methods so exact terms and semantic similarity can both contribute.

Idempotency. The property that repeating an operation has the same intended effect as performing it once, important when retrying writes.

Implementation category. One of Table 1's ten groupings of products by an implementation role. Categories are not mutually exclusive vendor memberships.

LLM. A large language model trained to model language patterns and generate responses. Its output still requires task-specific validation.

MCP and A2A. Model Context Protocol connects applications to tools and context. Agent2Agent supports task exchanges between agent services. Neither replaces authorization policy.

Orchestration and workflow. Orchestration controls execution, state, and recovery. A workflow specifies steps and branches in advance; it can contain bounded agent stages.

Provenance and trace. Provenance identifies a result's sources and revisions. A trace records the linked steps, calls, tool results, and outcome of one execution.

RAG and GraphRAG. Retrieval-augmented generation supplies retrieved evidence to a generative model. GraphRAG uses extracted relationships and summaries for graph-based retrieval and synthesis.

Working area. One of this article's five broad functional groupings: learning and generating, retrieving knowledge, taking action, coordinating work, and evaluating results.