



Free Microsoft AI-103 Practice Questions

Shared by Sargent on 13-08-2026

For More Free Questions and Preparation Resources

Check the Links on Last Page



Question 1

Question Type: MultipleChoice

Case Study: Mix Questions

Mix Questions

AI-103 Mix Questions IN THIS CASE STUDY

You have a Microsoft Foundry project that contains a high-traffic agent.

After a recent update, operational costs increase significantly.

Monitoring confirms that the volume of user traffic to the agent remains unchanged.

You suspect that changes to the request or response characteristics are causing the increase.

You need to identify whether the additional costs are driven by the model input size, the model output size, or expanded tool usage.

Which observability capability should you use?

Options:

- A- evaluation metrics
- B- token usage
- C- latency
- D- run success rate

Answer:

B

Explanation:

The correct capability is token usage. In Microsoft Foundry observability, token consumption is the primary signal for diagnosing model-cost changes when request volume is unchanged. Token usage lets you distinguish whether costs increased because prompts became larger, retrieved or tool-provided context expanded, responses became longer, or agent execution added more model calls. Microsoft Foundry monitoring dashboards track operational metrics such as token consumption, latency, error rates, and quality scores, and the agent monitoring dashboard is specifically intended to help analyze token usage, latency, success rates, and evaluation outcomes for production traffic.

This directly matches the scenario because the issue is not more traffic, but changed request or response characteristics. Input tokens reveal whether the prompt, chat history, grounding data, or tool outputs being sent to the model increased. Output tokens reveal whether the model is generating longer completions. Expanded tool usage can also increase cost indirectly by adding more tool results, intermediate calls, and context into subsequent model requests; Foundry tracing and observability capture tool usage and token consumption for agent runs.

Evaluation metrics assess response quality and safety, not cost drivers. Latency identifies performance delays, and run success rate measures reliability. Reference topics: Microsoft Foundry observability, agent monitoring dashboard, token consumption, cost analysis, tool usage, and production monitoring.

Question 2

Question Type: MultipleChoice

You have a Microsoft Foundry project named Project1 that contains the following:

- * An OpenAPI tool that calls an external API
- * A project connection named Connection1 that stores the API key of the external API

When an agent calls the OpenAPI tool, the API returns a 401 unauthorized error, and traces show that the API key header is NOT being sent.

You need to ensure that the OpenAPI tool automatically includes the API key from Connection1 on all requests.

What should you do?

Options:

- A- Configure the tool to use the default connection of Project1.
- B- Connect the tool to Connection1.
- C- Enable identity passthrough so that the tool uses the Microsoft Entra token of the caller.
- D- Add the API key header manually to the OpenAPI specification.

Answer:

B

Explanation:

The correct action is to connect the OpenAPI tool to Connection1. In Microsoft Foundry Agent

Service, an OpenAPI tool does not automatically use every project connection in the project. For API key authentication, the tool must be explicitly configured to use the project connection that contains the required secret. Microsoft's OpenAPI tool guidance states that API key or token authentication requires a project connection configured with the API key or token, and the tool is then created or configured to use that connection for authentication. The documentation also notes that the agent calls the external API by using the stored API key when the OpenAPI tool is configured with the project connection.

Option A is incorrect because a project's default connection is not automatically the authentication source for a specific OpenAPI tool. Option C is incorrect because identity passthrough or managed identity only applies when the target API accepts Microsoft Entra ID tokens; this scenario uses an external API key. Option D is also incorrect as a credential value should not be manually embedded in the OpenAPI specification. The specification defines the security scheme, while the secret value is stored in the Foundry connection. Reference topics: OpenAPI tools, project connections, API key authentication, tool authentication configuration, and agent tracing.

Question 3

Question Type: MultipleChoice

Note: This section contains one or more sets of questions with the same scenario and problem. Each question presents a unique solution to the problem. You must determine whether the solution meets the stated goals. More than one solution in the set might solve the problem. It is also possible that none of the solutions in the set solve the problem.

After you answer a question in this section, you will NOT be able to return. As a result, these questions do not appear on the Review Screen.

You have a Microsoft Foundry project that contains an agent. The agent generates summaries from retrieved policy documents.

Users report that some responses omit required regulatory clauses, even when the clauses are present in the retrieved content.

You need to improve response completeness.

Solution: You increase the value of the max_tokens parameter.

Does this meet the goal?

Options:

A- Yes

B- No

Answer:

B

Explanation:

The solution does not meet the goal. Increasing max_tokens only raises the maximum number of tokens the model is allowed to generate. Microsoft's Azure OpenAI reference defines max_tokens as the maximum number of tokens allowed for the generated answer, and the quota guidance notes that increasing it can help when responses are being truncated.

In this scenario, the problem is **not described** as output **truncation**. The required regulatory clauses are already present in the retrieved policy documents, but the agent omits them during summarization. That is a response completeness issue: Microsoft Foundry RAG evaluator guidance defines response completeness as the recall aspect of the response, meaning the response should not miss critical information compared with expected content or ground truth.

A larger token budget might permit a longer answer, but it does not force the model to identify, verify, or include each mandatory clause. It can also increase cost and latency. The appropriate control is a reflection or completeness verification pass that checks the draft against the retrieved policy clauses and regenerates or revises the response when required content is missing. Reference topics: RAG response completeness, model output limits, max_tokens, reflection, and response validation.

Question 4

Question Type: MultipleChoice

You are building a web app named App1 that generates responses by using a model deployed to a Microsoft Foundry project named Project1.

Before sending the prompts to the model, App1 must retrieve documents by using Azure AI Search.

You need to integrate Project1 and App1. The solution must meet the following requirements:

- * Multiple client applications must use the same search configuration.
- * A security policy must prevent key-based authentication.
- * Administrative effort must be minimized.

What should you do?

Options:

- A- Enable a managed identity for each application and call Azure AI Search directly.
- B- Create a custom HTTP connection in Foundry and manually configure Azure AI Search endpoints per application.
- C- Call Azure AI Search directly from each application by using Microsoft Entra authentication.
- D- Configure an Azure AI Search connection in Project1 and reference the connection in each application.

Answer:

D

Explanation:

The correct solution is to configure an Azure AI Search connection in Project1 and reference that connection from each application. Microsoft Foundry project connections are intended to centralize external resource configuration for a project. The official Foundry guidance states that when adding a connection, you select the external service, such as Azure AI Search, and choose the authentication method for that resource. This directly satisfies the requirement for multiple client applications to use the same search configuration instead of duplicating endpoints, indexes, and credentials in each app.

The security requirement is met by configuring the connection with keyless authentication. The Azure AI Search integration guidance specifies that a Foundry project connection requires the search endpoint and either key-based authentication or keyless authentication with Microsoft Entra ID. For keyless authentication, RBAC roles are assigned to the project's managed identity, eliminating hard-coded API keys.

Options A and C use secure identity-based access but force every client application to maintain its own Azure AI Search configuration, increasing administration and violating the shared-configuration requirement. Option B also increases administrative work and does not use the native Azure AI Search connection type. Reference topics: Microsoft Foundry project connections, Azure AI Search grounding, Microsoft Entra ID authentication, managed identities, and RBAC.

Question 5

Question Type: MultipleChoice

You have a chat app in a Microsoft Foundry project and an Azure AI Search vectorized index.

You need to connect to the index to meet the following requirements:

- * Complex questions must retrieve information from multiple chunks.

- * Multi-turn conversations must influence retrieval planning.
- * Retrievals must run in parallel to reduce latency.

Which retrieval approach should you use?

Options:

- A- classic Retrieval Augmented Generation (RAG)
- B- chain of thought
- C- agentic Retrieval Augmented Generation (RAG)
- D- iterative retrieval

Answer:

C

Explanation:

The correct answer is agentic Retrieval Augmented Generation (RAG) because the requirements describe the agentic retrieval pipeline in Azure AI Search. Agentic retrieval is designed for chat and copilot scenarios where a user's request can be complex, conversational, and dependent on prior turns. Azure AI Search agentic retrieval uses an LLM-assisted planning stage to break a complex request into focused subqueries, allowing the system to retrieve grounding information from multiple chunks rather than relying on a single query path. Microsoft's Azure AI Search guidance describes agentic retrieval as a multi-query pipeline for complex questions in chat and agent workflows, with subqueries that can include chat history for additional context.

This also satisfies the latency requirement because agentic retrieval runs the generated subqueries in parallel and then merges and reranks the best results for use by the generative model. Classic RAG is simpler and typically sends a single query to search, making it less suitable for multi-hop or conversational retrieval planning. Chain of thought is a reasoning technique, not an Azure AI Search retrieval approach, and iterative retrieval does not specifically provide the built-in query planning, conversation-aware retrieval, and parallel execution described here. Reference topics: Azure AI Search agentic retrieval, RAG with Azure AI Search, knowledge bases, query planning, and generative AI grounding.

Question 6

Question Type: MultipleChoice

You have a Microsoft Foundry project.

You plan to build a customer support solution that contains an agent. The solution must meet the following requirements:

- * Provide accurate, context-aware responses grounded in internal product documentation stored in Azure AI Search.
- * Require deep, multi-step reasoning across long contexts.
- * Generate detailed natural language responses.

Which type of model should you use to power the agent?

Options:

- A- a multimodal model
- B- a key phrase extraction model
- C- a small language model (SLM)
- D- a large language model (LLM)

Answer:

D

Explanation:

The correct model type is a large language model (LLM). The scenario requires an agent that can reason over retrieved documentation, synthesize context-aware answers, and generate detailed natural language responses. Microsoft Foundry RAG guidance defines Retrieval Augmented Generation as a pattern that combines search with large language models so responses are grounded in organizational data, which directly matches the use of Azure AI Search for internal product documentation.

An LLM is also the correct fit for deep, multi-step reasoning across long contexts. Azure AI Search agentic retrieval guidance states that complex chat and agent scenarios use an LLM to break a user query into smaller focused subqueries for better coverage over indexed content. Microsoft also describes Azure AI Search as a way to ground agents and chatbots in proprietary enterprise data for accurate, context-aware responses.

A multimodal model is unnecessary because the requirements are text-focused, not image, audio, or video based. A key phrase extraction model performs narrow text analytics and cannot generate detailed answers. An SLM may reduce cost and latency, but the stated need for deep reasoning and long-context synthesis favors an LLM. Reference topics: Microsoft Foundry RAG, Azure AI Search grounding, agentic retrieval, and model selection for generative AI agents.

Question 7

Question Type: MultipleChoice

You have a Microsoft Foundry project that contains an agent.

You need to process mixed-format documents that contain scanned text, tables, and multicolumn layouts. The extracted content must preserve the document structure and be converted into the Markdown format for downstream reasoning.

What should you configure first?

Options:

- A- an Azure Content Understanding in Foundry Tools analyzer
- B- a generative chat completion request
- C- an Azure Language in Foundry Tools text analysis model deployment
- D- an Azure OpenAI Responses API call that uses a multimodal model

Answer:

A

Explanation:

The correct answer is A. an Azure Content Understanding in Foundry Tools analyzer. In Azure Content Understanding, an analyzer is the reusable configuration that defines what content type is processed, which elements are extracted, and how the output is structured. Microsoft's analyzer guidance states that analyzers define extraction for text, layout, tables, fields, and output formats such as Markdown and JSON. This makes the analyzer the first required configuration step before downstream reasoning or agent orchestration can use the extracted content.

This matches the requirement because the documents contain scanned text, tables, and multicolumn layouts. Content Understanding document analysis is designed to transform unstructured documents into structured, machine-readable output while preserving document structures and relationships. Its Markdown representation converts unstructured documents into GitHub Flavored Markdown while maintaining content and layout for downstream use.

A generative chat completion request or Azure OpenAI Responses API call could reason over extracted content, but it is not the correct first step for OCR, layout preservation, and Markdown conversion. Azure Language focuses on text analysis after content has already been extracted. Reference topics: Content Understanding analyzers, document extraction, OCR, layout analysis, Markdown output, and downstream reasoning.

Question 8

Question Type: MultipleChoice

You have an app named App1 that uses a Microsoft Foundry multimodal model deployment.

App1 runs optical character recognition (OCR) on uploaded images and appends the OCR output to the prompt as additional context.

Some uploaded images contain embedded text.

You need to prevent potentially malicious instructions from being processed by the model.

What should you use?

Options:

- A- protected material text
- B- prompt shields for user prompts
- C- image moderation
- D- prompt shields for documents

Answer:

D

Explanation:

The correct answer is D. prompt shields for documents. The OCR text is extracted from uploaded images and appended as additional context, so it is third-party content rather than a direct trusted user instruction. Microsoft Foundry Prompt Shields distinguish between user prompt attacks, which are malicious instructions directly supplied as user prompts, and document attacks, which are hidden or embedded instructions in third-party content such as documents, emails, webpages, or grounded data. Microsoft also notes that Prompt Shields analyze indirect attacks embedded in input documents or images.

This scenario is an indirect prompt injection pattern: an uploaded screenshot may contain text that says to ignore the system prompt or perform unauthorized actions. Because the OCR output is being passed to the multimodal model as contextual content, the appropriate protection is Prompt Shields for documents, which scans externally sourced context for malicious embedded instructions before generation. Protected material text detects known copyrighted text in model outputs, not prompt injection. Image moderation detects harmful visual content categories, not malicious instructions in extracted context. Prompt Shields for user prompts are less precise here because the risk comes from document-like content derived from the uploaded image. Reference topics: Prompt Shields, document attacks, indirect prompt injection, multimodal safety, OCR-derived context, and Foundry guardrails.



To Get Premium Files for AI-103 Visit

<https://www.p2pexams.com/products/ai-103>

For More Free Questions Visit

<https://www.p2pexams.com/microsoft/pdf/ai-103>

20%
DISCOUNT

P2P
exams