PassQueen
a.k.a dumpsqueen.com

Developing AI Apps and Agents on Azure

Microsoft AI-103

Version Demo

Total Demo Questions: 11

Total Premium Questions: 111

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Develop generative AI apps by using Azure AI Foundry	23
Topic 2, Develop an agentic solution by using Azure AI Foundry	57
Topic 3, Develop AI apps by using Azure AI services	22
Topic 4, Case Study Contoso, Ltd Overview	9
Total	111

QUESTION NO: 1 - (SIMULATION)

You have a Python application that uses Azure OpenAt structured outputs to extract fields from unstructured receipt text and support ticket text. The application uses the following schema.

```
receipt_response_format = {
    "type": "json_schema",
    "json_schema": {
        "name": "ReceiptExtraction",
        "strict": True,
```

Answer Area

Statements	Yes	No
The schema requires that a receipt has a `discount_code` value.	○	○
The schema can be used as is for Azure OpenAI structured outputs.	○	○
The generated JSON object will preserve the schema property order of `merchant`, `order_number`, `discount_code`, `line_items`, and `total`.	○	○

ANSWER: See the explanation for the answer

Explanation:

Select the following answers:

The `discount_code` field allows a null value. Therefore, the output can include `"discount_code": null` when no discount code appears on the receipt; a substantive discount-code value is not required.

The schema is not valid for Azure OpenAI structured outputs without changes. Structured-output schemas must use the supported JSON Schema subset: object properties must be listed in `required`, optional-like values must be modeled by allowing `null`, and each object must set `additionalProperties` to `false`. Setting `strict` to `true` does not remove these schema requirements.

When structured output is generated, keys are emitted in the order in which they are declared in the schema. Thus, the specified receipt properties retain the stated order.

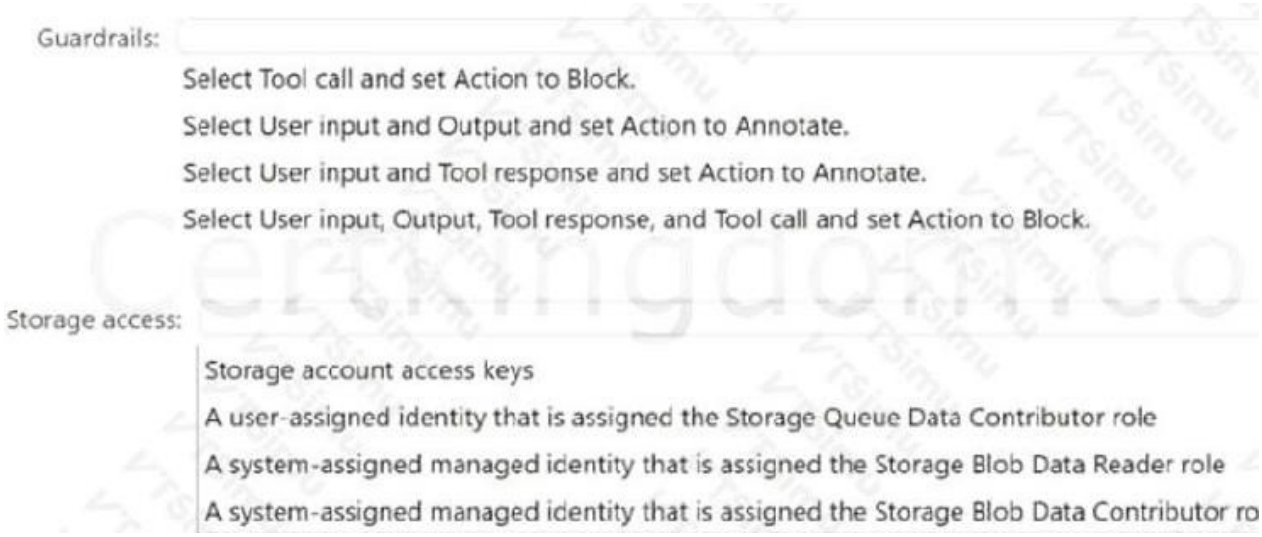
QUESTION NO: 2 - (HOTSPOT)

You have a Microsoft Foundry project that contains a customer support agent built by using the Foundry Agent Service. The agent uploads user-provided screenshots to Azure Storage through a ticketing tool and receives a blob URL for additional reasoning.

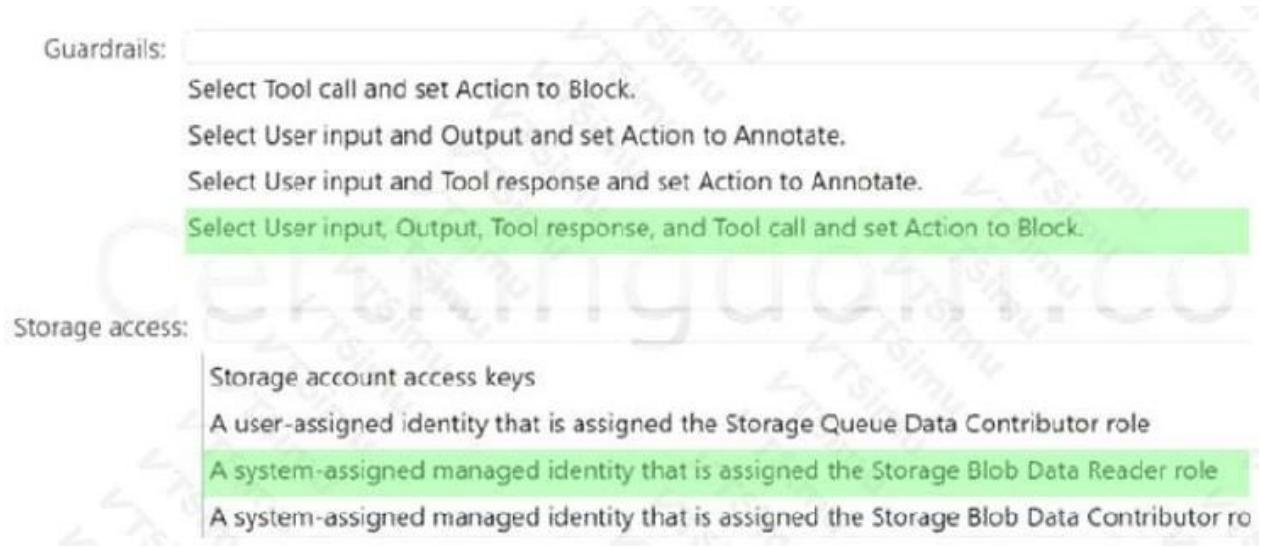
You need to use image moderation during agent runs and prevent harmful content from being returned during runs. Azure AI

Content Safety must access the images by using the blob URL. The solution must follow the principle of least privilege. What should you configure for Content Safety? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.



ANSWER:



Explanation:

Content Safety must be able to inspect content throughout the complete agent execution path. In this workflow, a user can submit content that leads the agent to invoke a ticketing tool, the tool can return a blob URL or associated information, and the agent can then generate a final response based on that material. Applying guardrails to User input, Output, Tool response, and Tool call ensures that safety checks apply at every relevant point where content enters, is passed to a tool, is returned by a tool, or is emitted to the user. Setting the action to Block is appropriate because the stated requirement is to prevent harmful content from being returned during runs. A blocking guardrail actively stops unsafe content from progressing through the relevant part of the run rather than merely recording moderation information.

For image retrieval, a system-assigned managed identity is the least-privileged authentication approach because the identity is managed by Azure and is bound to the Azure AI Content Safety resource. It avoids distributing storage account keys or maintaining an application credential. The Storage Blob Data Reader role supplies the required Azure Storage data-plane permission to read blobs, allowing Content Safety to fetch the screenshot identified by the blob URL for moderation. It does not grant write, delete, or broad storage-management permissions, which maintains the requested least-privilege design. Role assignment should be scoped as narrowly as practical, such as to the container holding the uploaded screenshots. Microsoft documents managed identities at Managed identities for Azure resources and documents Azure Blob data-access roles, including Storage Blob Data Reader, at Assign an Azure role for access to blob data.

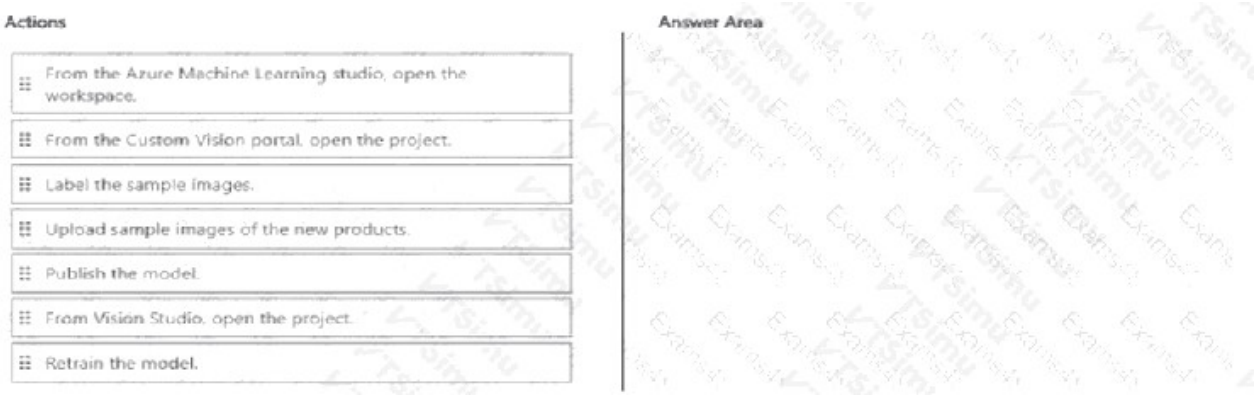
QUESTION NO: 3 - (DRAG DROP)

You have an app that uses Azure AI Custom Vision and a custom trained classifier to identify products in images. You need to add new products to the classifier. The solution must meet the following requirements:

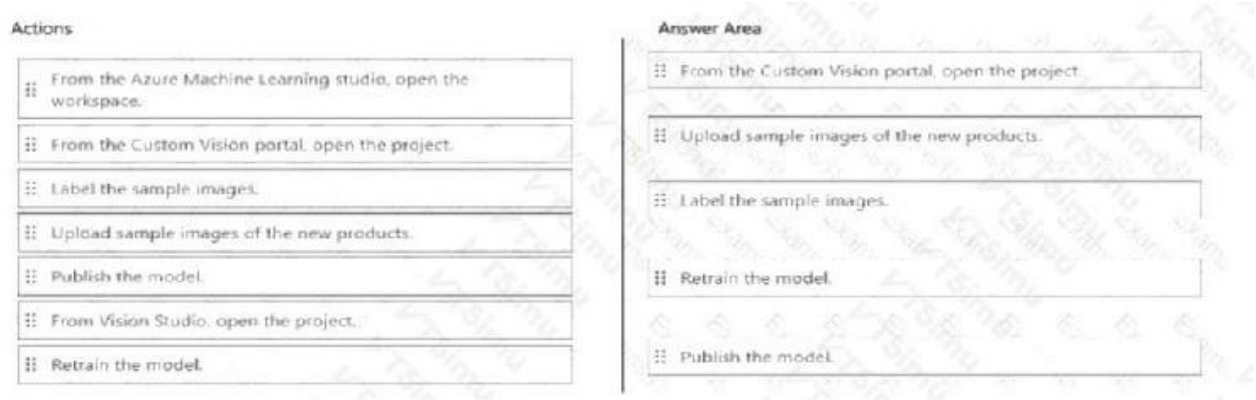
• Minimize how long it takes to add the products.

• Minimize development effort.

Which five actions should you perform in sequence? To answer, move the appropriate actions from the list of actions to the answer area and arrange them in the correct order.



ANSWER:



Explanation:

The appropriate workflow starts by opening the existing project from the **Custom Vision portal**. This is the portal designed specifically for managing Azure AI Custom Vision projects, including the image dataset, tags, training iterations, evaluation results, and publication settings. Using the existing project preserves the trained classifier and its accumulated labeled data, so the new product classes can be added without creating a separate model or writing integration code.

After opening the project, upload representative sample images for each new product. These images become part of the project's training dataset. Next, label the samples with the correct product tag. In Custom Vision, tags define the categories that an image-classification model learns to recognize, so each new product needs an appropriately named tag and correctly associated examples.

Once the new images are uploaded and tagged, retrain the model. Training creates a fresh iteration that learns from the existing dataset together with the newly added product images and labels. This direct portal-based process meets the goal of minimizing both elapsed time and development effort because it uses the managed Custom Vision workflow rather than requiring a custom machine-learning pipeline.

Finally, publish the trained iteration. Publishing exposes that specific trained version through the prediction resource, allowing the existing application to submit images and receive classifications that include the new products. Microsoft documents this lifecycle of adding tagged images, training an iteration, and publishing it in the Custom Vision image classification quickstart and the Custom Vision prediction API guidance.

QUESTION NO: 4

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 run an evaluation flow that scores responses for completeness and blocks responses that fall below a defined threshold.

Does this meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct because an evaluation flow measures and enforces a quality threshold, but it does not improve the agent's ability to generate complete summaries. Scoring an answer for completeness can identify when required regulatory clauses are absent, and blocking a response below the threshold prevents an incomplete answer from being delivered. However, the agent has not been given a mechanism to correct, regenerate, or augment that response. Consequently, users may receive no response rather than a complete response, while the underlying omission behavior remains unchanged.

To improve completeness, the generation workflow needs a remediation step. For example, the agent can be instructed to enumerate and preserve all applicable regulatory clauses, or a post-generation verification step can compare the draft with retrieved documents and trigger a targeted revision when clauses are missing. Evaluation remains valuable for establishing a baseline, validating changes, and monitoring quality over time, but it is not itself a generation-time correction mechanism. Microsoft describes Azure AI Foundry evaluations as tools to assess application quality through evaluators and metrics, including RAG-focused evaluation capabilities. See Azure AI Foundry RAG evaluators and Evaluate generative AI applications with Azure AI Foundry.

QUESTION NO: 5

You have a Microsoft Foundry agent that calls an OpenAPI tool to create service tickets.

The tool can make changes to customer records. You need to reduce the risk of unintended changes while retaining the ability to create tickets when a user explicitly requests the action.

Which two actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Require user confirmation before executing the action.

B. Use least-privilege credentials for the tool connection.

C. Increase the temperature parameter.

D. Store the external API key in the agent instructions.

E. Disable tracing for tool invocations.

ANSWER: A B

Explanation:

Require **user confirmation before executing the action** so that a user can review and approve a consequential operation before the tool sends the request to the external service. This creates a human approval point for changes to customer records.

Use **least-privilege credentials** for the tool connection. The connection identity should have only the permissions required to create the intended ticket records and should not receive broader administrative access to customer systems. Together, confirmation and scoped authorization reduce the impact of an incorrect tool selection or malicious instruction.

Increasing temperature makes outputs less deterministic and does not provide an authorization boundary. Storing the API key directly in agent instructions exposes a secret and is not an appropriate credential-management practice. See Use OpenAPI specified tools in Microsoft Foundry and AI agent design patterns.

QUESTION NO: 6

You have a Microsoft Foundry project that contains an agent.

The agent must access documents stored in an Azure Storage account. The organization prohibits public network access to the storage account and requires traffic to remain on the Microsoft network.

Which two actions should you perform? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Create a managed private endpoint to the Azure Storage account.

B. Approve the private endpoint connection on the Azure Storage account.

C. Enable anonymous blob access.

D. Add the storage account key to the agent instructions.

E. Enable public network access from all networks.

ANSWER: A B

Explanation:

Create a **managed private endpoint** from the Microsoft Foundry managed virtual network to the Azure Storage account and approve the **private endpoint connection** on the storage account. A managed private endpoint provides private connectivity from the managed network to an Azure resource, avoiding use of the public endpoint.

The target resource owner must approve the private endpoint connection before it can be used. After approval, the storage account can disable public network access while allowing the managed private connection required by the project. Microsoft documents managed virtual network isolation and managed private endpoints in the Microsoft Foundry managed network documentation.

QUESTION NO: 7

You have a Microsoft Foundry project that contains an agent.

The agent ingests scanned PDF vendor invoices that contain tables and embedded QR codes. The agent must preserve the PDF layout in the extracted output to ensure that downstream processing can reference sections and tables.

You plan to call Azure Content Understanding in Foundry Tools.

You need to extract content and layout elements and detect QR codes without requiring a language model deployment. Which built-in analyzer should you use?

A. prebuilt-layout

B. prebuilt-documentFieldSchema

C. prebuilt-read

D. prebuilt-documentSearch

ANSWER: A

Explanation:

The **prebuilt-layout** analyzer is appropriate because it is the Azure Content Understanding built-in analyzer designed to extract a document's text together with its physical and logical layout. For scanned invoice PDFs, this means the result can retain structural information required by downstream processes, including paragraphs, sections, tables, and their relationships within the original page layout. This is substantially more useful than plain text extraction when later processing must identify a particular table or reference a section in context.

It also supports barcode extraction as part of document layout analysis, enabling the service to identify and decode embedded QR codes. Therefore, a single analysis operation can provide both the structured layout output and QR-code information needed for invoice processing. As a prebuilt Content Understanding analyzer, it can be used directly in Foundry Tools and does not require the project to deploy a generative language model. Microsoft documents the available built-in analyzers and their document-analysis capabilities in the Content Understanding prebuilt analyzers documentation. Configuration and invocation guidance is available in the Use Content Understanding analyzers documentation.

QUESTION NO: 8

You have a Microsoft Foundry project that contains an agent for technical support.

You need to evaluate the agent before deployment. The evaluation must measure whether responses are supported by retrieved context and whether the responses address the users' questions.

Which two evaluators should you use? Each correct answer presents part of the solution.

NOTE: Each correct selection is worth one point.

A. Groundedness

B. Relevance

C. Latency

D. Token usage

E. Content Safety

ANSWER: A B

Explanation:

Groundedness and **relevance** are appropriate evaluators. Groundedness evaluates whether a generated response is supported by the context supplied to the application, such as documents returned by a retrieval pipeline. This is essential for verifying that support answers are based on retrieved technical content.

Relevance evaluates whether the response appropriately addresses the user's question. Used together, these evaluators identify responses that are supported by context but fail to answer the question, as well as responses that seem responsive but are not grounded in the retrieved evidence. Microsoft documents built-in quality evaluators in the Microsoft Foundry evaluation metrics documentation.

QUESTION NO: 9

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 multimodal AI generative model that accepts image uploads and uses extracted image text to generate responses.

You discover that users can upload unsafe images and embed hidden instructions into images to manipulate the model. You need to implement controls to mitigate the risk.

Solution: You configure protected material detection. Does this meet the goal?

A. Yes

B. No

ANSWER: B

Explanation:

No is correct. Protected material detection does not provide the controls needed for the risks in this scenario. This feature is intended to identify whether model-generated text or code matches known protected material, helping applications handle copyright-related concerns in generated output. It does not assess whether an uploaded image contains unsafe visual material, and it does not identify malicious instructions hidden in image text that could influence a multimodal model.

The stated goal involves two input-side protections. Image uploads should be evaluated with Azure AI Content Safety image analysis to detect harmful visual content according to supported harm categories and configured severity thresholds. Text extracted from an image is external, untrusted content; therefore, hidden instructions in that text represent an indirect prompt-injection risk. Prompt Shields can detect indirect attacks, also called document attacks, in content that is supplied to a generative AI model. Applying these relevant controls before image-derived content reaches the model helps reduce unsafe-input and instruction-manipulation risk. Microsoft distinguishes protected material detection from image moderation and Prompt Shields in its safety capabilities. See Azure AI Content Safety overview and Prompt Shields and indirect attack detection.

QUESTION NO: 10

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?

A. evaluation metrics

B. token usage

C. latency

D. run success rate

ANSWER: B

Explanation:

token usage is the appropriate observability capability because generative-model consumption and associated costs are measured from the tokens processed by the model. With request volume unchanged, a cost increase is most directly

investigated by comparing token consumption before and after the deployment update. Input-token telemetry reveals whether prompts, conversation history, retrieved grounding content, or tool results have increased the context sent to the model. Output-token telemetry shows whether the agent is generating longer responses. Together, these measurements identify whether the cost change comes from larger model requests, larger completions, or additional model work.

For an agent, expanded tool use can also raise token usage indirectly. Tool calls and their returned data may be included in later model turns, increasing the input context; orchestration can also result in more model invocations during a run. Token measurements therefore provide the cost-focused signal needed to quantify the effect of these behavioral changes. Azure AI Foundry application monitoring supports collection and analysis of generative AI telemetry, while agent tracing can provide run-level context when correlating consumption with tool activity. See Monitor applications in Azure AI Foundry and Tracing for Azure AI Foundry Agent Service.

QUESTION NO: 11 - (HOTSPOT)

You need to ensure that the marketing department can generate videos by using the model deployed to Project2. now should you complete the Python code? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Answer Area

```
import time
from openai import OpenAI
client = OpenAI(
    base_url="https://Contoso.openai.azure.com/openai/v1/",
    api_key=...,
)
video = client.videos.                    (
    model=deployment_name
    prompt="A video of ou
)
while video.status not in ["completed", "failed", "cancelled"]:
    time.sleep(20)
    video = client.videos.                (video.id)
print(video.status)
```

create
download_content
list
retrieve

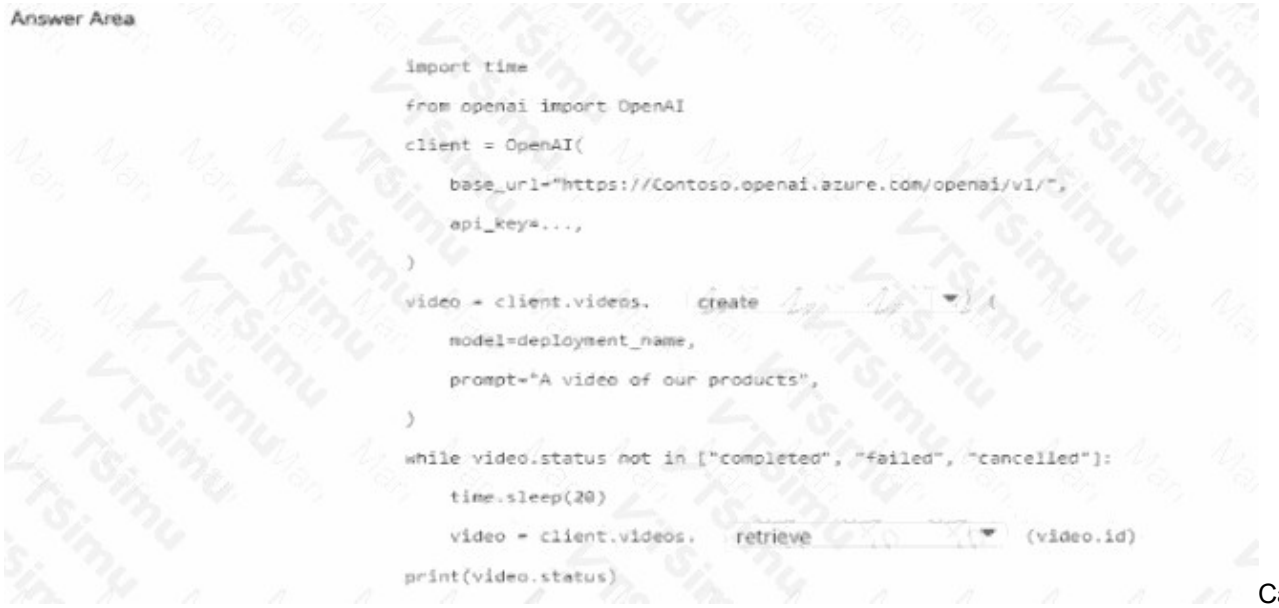
create
download_content
list
retrieve

ANSWER:

Answer:

Answer:

Explanation:



Call client.videos.create() to submit the prompt and start a video-generation job against the model deployment. Video generation is asynchronous: the initial response represents a job and contains properties such as its unique ID and current status rather than immediately returning the completed video. Microsoft's official Python example creates the job by using client.videos.create(model=..., prompt=...). The application must then poll the job. After waiting between requests, client.videos.retrieve(video.id) obtains the latest state of that specific job. Polling continues until the status becomes completed, failed, or cancelled. list would enumerate jobs rather than refresh the selected job, while download_content is used only after successful completion to retrieve the generated video file. To comply with Contoso's prohibition on API keys, the client should use a Microsoft Entra bearer-token provider. The OpenAI Python client accepts this provider through its api_key constructor parameter even though no static API key is used. Study Guide alignment: **design and implement video-generation solutions** and integrate generative workflows through Foundry SDKs.

Explanation:

The correct implementation uses `client.videos.create(...)` to submit the video-generation request and `client.videos.retrieve(video.id)` to monitor that specific request until it reaches a terminal state. The create call includes the deployment name and the marketing prompt, so it initiates video generation against the model deployed to

Project2. Video generation is an asynchronous operation: submitting the request returns a video job object rather than waiting for the completed media file. That job object supplies an identifier and a status value that the application can use to track progress.

The loop checks whether `video.status` is still outside the terminal states `completed`, `failed`, and `cancelled`. While the job remains active, the code waits briefly and calls `client.videos.retrieve(video.id)`. Passing the saved job ID is important because the application must obtain the current state of the exact video-generation job it created. Assigning that refreshed result back to `video` ensures that the next loop condition evaluates the latest reported status. When the job becomes completed, the application can proceed with the resulting video resource; the status printing line provides visibility into the final outcome. This follows the asynchronous job pattern documented for Azure OpenAI video generation. See the Azure OpenAI video generation documentation and the OpenAI Python library documentation for the Python client workflow.