

FDE FRAMEWORKS GUIDE



What Are Frameworks in Forward Deployed Engineering?

A software or AI framework is a structured collection of reusable components, abstractions, and conventions that help developers build applications faster and more reliably. Rather than constructing every integration, workflow, and data pipeline from scratch, FDEs use frameworks to focus their energy on solving the customer's specific business problem.

Frameworks matter in Forward Deployed Engineering because the work is inherently time-pressured and context-sensitive. Customers expect working solutions, not research prototypes. Frameworks provide the scaffolding – the connection points to AI models, databases, external APIs, and deployment environments – so the FDE can concentrate on the logic that is unique to the customer's situation.



Reduce Repetitive Work

Reuse proven patterns for authentication, data retrieval, model calls, and error handling rather than rebuilding them for every project.



Connect Systems

Integrate AI models, external APIs, databases, and enterprise tools through standardised abstractions and built-in connectors.



Organise Workflows

Structure agent behaviour, task sequencing, tool interactions, and decision logic in maintainable, testable code.



Support Quality

Enable testing, debugging, tracing, and deployment through built-in observability hooks and integration with evaluation tooling.

❏ **Key Takeaway:** Frameworks provide reusable building blocks, but the FDE must still deeply understand the customer's problem, data architecture, security requirements, and deployment environment. A framework does not make architectural decisions – the engineer does.

The FDE Technology Stack

A typical Forward Deployed Engineering project draws on several distinct categories of technology. These are not interchangeable substitutes – some are full frameworks, some are libraries, some are protocols, and others are infrastructure components. Understanding which layer each technology belongs to is essential before selecting a stack for a real customer project.

Each layer depends on the ones below it. Deployment and operations wrap every other layer – even the most sophisticated agent orchestration framework is useless if the application cannot be reliably deployed, monitored, and maintained in the customer's environment. When evaluating a stack, always consider compatibility across all six layers before committing to a specific tool in any one of them.

LangChain and LangGraph

Build AI Applications and Control Complex Workflows

LangChain and LangGraph are complementary frameworks from the same organisation, but they serve meaningfully different purposes. Understanding the distinction allows an FDE to choose the right tool for each part of an application.

LangChain

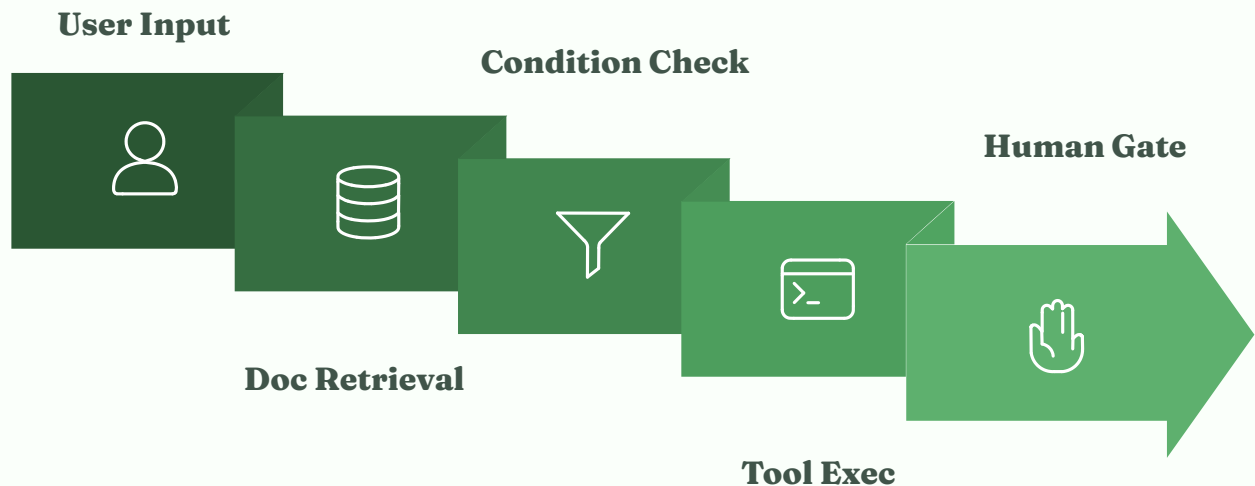
LangChain helps developers build LLM-powered applications by providing a unified interface for interacting with language models, tools, and data sources. It offers pre-built integrations with dozens of model providers, vector stores, and external APIs, making it practical for rapid application development. LangChain is particularly well-suited to building retrieval pipelines, prompt management workflows, and tool-calling applications.

- Unified model and tool integrations
- Prompt templates and chains
- Built-in retrieval components
- Wide ecosystem of connectors

LangGraph

LangGraph adds stateful, graph-based workflow control on top of LangChain's foundation. Instead of linear chains, LangGraph allows developers to define workflows as directed graphs with explicit branching, looping, checkpointing, and the ability to pause execution for human review. This makes it appropriate for enterprise workflows that involve conditional logic, multi-step approval processes, or operations that must be resumable after interruption.

- Graph-based stateful workflows
- Checkpointing and persistence
- Human-in-the-loop approval steps
- Explicit branching and loop control



FDE Use Case: An enterprise AI assistant that retrieves customer account information, checks account conditions, prepares a recommendation, and pauses for human approval before executing a sensitive downstream action – such as a refund or account change.

- ❑ **Consideration:** Complex LangGraph workflows require careful state schema design, thorough evaluation of branching logic, and robust error handling. Start with the simplest graph that solves the problem.

CrewAI

Coordinate Multiple AI Agents Around a Business Task

CrewAI is a framework for building role-based multi-agent systems. It introduces an organising metaphor that maps well onto how real business teams operate: agents are assigned specific roles and responsibilities, tasks are defined with dependencies, and the crew executes a structured workflow toward a shared goal.

CrewAI supports both sequential execution – where tasks run one after another – and hierarchical execution, where a manager agent coordinates the activities of specialist agents. This flexibility makes it useful for business processes that have clearly defined stages and reviewable outputs at each step.



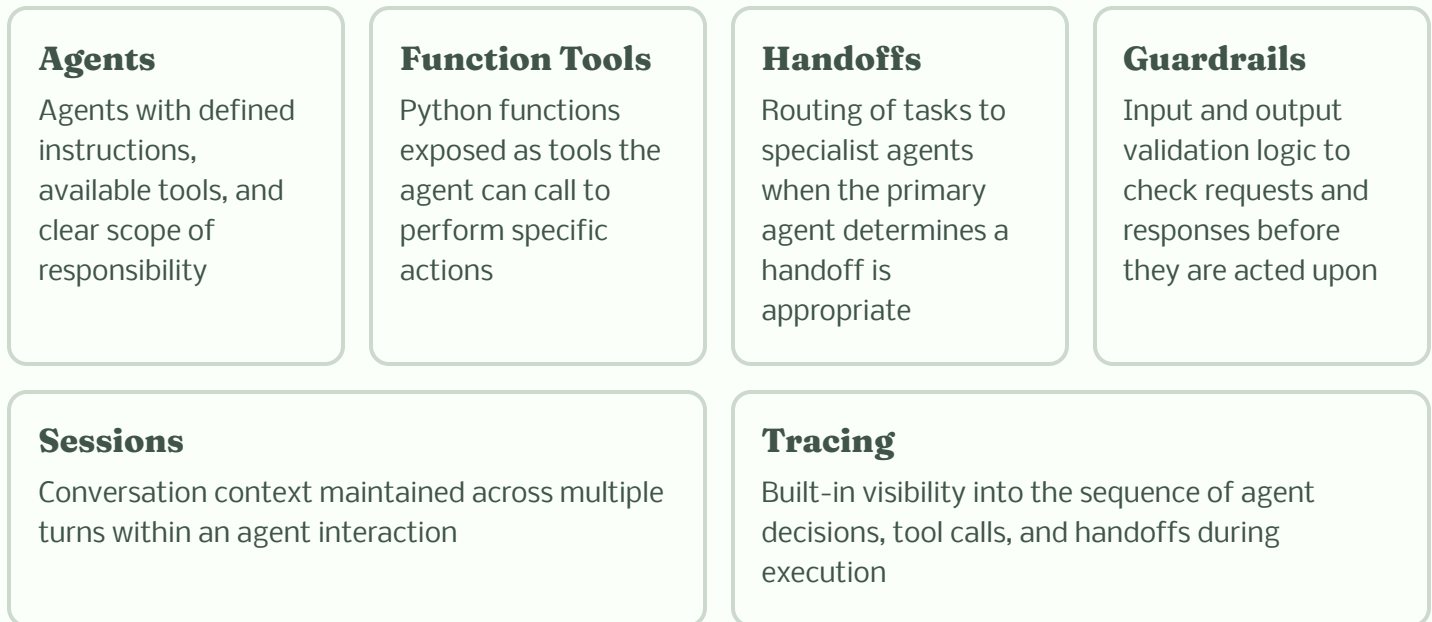
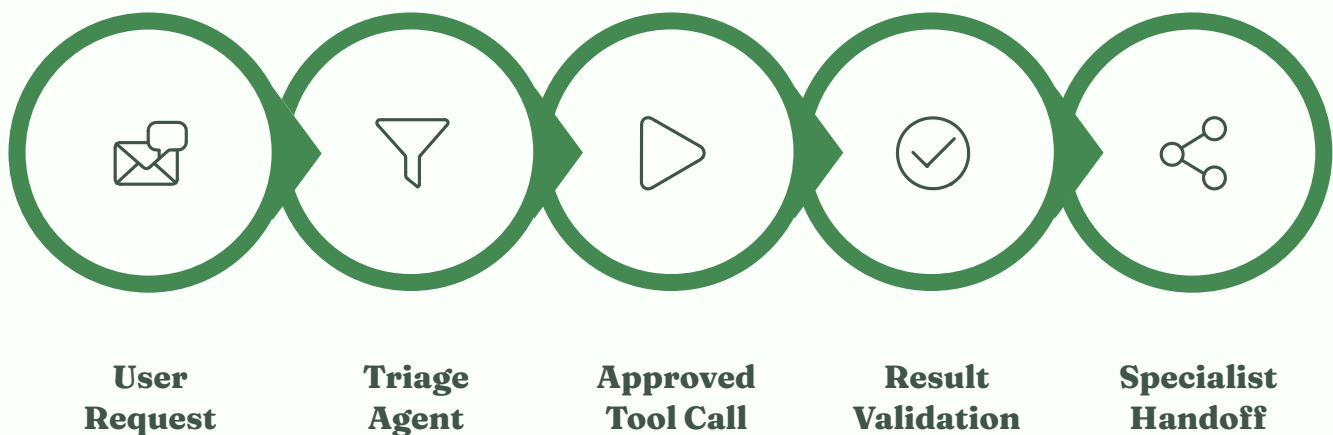
FDE Use Case: An insurance claims review workflow in which separate agent stages extract information from submitted documents, verify policy details, identify missing information, and prepare a formatted case summary – each stage producing auditable output for the next.

⚠ Important: Multiple agents are not automatically better. Each additional agent introduces latency, cost, coordination complexity, and additional failure points. A single well-designed agent or a conventional workflow pipeline is often simpler, cheaper, and more reliable. Choose multi-agent architecture only when the problem genuinely requires it.

OpenAI Agents SDK

Build Agents That Use Tools and Delegate Tasks

The OpenAI Agents SDK provides a focused development model for building agents that use tools and hand off tasks to specialist agents. It is designed around a small set of core primitives that cover the most common patterns in tool-using agent applications, making it practical to learn and apply quickly in a real customer engagement.



FDE Use Case: A customer support assistant that identifies the issue category, retrieves approved account information via defined function tools, routes billing questions to a billing specialist agent, and validates the response before returning it to the user.

Considerations: The SDK provides the agent runtime, but developers remain responsible for designing secure tool permissions, implementing application-level error handling, building evaluation pipelines, and managing deployment infrastructure. Tracing is a valuable starting point for debugging agent behaviour, but it does not replace systematic evaluation.

Microsoft Agent Framework

Build Agent Workflows for Enterprise Environments

Microsoft Agent Framework represents the current unified direction from the teams behind AutoGen and Semantic Kernel for building agent-based applications in enterprise environments. It is relevant for organisations that are already invested in Microsoft and Azure technologies and need an agent development approach that integrates with their existing identity, security, and service infrastructure.

AutoGen and Semantic Kernel continue to exist as distinct projects, but Microsoft Agent Framework brings together workflow orchestration, agent development primitives, middleware, and enterprise integration in a single cohesive model. When evaluating Microsoft's agent ecosystem, understand how these projects relate to one another before selecting a specific component.

Agent Development

Define agents with instructions, capabilities, and tool access in Python or .NET

Workflow Orchestration

Coordinate multi-step agent workflows with defined execution patterns and state management

Enterprise Integration

Connect to Microsoft 365, Azure services, identity providers, and approved business applications

Observability

Built-in middleware and monitoring hooks compatible with Azure observability tooling

FDE Use Case: An internal enterprise assistant that interacts with approved business services – such as SharePoint, Dynamics, and Azure Cognitive Search – retrieves relevant records, and routes tasks through defined approval workflows enforced by organisational identity controls.

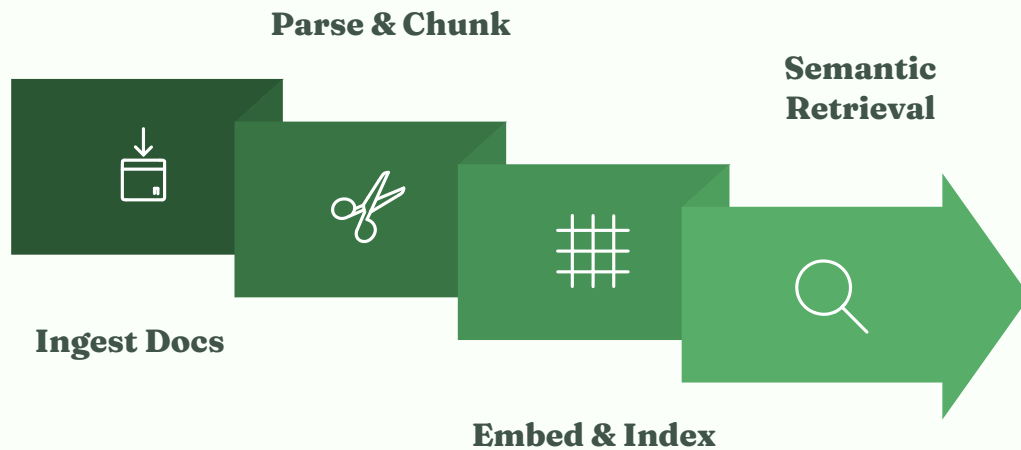


Selection Note: Confirm platform compatibility, deployment requirements, licensing, and available integrations against the customer's actual environment before recommending Microsoft Agent Framework over framework-agnostic alternatives.

LlamaIndex

Connect AI Applications to Enterprise Knowledge

LlamaIndex is a data framework for building AI applications that depend on retrieving information from documents and structured data sources. Where LangChain provides broad application development capabilities, LlamaIndex focuses specifically on the ingestion, indexing, and retrieval pipeline – making it the more natural choice when document retrieval is the core of the application.



01

Connect to Data Sources

Ingest documents from file systems, SharePoint, Confluence, databases, and other enterprise repositories using built-in data loaders.

02

Parse and Chunk

Break documents into meaningful segments that can be retrieved independently and that fit within the model's context window.

03

Embed and Index

Convert text chunks into vector representations and store them in a vector index or hybrid search system for retrieval.

04

Retrieve and Assemble Context

Retrieve the most relevant chunks for a given query and assemble them into a context block supplied to the language model.

05

Generate and Attribute

Produce an answer grounded in the retrieved context, with source attribution so users and reviewers can verify the response.

FDE Use Case: A company wants employees to query internal policies, product documentation, and technical manuals in natural language without manually searching multiple repositories. LlamaIndex ingests and indexes approved documents; queries return relevant passages alongside the generated answer.

⚠ Important: LlamaIndex does not eliminate the need for data governance, access controls, or retrieval-quality evaluation. The quality of the system depends on document quality, chunking strategy, embedding model selection, retrieval configuration, and regular testing against real user queries.

FastAPI and Application Frameworks

Turn AI Logic Into a Usable Application

FastAPI is a modern Python framework for building APIs. It is not an AI orchestration framework – it is the application layer that makes AI logic accessible to web interfaces, internal systems, mobile applications, and external clients. In a typical FDE project, FastAPI wraps the AI and retrieval logic and exposes it through structured HTTP endpoints that the customer's existing systems can call.

What FastAPI Provides

- Typed request and response validation using Pydantic models
- Automatic API documentation (OpenAPI/Swagger) generated from code
- Asynchronous request handling for high-concurrency workloads
- Dependency injection for authentication, database sessions, and shared services
- Middleware hooks for logging, rate limiting, and error handling
- Straightforward integration with AI frameworks, vector stores, and databases

FDE Use Case

An AI document-processing service accepts an authenticated POST request containing a document reference, retrieves the document from an approved store, sends it to an AI summarisation workflow, validates the structured output, and returns a JSON response to the customer's application – with full request logging and error codes.



Customer Web App

FastAPI Endpoint

Auth & Validation

AI Framework Call

Important Distinction: FastAPI is responsible for the application boundary – receiving requests, enforcing authentication, validating inputs, and returning structured responses. The AI reasoning, retrieval, and orchestration logic lives in the framework layer beneath it. Conflating these two layers leads to poorly structured, difficult-to-test codebases.

Framework Comparison Matrix

These technologies are not direct substitutes for one another. Each addresses a different layer of the application, and they are commonly used together in production deployments. The table below summarises the primary purpose and selection criteria for each framework covered in this guide.

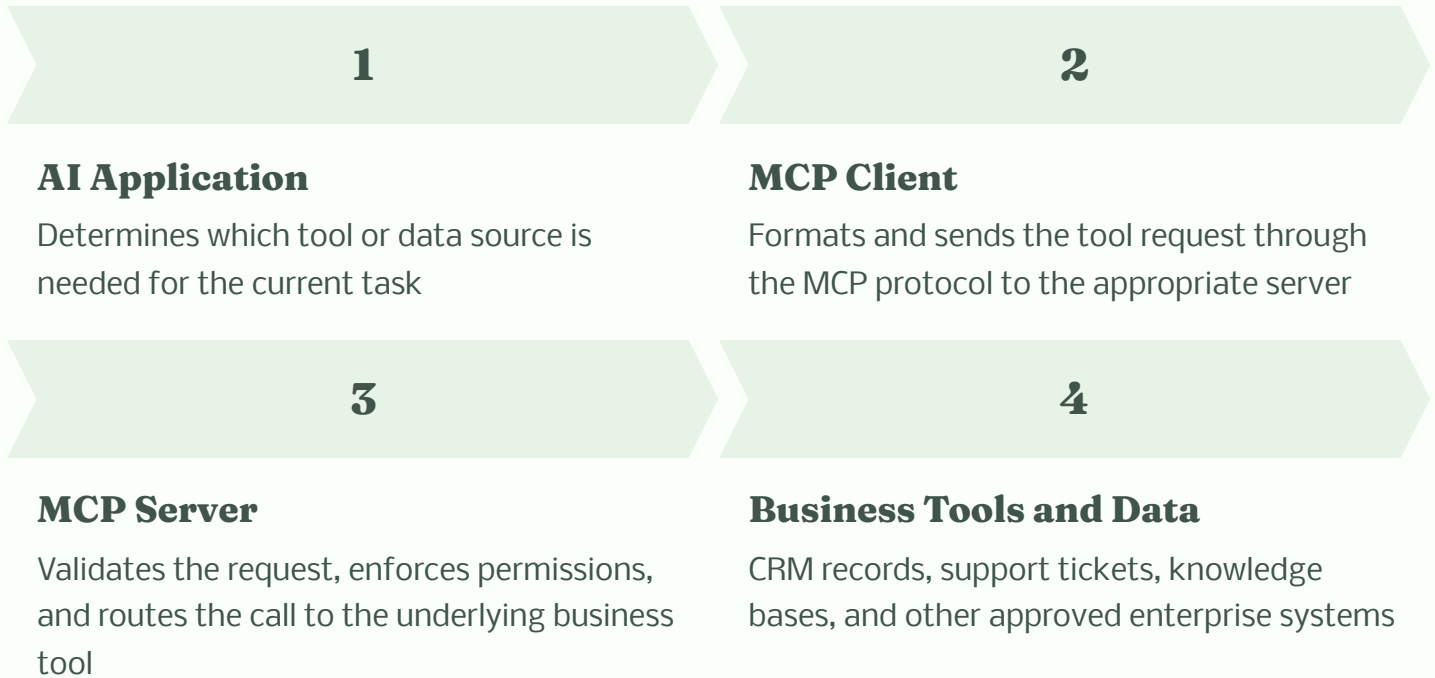
Framework	Primary Purpose	Consider Using When	Key Consideration
LangChain	LLM application development and integrations	You need reusable components and model or tool integrations	Wide ecosystem; evaluate what you actually need
LangGraph	Stateful graph-based orchestration	Workflow needs explicit control, branching, persistence, or approval points	State management complexity increases with graph size
CrewAI	Role-based agent collaboration	You want to coordinate agents around well-defined tasks	More agents mean more cost, latency, and failure points
OpenAI Agents SDK	Tool-using agents and handoffs	You need focused agent implementation using its supported capabilities	Security, evaluation, and deployment remain your responsibility
Microsoft Agent Framework	Agent development and workflow orchestration	Requirements align with Microsoft or Azure technologies	Confirm compatibility and integration support before committing
LlamaIndex	Data-centric AI and retrieval	Application depends heavily on documents and enterprise knowledge	Data quality and retrieval evaluation are critical
FastAPI	Application APIs and backend services	You need to expose application functionality through HTTP endpoints	Application framework, not an AI orchestration tool

📌 **Combining Frameworks:** These technologies are designed to complement one another. A production FDE deployment might use LlamaIndex for document retrieval, LangGraph for multi-step orchestration, and FastAPI to expose the application to the enterprise system. The right combination depends on the use case, team expertise, and deployment constraints – not on which tools are currently trending.

MCP and Enterprise Tool Integration

Connect AI Applications to Business Systems

The **Model Context Protocol (MCP)** is an open protocol for connecting AI applications with external tools and context sources through a common, standardised interface. MCP is not an agent framework – it is a communication protocol that defines how an AI application can discover, call, and receive results from external tools and data sources without requiring custom integration code for every client-server pair.



FDE Use Case: An AI assistant needs to retrieve CRM records, look up open support tickets, and search an internal knowledge base. Rather than building a bespoke integration for each system, the FDE exposes each tool through an MCP server. The AI application uses a single MCP client interface to call any of them, with authentication and permissions enforced at the server layer.

What MCP Helps With

Reducing the amount of custom integration code required per client-server pair; providing a common discovery and invocation pattern; enabling tool reuse across multiple AI applications.

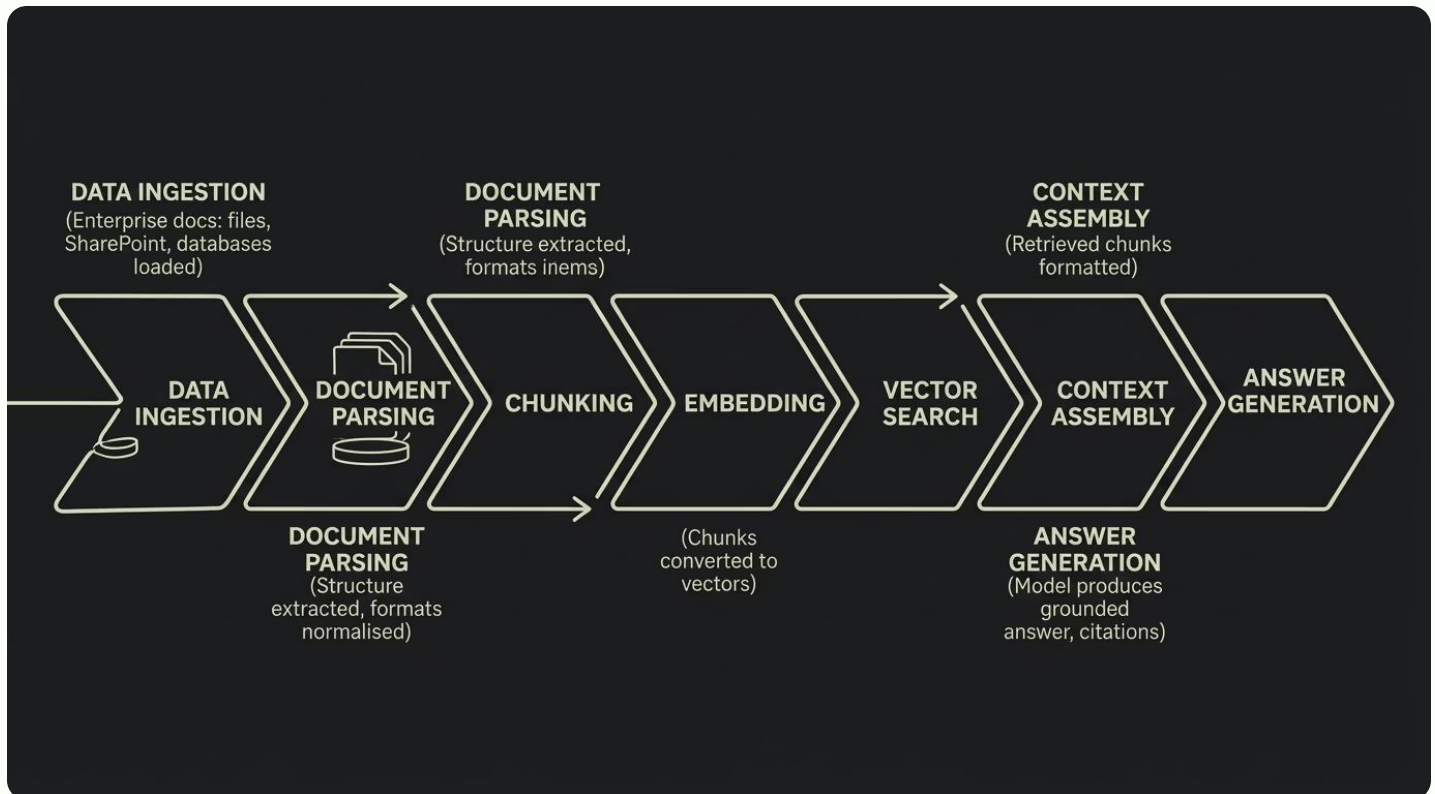
What MCP Does Not Replace

Authentication and authorisation at the tool level; input validation and output sanitisation; per-tool permission restrictions; audit logging; and data governance for sensitive business information.

RAG, Vector Search, and Data Frameworks

Give AI Applications Access to Relevant Information

Retrieval-Augmented Generation (RAG) is a pattern, not a specific product. It describes the practice of retrieving relevant information from an external data source at query time and supplying that information as context to a language model before generating a response. This grounding allows the model to produce answers based on controlled, current, and verifiable information rather than relying solely on its training data.



Vector Databases vs RAG Frameworks

A vector database – such as Pinecone, Weaviate, pgvector, or Azure AI Search – stores and retrieves vector representations of text. It is one component in a RAG system, not the entire system. A RAG application also requires document ingestion, parsing, chunking, embedding, context assembly, and generation logic. Frameworks such as LlamaIndex and LangChain provide tooling across all of these stages; the vector database provides only the storage and similarity search layer.

What Determines Retrieval Quality

The quality of a RAG system is determined by the quality of the source documents, the chunking strategy, the embedding model, the retrieval configuration, the context assembly logic, and the consistency of evaluation against real user queries. Switching vector databases or embedding models rarely fixes a retrieval problem caused by poor chunking or stale source documents. Evaluate each stage independently before attributing failure to any single component.

FDE Use Case: An enterprise knowledge assistant must answer employee questions using a controlled, permissions-governed collection of company documents – returning accurate, attributed answers with references to the specific source sections.

Testing, Evaluation, and Observability

Make Framework-Based Solutions Reliable

A successful prototype demonstrates that an idea is feasible. It does not demonstrate that a solution is reliable, accurate, secure, or maintainable at production scale. The gap between a prototype and a production-grade application is where most FDE projects encounter their most significant challenges. Closing that gap requires a systematic approach to quality across four areas.

1. Functional Testing

Test API endpoints, integrations, tool calls, and expected workflow paths. Validate that permission controls, authentication, and error handling behave correctly under normal and edge-case conditions. Automate regression tests so that changes in one component do not silently break another.

2. AI Evaluation

Measure answer accuracy, relevance, groundedness, and retrieval quality using a representative set of test queries. Evaluate edge cases, adversarial inputs, and scenarios where the model should acknowledge insufficient evidence. Track evaluation metrics across model or configuration changes.

3. Observability

Capture errors, latency, model usage, tool execution results, and retrieval outcomes in a structured logging system. Trace the sequence of operations leading to a failure so that root cause analysis is tractable. Monitor resource consumption and application health in production.

4. Safety and Governance

Restrict tool permissions to the minimum required scope. Validate model outputs before they trigger consequential actions. Require human approval for high-impact operations. Protect sensitive information in prompts, logs, and stored outputs in compliance with the customer's data governance requirements.

**Build
Prototype**

Test & Validate



**Evaluate
& Improve**

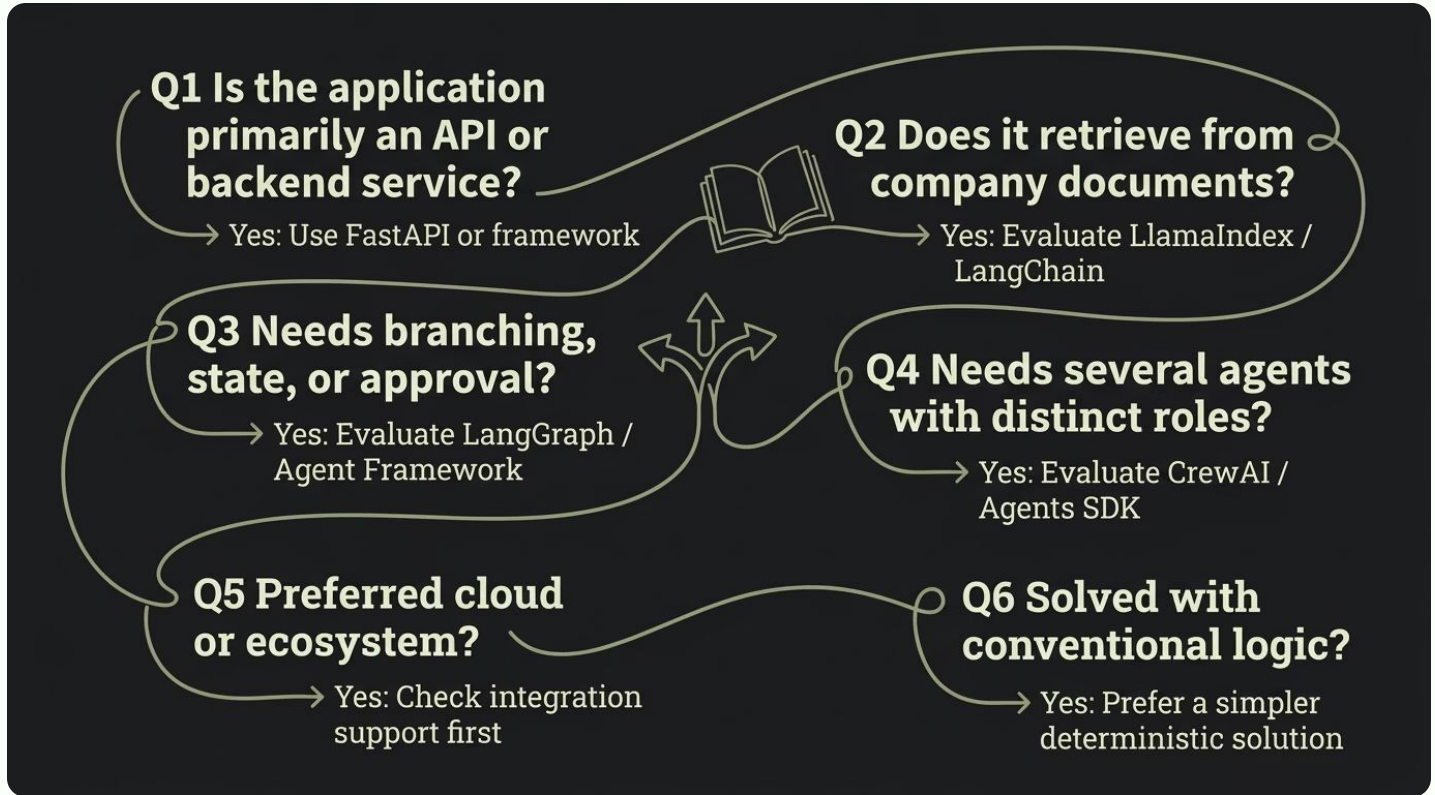
**Deploy &
Observe**

FDE Example: If an AI assistant starts returning incorrect answers, investigate the source documents, retrieval results, prompt construction, model output, and integration logic systematically – in that order – before concluding that the architecture itself needs to change.

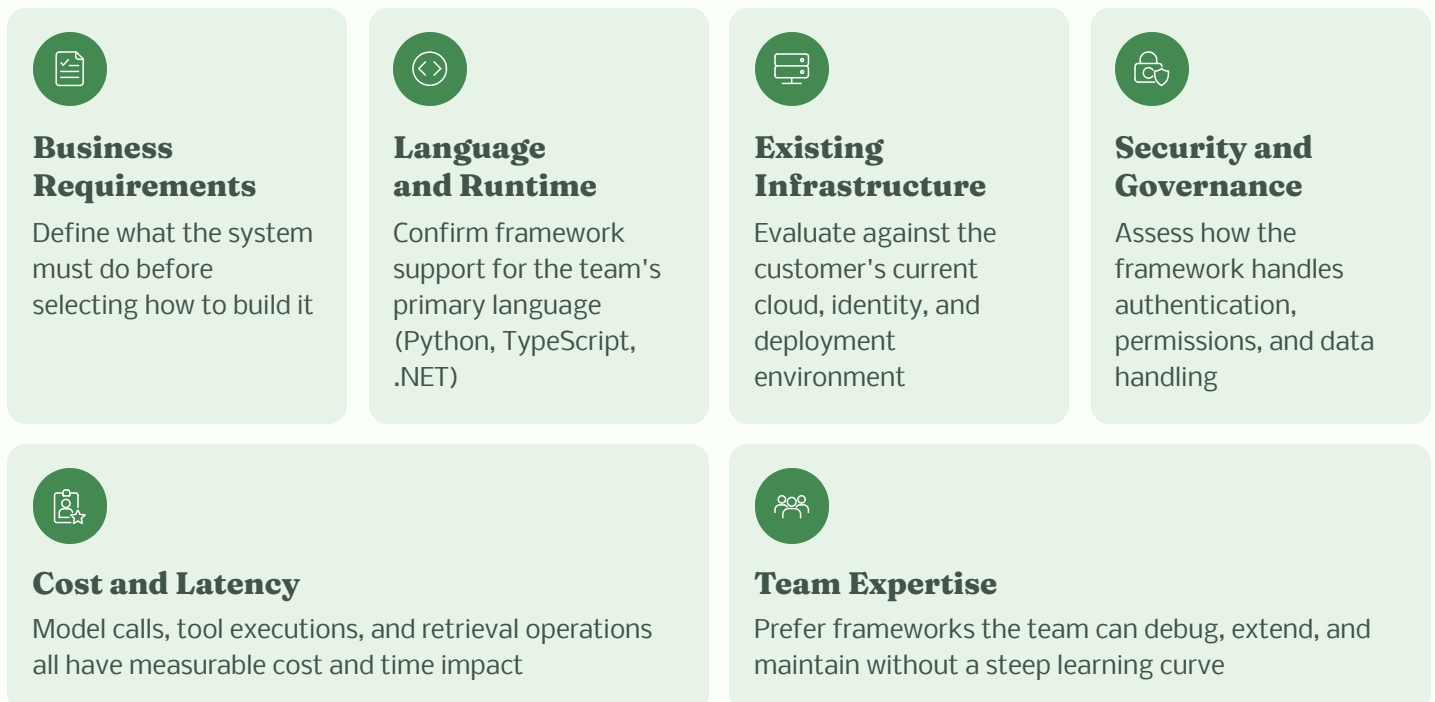
How to Choose the Right Framework

Select Based on the Problem, Not Popularity

Framework selection is an architectural decision, not a preference. The right framework is the one that most directly addresses the customer's technical requirements within the constraints of their infrastructure, team, and timeline. Work through these questions before committing to a specific tool.



Selection Checklist



Practical FDE Project: Enterprise Knowledge Assistant

Apply the Frameworks to a Realistic Business Problem

Scenario: A company has hundreds of internal documents covering policies, technical procedures, and product information. Employees spend significant time searching through repositories manually – and frequently cannot locate authoritative answers. The business goal is an assistant that retrieves relevant documents and answers questions with grounded, attributed responses.



Step 1: Understand the Problem

Interview users to understand what questions they ask most frequently and what a useful answer looks like. Identify the authoritative document sources, access restrictions, and acceptable response latency.



Step 2: Prepare the Data

Connect approved document sources – SharePoint, Confluence, file systems – parse files, split text into meaningful chunks, generate embeddings, and build a searchable index. Apply document-level access controls.



Step 3: Build Retrieval

Configure semantic retrieval to return the most relevant passages for each query. Validate that access restrictions are enforced at retrieval time, not only at ingestion.



Step 4: Generate Answers

Supply retrieved context to the LLM with instructions to ground its response in the provided passages and to explicitly acknowledge when the available evidence is insufficient.



Step 5: Expose the Application

Use FastAPI to provide an authenticated HTTP endpoint that the client application calls. Validate inputs, enforce rate limits, and return structured JSON responses with source attribution.



Step 6: Test and Deploy

Evaluate answer accuracy, source attribution, access controls, latency under realistic load, and failure behaviour. Deploy with logging and monitoring. Establish a regular evaluation cycle.



Suggested Stack: LlamaIndex or LangChain for retrieval components → appropriate LLM provider → vector or hybrid search store → FastAPI for the HTTP API. Add LangGraph only if the workflow requires branching logic or approval steps beyond simple retrieval and generation.

Key Takeaways and Next Steps

Build Solutions That Work Beyond the Prototype

1 Choose Frameworks for the Problem

Select frameworks according to business and technical requirements – not popularity, novelty, or what was used on a previous project. The simplest architecture that reliably solves the problem is the right architecture.

2 Understand the Distinctions

AI orchestration frameworks, application frameworks, retrieval frameworks, protocols, and infrastructure components each occupy a different layer of the stack. Conflating them leads to poor architectural decisions and difficult-to-debug systems.

3 Prefer Simplicity

Use the simplest architecture that reliably meets the requirements. Add agents, graph workflows, or multi-step orchestration only when the problem genuinely requires them – not as default design choices.

4 Design for Production from Day One

Security, evaluation, observability, and error handling are not features to add later. Design them into the application architecture from the beginning of the project.

5 Validate in the Real Environment

Validate the solution against the customer's real workflow, data, users, and infrastructure before scaling or handing over ownership. A prototype that works in isolation is not a production application.

Forward Deployed Engineers do more than select technologies. They connect customer needs with practical engineering decisions, integrate solutions into existing systems, and ensure those solutions deliver measurable business value – from first prototype to production deployment.

Ready to Develop Your Forward Deployed Engineering Skills?

Explore the **GSDC Certified Forward Deployed Engineer Certification** to learn how it relates to building and implementing solutions for real-world business challenges.

[Explore the GSDC FDE Certification](#)

Certified Forward Deployed Engineer



EBOOK

Extensive and exclusive Ebook created by world's experts to help you with understanding core concepts.



LEARNING MATERIALS

Get access to learning materials such as videos, ebooks, templates, and practice exams, which will help you clear the certification exam.



CREATED BY EXPERTS

GSDC certifications are created and authored by world's leading experts in the field.

LEARNING OBJECTIVE

- Gain a strong understanding of Forward Deployed Engineering concepts, roles, tools, technologies, and development practices.
- Identify business and customer challenges and design AI-driven solutions aligned with real-world requirements.
- Apply modern AI frameworks, APIs, data integration, and deployment practices to build and implement solutions within existing business systems.
- Apply testing, evaluation, security, monitoring, and optimization practices to deliver reliable, scalable, and production-ready AI solutions.

Enroll now with the code **LEARN20** To avail **20%** discount

Enroll Now

www.gsdouncil.org 