User: fileNode DevGenAIQuiz_Day1.pdf User: For each answer in this quiz write a one sentence explanation as to why the answer is correct or wrong Model: # Quiz Answer Explanations

Module 1

Q1: What is the main step after tokenization and encoding when a foundation model is processing text?

- ❌ **Categorization** – Categorization is a downstream task, not a core step in the model's internal text processing pipeline.
- ✅ **Embedding** – After tokens are encoded, they are converted into dense vector representations (embeddings) that capture semantic meaning.
- ❌ **Decoupling** – Decoupling is not a standard step in the foundation model text processing pipeline.
- ❌ **Correlation** – Correlation is not a recognized step in the tokenization-to-inference processing flow.

Q2: In which layer of the AWS generative AI stack is Amazon Bedrock categorized?

- ❌ **Applications to boost productivity** – This layer refers to end-user applications like Amazon Q, not the developer platform.
- ✅ **Models and tools to build generative AI applications** – Amazon Bedrock is a managed service providing access to foundation models and tools for building generative AI apps.
- ❌ **Infrastructure to build and train AI models** – This layer refers to compute services like Amazon EC2 and AWS Trainium, not Bedrock.
- ❌ **High-performance compute** – This describes hardware/infrastructure, not the middle tools-and-models layer where Bedrock sits.

Q3: Which optional application component could be used to make sure that the foundation model being used can appreciate context from previous interactions?

- ❌ **Enterprise datasets** – Enterprise datasets provide domain knowledge but don't track prior conversation context.
- ❌ **Vector stores** – Vector stores enable semantic search over documents but don't inherently manage conversation history.
- ✅ **Memory** – Memory components store and retrieve previous interactions so the model can maintain conversational context.
- ❌ **Agents** – Agents orchestrate actions and tool use but are not specifically designed to retain conversation history.

Q4: Foundation models are pre-trained on which type of data?

- ❌ **Labeled data only** – Labeled data is used in supervised learning, whereas foundation models primarily use self-supervised learning on unlabeled data.
- ✅ **A broad spectrum of generalized and unlabeled data** – Foundation models are trained on massive, diverse, unlabeled datasets from the internet and other sources.
- ❌ **Only structured data** – Foundation models ingest unstructured data like text and images, not just structured/tabular data.
- ❌ **Only image data** – Foundation models can be trained on text, images, audio, and other modalities, not just images.

Q5: Which of the following is NOT listed as a modality supported by foundation models?

- ❌ **Text** – Text is one of the primary modalities supported by foundation models.
- ❌ **Images** – Image generation and understanding is a well-established foundation model modality.
- ❌ **Audio** – Audio processing (speech-to-text, music generation) is a supported modality.
- ✅ **Quantum data** – Quantum data is not a recognized modality for current foundation models.

Q6: What are the three key characteristics of context in a foundation model interaction?

- ❌ **Latency, scalability, and throughput** – These are infrastructure performance metrics, not characteristics of context quality.
- ❌ **Creativity, temperature, and randomness** – These relate to inference parameters that control output generation, not context quality.
- ✅ **Relevance, accuracy, and completeness** – Good context must be relevant to the query, factually accurate, and complete enough to support a quality response.
- ❌ **Token limits, model size, and inference cost** – These are technical constraints, not characteristics that define context quality.

Module 2

Q1: A developer is building a monetized coding assistant powered by a large language model. Which inference parameter setting would produce the most precise and deterministic responses?

- ❌ **top_p equals 0.9** – A top_p of 0.9 allows a wide range of tokens, producing more varied and less deterministic outputs.
- ✅ **top_p equals 0.2** – A low top_p value restricts sampling to only the most probable tokens, yielding precise and deterministic code suggestions.
- ❌ **temperature equals 1.5** – A temperature above 1.0 increases randomness significantly, which is undesirable for a coding assistant.
- ❌ **top_k equals 200** – A top_k of 200 considers many candidate tokens, leading to diverse rather than precise outputs.

Q2: True or False – AWS Management Console and AWS SDK for Python SDK (Boto3) use different APIs to interact with Amazon Bedrock.

- ❌ **TRUE** – This is incorrect because both the Console and Boto3 ultimately call the same underlying Amazon Bedrock APIs.
- ✅ **FALSE** – Both the AWS Management Console and Boto3 SDK use the same Amazon Bedrock API endpoints under the hood.

Q3: A developer is creating an AI assistant with low time-to-first-token latency and the flexibility to swap large language models without code changes. Which API should they use?

- ❌ **InvokeModel** – InvokeModel returns the full response at once without streaming and uses model-specific request formats.
- ❌ **InvokeModelWithResponseStream** – While this streams responses, it uses model-specific formats and doesn't provide model-agnostic flexibility.
- ❌ **Converse** – Converse provides a unified interface across models but returns the complete response without streaming.
- ✅ **ConverseStream** – ConverseStream combines streaming (low time-to-first-token) with the model-agnostic Converse API format.

Q4: Which primary inference parameter range is used to control how factual or creative the model responses are?

- ❌ **top_p range of 0 to 1** – top_p controls nucleus sampling probability mass but is not the primary creativity/factuality control.
- ❌ **top_k range of 1 to n** – top_k limits the number of candidate tokens considered but is secondary to temperature for controlling creativity.
- ✅ **temperature range of 0 to 2** – Temperature is the primary parameter that scales the probability distribution, directly controlling the factual-to-creative spectrum.
- ❌ **max_tokens range of 1 to n** – max_tokens controls response length, not the factual or creative nature of the output.

Q5: What is a key benefit of using the Converse API compared to model-specific APIs?

- ❌ **It requires different parameters for each model** – This describes the opposite of Converse's benefit; model-specific APIs require different parameters.
- ✅ **It provides a consistent request and response structure across models** – The Converse API abstracts away model-specific formatting, offering a unified interface.
- ❌ **It only works with Amazon models** – The Converse API works with multiple model providers available through Amazon Bedrock, not just Amazon's own models.
- ❌ **It requires manual prompt format translation** – The Converse API eliminates the need for manual prompt format translation between different models.

Q6: Which AWS SDK for Python feature is used to instantiate a Bedrock client for programmatic interaction?

- ❌ **bedrock.connection** – This is not a valid Boto3 method or class for creating a Bedrock client.
- ✅ **boto3.client("bedrock-runtime")** – This is the standard Boto3 pattern to create a service client for the Amazon Bedrock runtime.
- ❌ **bedrock.initialize** – This is not a recognized Boto3 method for setting up Bedrock access.
- ❌ **bedrock.setup** – This is not a valid Boto3 function; Boto3 uses the client/resource pattern for service instantiation.

Module 3

Q1: In which prompting technique are instructions given to the LLM without giving the model examples?

- ✅ **Zero-shot prompting** – Zero-shot prompting provides only instructions without any examples, relying on the model's pre-trained knowledge.
- ❌ **Few-shot prompting** – Few-shot prompting explicitly includes examples in the prompt to guide the model's behavior.
- ❌ **Chain-of-thought (CoT) prompting** – CoT prompting focuses on eliciting step-by-step reasoning, typically with examples of reasoning chains.
- ❌ **Tree-of-Thought** – Tree-of-Thought explores multiple reasoning paths simultaneously and is not about withholding examples.

Q2: What is NOT one of the main elements of a well-crafted prompt?

- ❌ **Instructions** – Instructions are a core element that tell the model what task to perform.
- ❌ **Context** – Context provides background information the model needs to generate a relevant response.
- ✅ **Training data** – Training data is used during model pre-training, not as an element within a prompt.
- ❌ **Output indicator** – Output indicators specify the desired format or structure of the model's response.

Q3: Which advanced prompting technique involves retrieving relevant documents from a large body of data to provide context?

- ❌ **Self-consistency** – Self-consistency generates multiple reasoning paths and selects the most common answer, without external retrieval.
- ❌ **Tree of thoughts (ToT)** – ToT explores branching reasoning paths but doesn't involve retrieving external documents.
- ✅ **Retrieval Augmented Generation (RAG)** – RAG retrieves relevant documents from external sources and injects them into the prompt as context.
- ❌ **ReAct prompting** – ReAct combines reasoning and acting with tool use but is not specifically about document retrieval for context.

Q4: What is the primary goal of prompt engineering according to the module?

- ❌ **To retrain model weights** – Prompt engineering works at inference time and does not modify the model's

internal weights.

- ✅ **To optimize model output without modifying model parameters** – Prompt engineering crafts inputs to elicit better responses without changing the model itself.
- ❌ **To replace fine-tuning entirely** – Prompt engineering complements fine-tuning but cannot fully replace it for all use cases.
- ❌ **To limit model responses** – The goal is to optimize and improve responses, not merely restrict them.

Q5: Which prompting technique improves performance on complex reasoning tasks by breaking them into intermediate steps?

- ❌ **Zero-shot prompting** – Zero-shot provides instructions without examples and doesn't specifically guide step-by-step reasoning.
- ❌ **Few-shot prompting** – Few-shot provides examples but doesn't necessarily decompose problems into intermediate reasoning steps.
- ✅ **Chain-of-thought (CoT) prompting** – CoT explicitly encourages the model to show intermediate reasoning steps before reaching a final answer.
- ❌ **Instruction fine-tuning** – Instruction fine-tuning modifies model weights and is not a prompting technique.

Q6: Which is a key benefit of using prompt templates as described in the module?

- ❌ **They eliminate the need for prompt testing** – Prompt templates still need testing and iteration to ensure quality outputs.
- ✅ **They improve consistency and scalability** – Templates provide standardized prompt structures that can be reused across applications and inputs.
- ❌ **They automatically fine-tune models** – Templates format prompts at inference time and have no effect on model weights.
- ❌ **They only work with a single model** – Prompt templates can be designed to work across multiple models, especially with the Converse API.

Module 4

Q1: Which benefit can be realized by using Amazon Bedrock batch inference?

- ❌ **Faster real-time responses** – Batch inference processes jobs asynchronously, not in real-time, so it doesn't speed up live responses.
- ✅ **Reduced cost** – Batch inference offers up to 50% cost savings compared to on-demand pricing by processing large volumes offline.
- ❌ **Direct communication of prompts to Amazon Bedrock** – This describes general API access, not a unique benefit of batch inference.
- ❌ **Use of on-demand pricing** – Batch inference uses its own discounted pricing model, not standard on-demand rates.

Q2: What is the name of the process of obtaining summarization information for large datasets in smaller segments?

- ❌ **Batching** – Batching groups multiple requests together for processing but doesn't describe breaking content into smaller pieces.
- ✅ **Chunking** – Chunking divides large documents into smaller, manageable segments for processing tasks like summarization.
- ❌ **Sharding** – Sharding is a database partitioning technique for distributing data across multiple servers.
- ❌ **Partitioning** – Partitioning is a general data management term that doesn't specifically describe the text segmentation process for summarization.

Q3: Which generative AI application component is responsible for implementing long-term memory and/or prompt history?

history?

- ❌ **Foundation model** – The foundation model processes inputs but doesn't inherently manage persistent memory or history storage.
- ❌ **Amazon Bedrock InvokeModel or Converse APIs** – These APIs handle inference requests but don't manage conversation history.
- ✅ **Frontend or backend** – The application's frontend or backend code is responsible for storing, managing, and injecting conversation history.
- ❌ **End user** – The end user interacts with the system but doesn't implement the memory management components.

Q4: What is the primary purpose of Amazon Bedrock prompt caching?

- ✅ **To reduce cost and latency for repeated prompts** – Prompt caching stores frequently used prompt contexts to avoid reprocessing, saving both time and money.
- ❌ **To increase model throughput** – While caching may indirectly improve throughput, its primary purpose is cost and latency reduction.
- ❌ **To store model weights** – Model weights are stored by the service infrastructure, not by the prompt caching feature.
- ❌ **To improve accuracy** – Prompt caching doesn't change model outputs; it optimizes the efficiency of processing repeated prompts.

Q5: Which inference mode in Amazon Bedrock provides guaranteed throughput for large consistent inference workloads?

- ❌ **Batch inference mode** – Batch inference processes jobs asynchronously but doesn't guarantee real-time throughput levels.
- ❌ **On-demand inference mode** – On-demand mode provides pay-per-use access without throughput guarantees.
- ✅ **Provisioned throughput mode** – Provisioned throughput reserves dedicated capacity, guaranteeing consistent performance for sustained workloads.
- ❌ **All of the above** – Only provisioned throughput provides guaranteed throughput; the other modes do not.

Q6: In the text summarization workflow, which AWS service is used to process and store the results of batch inference jobs?

- ✅ **Amazon DynamoDB** – DynamoDB is used to store and manage the processed summarization results for efficient retrieval.
- ❌ **Amazon S3** – While S3 stores the raw batch inference output files, DynamoDB is used for processing and storing the structured results.
- ❌ **Amazon SQS** – SQS is a message queuing service, not a data storage service for inference results.
- ❌ **Amazon Lambda** – Lambda is a compute service for running code, not a storage service for inference results.

Module 5

Q1: What is the flow for hydrating a knowledge base in a RAG solution?

- ❌ **Enterprise data to embeddings to chunks to storage** – This order is incorrect because data must be chunked before embeddings are generated.
- ✅ **Enterprise data to chunks to embeddings to storage** – Data is first chunked into smaller pieces, then converted to embeddings, and finally stored in a vector store.
- ❌ **User requests to embeddings to populate knowledge base** – Knowledge bases are hydrated from enterprise data sources, not from user requests.
- ❌ **User requests to chunks to embeddings to storage** – The hydration process uses enterprise data, not user queries, as the source.

Q2: Which statements correctly describe Amazon Bedrock RetrieveAndGenerate and Retrieve methods? (Select TWO)

Q2: Which statements correctly describe Amazon Bedrock RetrieveAndGenerate and Retrieve methods? (Select TWO)

- ❌ **Retrieve produces FM-generated responses** – Retrieve only returns relevant chunks; it does not invoke a foundation model to generate responses.
- ✅ **Retrieve gets chunks relevant to the query and returns them as an array** – The Retrieve API performs semantic search and returns matching document chunks.
- ✅ **RetrieveAndGenerate fully manages the RAG pipeline end to end** – RetrieveAndGenerate handles retrieval, context injection, and response generation in a single call.
- ❌ **RetrieveAndGenerate produces contexts** – RetrieveAndGenerate produces complete FM-generated responses, not just contexts.

Q3: What are common enterprise use cases for RAG? (Select TWO)

- ✅ **Building powerful question-answering systems** – RAG excels at Q&A by grounding model responses in retrieved enterprise knowledge.
- ❌ **Translating videos from one language to another** – Video translation is a multimodal task not typically associated with RAG.
- ✅ **Enhancing content generation like writing articles and reports** – RAG improves content generation by providing relevant source material as context.
- ❌ **Converting marketing posts to speech without context** – Text-to-speech conversion doesn't require the retrieval-augmentation that defines RAG.

Q4: Which type of knowledge base store is best for general unstructured text with semantic similarity-based retrieval?

- ❌ **Graph database** – Graph databases excel at relationship-based queries but are not optimized for semantic similarity search on unstructured text.
- ✅ **Vector store** – Vector stores are purpose-built for storing embeddings and performing semantic similarity searches on unstructured text.
- ❌ **Hybrid store** – Hybrid stores combine multiple search methods but vector stores are the best fit for purely semantic similarity retrieval.
- ❌ **Relational database** – Relational databases are designed for structured data and lack native semantic similarity search capabilities.

Q5: What is a key consideration when choosing chunk size for RAG?

- ❌ **Smaller chunks provide higher level of context** – Smaller chunks actually provide less context per chunk, not more.
- ✅ **Smaller chunks provide higher retrieval speed** – Smaller chunks are faster to search and match, improving retrieval performance.
- ❌ **Larger chunks provide lower level of context** – Larger chunks actually contain more context, not less.
- ❌ **Chunk size does not affect performance** – Chunk size significantly impacts both retrieval accuracy and processing speed.

Q6: Which Amazon Bedrock API joins Retrieve with InvokeModel to provide the full RAG workflow?

- ❌ **Retrieve** – Retrieve only handles the search/retrieval portion and does not invoke a model for generation.
- ❌ **GenerateQuery** – GenerateQuery is not the API that combines retrieval and generation in Bedrock's RAG workflow.
- ✅ **RetrieveAndGenerate** – RetrieveAndGenerate combines document retrieval with model invocation to deliver the complete RAG pipeline.
- ❌ **StartIngestionJob** – StartIngestionJob is used to load data into a knowledge base, not to perform retrieval and generation.

Module 6

Q1: Which Amazon DynamoDB table is used for storing chat history when using LangChain with DynamoDB for conversation memory?

- ❌ **User table** – A user table would store user profiles, not conversation history.
- ❌ **Conversation table** – While intuitive, LangChain's DynamoDB integration uses a session-based table structure.
- ✅ **Session table** – LangChain's DynamoDB chat history integration uses a session table keyed by session ID to store messages.
- ❌ **Message table** – Individual messages are stored as items within the session table, not in a separate message table.

Q2: What is a key advantage of using open source frameworks like LangChain with Amazon Bedrock compared to using Bedrock's APIs directly?

- ❌ **Open source frameworks provide faster performance and lower latency than Bedrock's built-in APIs** – Frameworks add abstraction layers that may introduce slight overhead, not performance improvements.
- ✅ **Open source frameworks offer additional abstractions and components** – LangChain provides chains, memory, agents, and other building blocks that simplify complex application development.
- ❌ **Open source frameworks are officially supported by AWS with a guarantee** – Open source frameworks are community-maintained and don't carry AWS support guarantees.
- ❌ **Open source frameworks can only be used with Amazon Bedrock** – LangChain is model-agnostic and works with many providers beyond Amazon Bedrock.

Q3: Which programming languages is LangChain currently available in?

- ❌ **Python only** – LangChain expanded beyond Python to support JavaScript/TypeScript as well.
- ❌ **Python and JavaScript only** – This omits TypeScript, which is also officially supported.
- ✅ **Python, TypeScript, and JavaScript** – LangChain offers official libraries for Python and JavaScript/TypeScript (LangChain.js).
- ❌ **All major programming languages** – LangChain is not available in languages like Java, Go, or C# as official implementations.

Q4: What is a chain in LangChain?

- ❌ **A security mechanism for API access** – Chains have nothing to do with security or API authentication.
- ✅ **A set of components that run together** – A chain links multiple LangChain components (prompts, models, parsers) into a sequential or conditional workflow.
- ❌ **A type of database connection** – Chains orchestrate LLM workflows, not database connections.
- ❌ **A method for encrypting data** – Chains are workflow orchestration constructs, unrelated to encryption.

Q5: In LangChain, what component is used to define reusable prompt structures?

- ❌ **PromptBuilder** – PromptBuilder is not a standard LangChain class name.
- ✅ **ChatPromptTemplate** – ChatPromptTemplate is the LangChain class used to create reusable, parameterized prompt templates for chat models.
- ❌ **MessageFormatter** – MessageFormatter is not a recognized LangChain component for prompt templating.
- ❌ **TemplateEngine** – TemplateEngine is not part of LangChain's API for defining prompt structures.

Q6: What are key LangChain integration components? (Select THREE)

- ✅ **Models** – Models are a core LangChain integration component for connecting to various LLM providers.
- ✅ **Prompt templates** – Prompt templates are a fundamental LangChain component for structuring inputs to models.
- ✅ **Indexes** – Indexes (including vector store integrations and retrievers) are key LangChain components for document retrieval.
- ❌ **Databases** – While LangChain can connect to databases, "Databases" is not listed as one of the key integration component categories.

Module 7

Q1: Which RAG metric is used for evaluating the generation stage?

- ❌ **Context recall** – Context recall evaluates the retrieval stage by measuring how well retrieved contexts cover the needed information.
- ❌ **Context precision** – Context precision evaluates the retrieval stage by measuring the relevance of retrieved contexts.
- ❌ **Context entity recall** – Context entity recall evaluates retrieval by checking if key entities are present in retrieved contexts.
- ✅ **Faithfulness** – Faithfulness evaluates the generation stage by measuring whether the generated response is factually consistent with the retrieved contexts.

Q2: What RAG evaluation metric uses an LLM-generated response to create artificial questions and measure similarity with the original query?

- ❌ **Context recall** – Context recall compares retrieved contexts against ground truth, not by generating artificial questions.
- ❌ **Noise sensitivity** – Noise sensitivity measures how robust retrieval is to irrelevant information, not through artificial question generation.
- ✅ **Response relevancy** – Response relevancy generates questions from the response and measures their similarity to the original query to assess relevance.
- ❌ **Faithfulness** – Faithfulness checks factual consistency between the response and context, not through artificial question generation.

Q3: Which mechanism helps lower inference latency by dynamically predicting the response quality?

- ✅ **Intelligent prompt routing** – Intelligent prompt routing dynamically selects the optimal model based on predicted response quality, reducing latency by using simpler models when appropriate.
- ❌ **Prompt caching** – Prompt caching reduces latency for repeated prompts but doesn't predict response quality.
- ❌ **Latency-optimized inference** – This is a general optimization approach, not a mechanism that predicts response quality to route requests.
- ❌ **Prompt catalog** – A prompt catalog stores and manages prompts but doesn't dynamically route or predict quality.

Q4: What are the three main stages in the LLM selection process?

- ✅ **Quick short listing** – Quick short listing is the first stage where candidates are narrowed down based on high-level criteria.
- ✅ **Case-based benchmarking** – Case-based benchmarking is the second stage where models are tested against specific use case requirements.
- ✅ **Priority-based model selection** – Priority-based model selection is the final stage where the best model is chosen based on weighted priorities.
- ❌ **Academic benchmarking** – Academic benchmarking is not listed as one of the three main stages in the LLM selection process.

Q5: What are the three types of model evaluation methods available with Amazon Bedrock? (Pick three)

- ✅ **Programmatic evaluation** – Programmatic evaluation uses automated metrics and code-based assessments within Amazon Bedrock.
- ✅ **Human evaluation** – Human evaluation leverages human reviewers to assess model outputs for quality and appropriateness.
- ✅ **LLM as a judge** – LLM as a judge uses another foundation model to evaluate the quality of a model's responses.

- ❌ **Automated testing** – Automated testing is not listed as one of the three distinct evaluation methods in Amazon Bedrock.

Q6: Which retrieval evaluation metric measures how well the retrieved contexts contain the information needed to answer the query?

- ❌ **Context precision** – Context precision measures the proportion of retrieved contexts that are relevant, not completeness of information.
- ✅ **Context recall** – Context recall measures whether all the necessary information to answer the query is present in the retrieved contexts.
- ❌ **Context entity recall** – Context entity recall specifically checks for entity coverage, which is narrower than overall information completeness.
- ❌ **Response relevancy** – Response relevancy evaluates the generated response's relevance to the query, not the quality of retrieved contexts.

Module 8

Q1: What is responsible AI?

- ✅ **The practice of designing, developing, and using AI technology with the goal of minimizing risks related to user safety** – Responsible AI encompasses ethical design principles focused on safety, fairness, and minimizing potential harms.
- ❌ **The practice of maximizing AI performance regardless of ethical considerations** – Ignoring ethics is the opposite of responsible AI.
- ❌ **The practice of limiting AI use to only large enterprises** – Responsible AI applies to all organizations and use cases, not just large enterprises.
- ❌ **The practice of automating all decision-making** – Full automation without human oversight contradicts responsible AI principles.

Q2: Which of the following is NOT one of the emerging risks and challenges with generative AI mentioned in the module?

- ❌ **Veracity (hallucinations)** – Hallucinations are a well-documented risk where models generate factually incorrect information.
- ❌ **Toxicity and safety** – Toxic or harmful content generation is a recognized challenge with generative AI.
- ❌ **Data privacy** – Data privacy concerns around training data and user inputs are a key challenge in generative AI.
- ✅ **High computational costs** – While a practical concern, high computational costs are not listed as an emerging ethical risk/challenge in the module.

Q3: What is an effective technique for mitigating bias in AI systems?

- ❌ **Removing all demographic data from training datasets** – Removing all demographic data can actually worsen bias by preventing the model from learning fair representations.
- ✅ **Enhancing datasets with more representative samples** – Adding diverse, representative data helps the model learn balanced patterns and reduces systematic bias.
- ❌ **Limiting AI systems to only factual information** – Restricting to facts doesn't address bias that can exist within factual but unrepresentative data.
- ❌ **Testing models only with ideal use cases** – Testing only ideal cases masks bias that would appear in real-world, diverse scenarios.

Q4: A financial services company needs its AI-assisted loan approval system to provide decision details to meet regulatory requirements. Which responsible AI principle is MOST relevant?

- ❌ **Transparency** – Transparency is about openness regarding AI use, but the need to explain specific decisions points to explainability

points to explainability.

- ✅ **Explainability** – Explainability requires the system to provide clear, understandable reasons for individual decisions, which is what regulators demand.
- ❌ **Fairness** – Fairness ensures equitable treatment across groups but doesn't specifically address the requirement to explain individual decisions.
- ❌ **Governance** – Governance covers overall AI management policies but doesn't specifically address providing decision-level explanations.

Q5: What does the contextual grounding check in Amazon Bedrock Guardrails help reduce?

- ✅ **Hallucinations** – The contextual grounding check verifies that model responses are grounded in the provided context, directly reducing hallucinated content.
- ❌ **Training time** – Guardrails operate at inference time and have no effect on model training duration.
- ❌ **Model size** – Contextual grounding checks don't affect the size or architecture of the model.
- ❌ **API costs** – While reducing hallucinations improves quality, the grounding check itself doesn't directly lower API costs.

Q6: Which scenario would be BEST handled by deterministic controls in a generative AI application?

- ❌ **Analyzing the emotional tone of user-generated content to detect potential harassment** – Emotional tone analysis requires nuanced judgment that probabilistic AI models handle better than deterministic rules.
- ✅ **Verifying that a user has permission to access sensitive customer data before sending it to the model** – Permission verification is a binary, rule-based check perfectly suited to deterministic controls.
- ❌ **Evaluating whether a model's response contains subtle bias or insensitivity** – Detecting subtle bias requires nuanced analysis that goes beyond deterministic rule-based checks.
- ❌ **Detecting complex phishing attempts in natural language** – Complex phishing detection requires pattern recognition and contextual understanding beyond simple deterministic rules.

Module 9

Q1: Which component allows AI agents to act based on their reasoning generated by a large language model (LLM)?

- ❌ **Memory** – Memory stores past interactions and state but doesn't enable the agent to take actions.
- ❌ **Guardrails** – Guardrails constrain and filter agent behavior but don't enable action-taking.
- ✅ **Tools** – Tools (APIs, functions, services) are the components that agents invoke to execute actions based on LLM reasoning.
- ❌ **Evaluation** – Evaluation assesses agent performance but doesn't provide the capability to act.

Q2: Which agentic framework offers model-driven development designed to simplify and accelerate the construction of AI agents?

- ❌ **LangChain** – LangChain is a general-purpose LLM application framework, not specifically designed as a model-driven agent builder.
- ❌ **CrewAI** – CrewAI focuses on multi-agent role-based collaboration, not model-driven simplicity.
- ❌ **LangGraph** – LangGraph provides graph-based agent workflows but is focused on stateful orchestration rather than model-driven simplicity.
- ✅ **Strands** – Strands (from AWS) offers a model-driven development approach designed to simplify and accelerate AI agent construction.

Q3: A developer needs to set up unified access to tools for their AI agent to avoid building custom APIs to access them. Which should they use?

- ✅ **Model Context Protocol (MCP)** – MCP provides a standardized protocol for connecting AI agents to diverse tools and data sources through a unified interface.
- ❌ **Amazon Agent [illegible]** – [illegible]

- ❌ **Agent-to-Agent (A2A) Protocol** – A2A is designed for inter-agent communication, not for unifying tool access.
- ❌ **CrewAI** – CrewAI is an agent framework, not a protocol for standardizing tool access.
- ❌ **Strands** – Strands is an agent development framework, not a tool access standardization protocol.

Q4: What is the primary purpose of the ReAct framework in AI agents?

- ❌ **To store conversation history** – ReAct is a reasoning-action framework, not a memory management system.
- ✅ **To combine reasoning and acting in iterative loops** – ReAct alternates between reasoning (thinking through a problem) and acting (taking actions), iterating until the task is complete.
- ❌ **To provide guardrails for AI responses** – ReAct focuses on reasoning and action, not on constraining or filtering responses.
- ❌ **To manage multiple agent collaboration** – ReAct operates within a single agent's reasoning loop, not for multi-agent coordination.

Q5: Which protocol is best suited for enabling multiple agents to interact with each other in a peer-to-peer fashion?

- ❌ **Model Context Protocol (MCP)** – MCP standardizes tool and data access for agents, not inter-agent communication.
- ✅ **Agent-to-Agent (A2A) Protocol** – A2A is specifically designed to enable direct peer-to-peer communication and collaboration between multiple agents.
- ❌ **HTTP** – HTTP is a general transport protocol that doesn't provide the agent-specific semantics needed for peer-to-peer agent interaction.
- ❌ **JSON-RPC** – JSON-RPC is a remote procedure call protocol that lacks the agent-specific coordination capabilities of A2A.

Q6: What is the role of an MCP server in the Model Context Protocol architecture?

- ❌ **It hosts the large language model that powers the agent** – The LLM is hosted separately; the MCP server provides tool access, not model hosting.
- ✅ **It provides a unified API that abstracts access to multiple tools and resources** – The MCP server exposes tools, data, and resources through a standardized interface for agents to consume.
- ❌ **It stores conversation history and agent memory** – Memory management is handled by other components, not the MCP server.
- ❌ **It manages peer-to-peer communication between agents** – Peer-to-peer agent communication is handled by A2A, not MCP.

Module 10

Q1: Which benefit does Amazon Bedrock Agents provide that neither inline agents nor AgentCore provide?

- ❌ **Visual builder, versions, and prompt management** – While Bedrock Agents offers a visual builder, the distinguishing feature asked about is operational management.
- ✅ **PrepareAgent operations and alias management** – PrepareAgent operations and alias management are unique to Amazon Bedrock Agents, enabling versioned deployments and controlled rollouts.
- ❌ **On-the-fly definition at each agent invocation** – On-the-fly definition is a feature of Inline Agents, not Bedrock Agents.
- ❌ **Modular building blocks for agents** – Modular building blocks describe AgentCore's approach, not Bedrock Agents' unique benefit.

Q2: What is a key feature of Amazon Bedrock AgentCore?

- ❌ **Visual builder, versions, and prompt management** – Visual builder and prompt management are features of Amazon Bedrock Agents, not AgentCore.
- ❌ **PrepareAgent operations and alias management** – PrepareAgent and alias management are specific to Amazon

Bedrock Agents.

- ❌ **On-the-fly definition at each agent invocation** – On-the-fly definition describes Inline Agents' capability.
- ✅ **Modular building blocks for agents** – AgentCore provides modular, composable infrastructure building blocks that developers can assemble for custom agent architectures.

Q3: Which characteristics describe the relationship between Amazon Bedrock Flows, Agents, Inline Agents, and AgentCore?

- ❌ **Developers must choose one to use throughout their entire application** – These constructs are designed to be complementary, not mutually exclusive.
- ✅ **Developers can mix and match any of these constructs for various components** – AWS designed these services to be composable, allowing developers to use different constructs for different parts of their application.
- ❌ **Flows, Agents, and Inline Agents can be used together but not with AgentCore** – AgentCore can be combined with the other constructs as well.
- ❌ **Inline Agents and AgentCore can be used together but not with Flows or Agents** – All four constructs are interoperable and can be mixed freely.

Q4: What is the primary purpose of Amazon Bedrock Flows?

- ❌ **To create autonomous AI agents with reasoning capabilities** – Autonomous agents are the purpose of Bedrock Agents, not Flows.
- ✅ **To build and scale controllable generative AI workflows** – Bedrock Flows provides a visual and programmatic way to create structured, controllable multi-step generative AI workflows.
- ❌ **To provide dynamic agent configuration at runtime** – Dynamic runtime configuration is a feature of Inline Agents, not Flows.
- ❌ **To enable multi-agent collaboration** – Multi-agent collaboration is handled through agent orchestration patterns, not Flows specifically.

Q5: Which Amazon Bedrock AgentCore component provides secure cloud-based browser interaction for AI agents?

- ❌ **AgentCore Runtime** – AgentCore Runtime provides the execution environment for agents, not browser interaction capabilities.
- ❌ **AgentCore Identity** – AgentCore Identity manages authentication and authorization, not browser interaction.
- ✅ **AgentCore Browser** – AgentCore Browser provides secure, cloud-based browser interaction capabilities that allow AI agents to navigate and interact with web content.
- ❌ **AgentCore Gateway** – AgentCore Gateway manages API access and routing, not browser-based interactions.

Q6: What does the supervisor-collaborator multi-agent pattern enable?

- ❌ **Sequential processing of tasks by a single agent** – This describes a simple single-agent workflow, not a multi-agent collaboration pattern.
- ✅ **Hierarchical collaboration with domain specialist sub-agents** – The supervisor-collaborator pattern has a supervisor agent that delegates tasks to specialized sub-agents and synthesizes their results.
- ❌ **Direct peer-to-peer communication between all agents** – Peer-to-peer communication describes a different multi-agent pattern, not the hierarchical supervisor-collaborator model.
- ❌ **Random assignment of tasks to available agents** – The supervisor intelligently delegates tasks based on agent specializations, not randomly.