Vendor: NVIDIA

Exam Code: NCP-AAI

Exam Name: Agentic AI

Version: DEMO

QUESTION 1
An engineer has created a working AI agent solution providing helpful services to users. However, during live testing, the AI agent does not perform tasks consistently. Which two potential solutions might help with this issue? (Choose two.)

A. Remove schema validations and assertions on tool outputs to avoid inconsistency.
B. Increase randomness (e.g., temperature) and remove fixed seeds to avoid determinism.
C. Identify where dividing the tasks into subtasks and handling them by multiple agents can help.
D. Refine the prompt given to the AI Agent; be clear on objectives

Answer: CD
Explanation:
The selected design maps to Identify where dividing the tasks into subtasks and handling them by multiple agents can help and Refine the prompt given to the AI Agent be clear on objectives, which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. For optimization, NeMo Agent Toolkit profiling and evaluation expose workflow timing, token flow, tool latency, and quality metrics that single-output grading cannot capture. The agent should not infer operational details from latent model knowledge when it can bind to structured tools, retrievers, schemas, and examples. This reduces hallucinated endpoints, malformed parameters, stale facts, and brittle parsing when APIs, documents, or user inputs change.

QUESTION 2
A development team is building a customer support agent that interacts with users via chat. The agent must reliably fetch information from external databases, handle occasional API failures without crashing, and improve its responses by learning from user feedback over time. Which of the following tasks is most critical when enhancing an AI agent to handle real-world interactions and improve over time?

A. Applying a well-structured training process with foundational generative models and prompt engineering
B. Utilizing internal knowledge bases to support agent responses alongside external APIs
C. Implementing retry logic for error handling and integrating user feedback loops for iterative improvement
D. Designing conversation flows that provide consistent responses based on predefined scripts

Answer: C
Explanation:
The selected design maps to Implementing retry logic for error handling and integrating user feedback loops for iterative improvement, which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. For knowledge-grounded agents, the clean architecture is a RAG path with retrievers and vector indexes externalized from the LLM, then evaluated for retrieval quality and answer faithfulness. The agent should not infer operational details from latent model knowledge when it can bind to structured tools, retrievers, schemas, and examples. This reduces hallucinated endpoints, malformed parameters, stale facts, and brittle parsing when APIs, documents, or user inputs change.

QUESTION 3
What NVIDIA framework can be used to train a better agent?

A. NeMo-RL
B. NeMo Guardrails
C. TensorRT-LLM

NCP-AAI
Answer: A
Explanation:
The selected design maps to NeMo-RL, which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. For learning and adaptation, NeMo RL, NeMo Gym, and NeMo Framework fine-tuning provide the training path, while deployment still requires external state and guardrailed execution. The agent should not infer operational details from latent model knowledge when it can bind to structured tools, retrievers, schemas, and examples. This reduces hallucinated endpoints, malformed parameters, stale facts, and brittle parsing when APIs, documents, or user inputs change.

QUESTION 4
You are evaluating your RAG pipeline. You notice that the LLM-as-a-Judge consistently assigns high similarity scores to responses that contain irrelevant information. What should you investigate as the most likely potential cause with the least development effort?

A. The temperature setting used by the LLM during response generation.
B. The size of the knowledge base used to power the RAG pipeline.
C. The quality of the synthetic questions used for evaluation.
D. The prompt used to instruct the LLM-as-a-Judge to assess the response.

Answer: D
Explanation:
The selected design maps to The prompt used to instruct the LLM-as-a-Judge to assess the response, which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. For knowledge-grounded agents, the clean architecture is a RAG path with retrievers and vector indexes externalized from the LLM, then evaluated for retrieval quality and answer faithfulness. The evaluation target is the full agent workflow: planning quality, tool selection, intermediate state, latency, retries, user feedback, and final task completion. Instrumentation must expose where degradation starts so remediation can focus on prompts, tool schemas, retrieval, model parameters, or infrastructure rather than random retuning.

QUESTION 5
You're managing an agentic AI responsible for customer support ticket triage. The agent has been consistently accurate in routing tickets to the appropriate departments. However, a team leader has noticed a significant increase in the number of tickets requiring "escalation" ?cases where the agent initially misclassified a complex issue as a simple, routine one, leading to delays and frustrated customers. What would be an appropriate first step in resolving this issue?

A. Analyzing the agent's decision-making process, focusing on the specific criteria it uses to classify tickets, and identifying potential biases or blind spots.
B. Adjusting the agent's reward function to prioritize speed of resolution over accuracy, as a first step in analysis of the problem.
C. Increasing the agent's autonomy, granting it more decision-making power during triage to improve its efficiency.
D. Conducting a "red-teaming" exercise, having human agents deliberately create complex and ambiguous scenarios to analyze the agent's robustness.

Answer: A
Explanation:
The selected design maps to Analyzing the agent s decision-making process focusing on the specific criteria it uses to classify tickets and identifying..., which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. NeMo Agent Toolkit evaluation,

profiling, and OpenTelemetry-style observability are built for workflow-level measurement, not just isolated answer inspection. The evaluation target is the full agent workflow: planning quality, tool selection, intermediate state, latency, retries, user feedback, and final task completion. Instrumentation must expose where degradation starts so remediation can focus on prompts, tool schemas, retrieval, model parameters, or infrastructure rather than random retuning.

QUESTION 6
Which two optimization strategies are MOST effective for improving agent performance on NVIDIA GPU infrastructure? (Choose two.)

A. Using multi-GPU coordination to distribute workloads, enabling higher throughput and efficiency for scaling agent tasks.
B. Applying TensorRT-LLM optimizations to reduce inference latency by improving kernel efficiency and memory usage.
C. Expanding GPU memory capacity to support larger models, assuming this alone guarantees meaningful performance improvements.
D. Manually tuning kernel launch parameters to optimize individual operations while overlooking overall pipeline performance dynamics.

Answer: AB
Explanation:
The selected design maps to Using multi-GPU coordination to distribute workloads enabling higher throughput and efficiency for scaling agent tasks and Applying TensorRT-LLM optimizations to reduce inference latency by improving kernel efficiency and memory usage, which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. The deployment logic aligns with NVIDIA NIM for containerized inference, TensorRT-LLM for optimized engines, and Triton for batching, scheduling, and Prometheus-visible inference metrics. Performance comes from matching workload shape to serving topology: small requests, large reasoning calls, embeddings, rerankers, and multimodal models should scale on separate resource signals. GPU utilization, queue depth, dynamic batching, model precision, and container lifecycle are therefore first-class design variables, not after-the-fact tuning knobs.

QUESTION 7
You are rolling out a multimodal conversational agent on NVIDIA's stack: the model is containerized as a TensorRT-LLM engine, served via Triton Inference Server behind NIM microservices for routing and scaling, and protected by NeMo Guardrails for safety and compliance. During early testing, end- to-end latency exceeds your target budget, and you need to tune batching, model precision, and guardrail checks while maintaining both throughput and enforcement of safety policies. Which configuration change is most effective for reducing latency under these constraints while still enforcing NeMo Guardrails policies?

A. Quantize the TensorRT-LLM engine to FP16, tune Triton's dynamic batching, and integrate NeMo Guardrails alongside inference to run policy checks in parallel.
B. Quantize the TensorRT-LLM engine to INT8, disable dynamic batching, and invoke Guardrails checks synchronously within the inference path.
C. Deploy separate Triton servers for model inference and guardrail validation, routing requests sequentially and merging outputs at the application layer.
D. Keep FP32 precision, increase batch size aggressively, and perform Guardrails checks in a downstream microservice after inference.

Answer: A
Explanation:
The selected design maps to Quantize the TensorRT-LLM engine to FP16 tune Triton s dynamic

batching and integrate NeMo Guardrails alongside inference to..., which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. The NVIDIA stack component that anchors this design is NeMo Guardrails, because rails can be placed before retrieval, during dialog, around tool execution, and after generation. Performance comes from matching workload shape to serving topology: small requests, large reasoning calls, embeddings, rerankers, and multimodal models should scale on separate resource signals. GPU utilization, queue depth, dynamic batching, model precision, and container lifecycle are therefore first-class design variables, not after- the-fact tuning knobs.

QUESTION 8
A healthcare AI company is deploying diagnostic agents that process medical imaging and patient data. The system must deliver consistent sub-100ms inference times for critical diagnoses while supporting deployment across multiple hospital sites with different NVIDIA GPU configurations (from RTX 6000 workstations to DGX systems). The agents need to maintain high accuracy while being portable across different hardware environments and capable of running efficiently on various GPU memory configurations. Which optimization strategy would deliver the BEST performance improvements while maintaining deployment flexibility across diverse NVIDIA hardware configurations?

A. Deploy agents with NVIDIA CUDA-optimized Docker containers using a sequential inference architecture that processes each layer individually with GPU-to-CPU memory transfers between operations to avoid memory issues.
B. Deploy agents using NVIDIA NIM containers with CPU-optimized inference to avoid GPU memory constraints and ensure consistent performance across different hospital infrastructure configurations.
C. Deploy models using NVIDIA TensorRT optimization in their original FP32 precision format without any quantization or memory optimization, requiring 32GB+ GPU memory across all deployment sites.
D. Deploy agents using model optimizations with post-training quantization with Nvidia NIM deployment for portable performance across different GPU platforms and memory configurations.

Answer: D
Explanation:
The selected design maps to Deploy agents using model optimizations with post-training quantization with Nvidia NIM deployment for portable performance across different GPU..., which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. The deployment logic aligns with NVIDIA NIM for containerized inference, TensorRT-LLM for optimized engines, and Triton for batching, scheduling, and Prometheus-visible inference metrics. Performance comes from matching workload shape to serving topology: small requests, large reasoning calls, embeddings, rerankers, and multimodal models should scale on separate resource signals. GPU utilization, queue depth, dynamic batching, model precision, and container lifecycle are therefore first-class design variables, not after-the-fact tuning knobs.

QUESTION 9
You are tasked with deploying a multi-modal agentic system that must respond to user queries with minimal latency while maintaining guardrails for safe and context-aware interactions. Which of the following configurations best leverages NVIDIA's AI stack to meet these requirements?

A. Integrate NeMo Guardrails, configure NIM microservices for optimized inference, use TensorRT-LLM for deployment, and profile the system using Triton Inference Server with multi-modal support.
B. Integrate NeMo Guardrails, use Omniverse to generate synthetic data, configure NIM microservices for optimized inference, use TensorRT-LLM for deployment, and profile the system

using NeMo Agent Toolkit for multi-modal support.

C. Use NeMo Guardrails for safety, deploy the model with Triton Inference Server using default settings, and rely on hardware accelerators like GPU/TPU inference for cost efficiency.

D. Use NIM microservices for deployment, optionally use NeMo Guardrails unless one wants to minimize the inference overhead.

Answer: A
Explanation:
The selected design maps to Integrate NeMo Guardrails configure NIM microservices for optimized inference use TensorRT-LLM for deployment and profile the system using..., which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. The NVIDIA stack component that anchors this design is NeMo Guardrails, because rails can be placed before retrieval, during dialog, around tool execution, and after generation. Performance comes from matching workload shape to serving topology: small requests, large reasoning calls, embeddings, rerankers, and multimodal models should scale on separate resource signals. GPU utilization, queue depth, dynamic batching, model precision, and container lifecycle are therefore first-class design variables, not after-the-fact tuning knobs.

QUESTION 10
Integrate NeMo Guardrails, configure NIM microservices for optimized inference, use TensorRT-LLM for deployment, and profile the system using Triton Inference Server with multi-modal support. Which of the following strategies aligns with best practices for operationalizing and scaling such Agentic systems?

A. Use Docker containers orchestrated by Kubernetes, implement MLOps pipelines for CI/CD, monitor agent health with Prometheus/Grafana.

B. Deploy agents on bare-metal servers to maximize performance and avoid container overhead, using manual scripts for orchestration and monitoring.

C. Deploy all agents on a single high-performance GPU node to reduce latency, and use cron jobs for periodic health checks and updates.

D. Run agents as independent serverless functions to minimize infrastructure management, relying primarily on cloud provider auto-scaling and logging tools.

Answer: A
Explanation:
The selected design maps to Use Docker containers orchestrated by Kubernetes implement MLOps pipelines for CI/CD monitor agent health with Prometheus/Grafana, which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. The NVIDIA stack component that anchors this design is NeMo Guardrails, because rails can be placed before retrieval, during dialog, around tool execution, and after generation. Performance comes from matching workload shape to serving topology: small requests, large reasoning calls, embeddings, rerankers, and multimodal models should scale on separate resource signals. GPU utilization, queue depth, dynamic batching, model precision, and container lifecycle are therefore first-class design variables, not after-the-fact tuning knobs.

QUESTION 11
When evaluating optimization opportunities between NeMo Guardrails, NIM microservices, and TensorRT-LLM in a production healthcare agent, which analysis approach best identifies optimization opportunities across the NVIDIA stack?

A. Conduct stress testing of individual microservices and guardrails to measure peak throughput and determine theoretical performance limits of each module.

B. Use default configurations to establish a deployment baseline, focusing on stability before conducting deeper performance profiling.
C. Create end-to-end latency waterfalls that capture guardrail overhead, NIM queuing delays, and TensorRT optimization benefits while assessing overall pipeline efficiency.
D. Tune each component individually, focusing primarily on local performance metrics with secondary attention to integration patterns.

Answer: C
Explanation:
The selected design maps to Create end-to-end latency waterfalls that capture guardrail overhead NIM queuing delays and TensorRT optimization benefits while assessing overall..., which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. The NVIDIA stack component that anchors this design is NeMo Guardrails, because rails can be placed before retrieval, during dialog, around tool execution, and after generation. Performance comes from matching workload shape to serving topology: small requests, large reasoning calls, embeddings, rerankers, and multimodal models should scale on separate resource signals. GPU utilization, queue depth, dynamic batching, model precision, and container lifecycle are therefore first-class design variables, not after-the-fact tuning knobs.

QUESTION 12
When evaluating GPU utilization inefficiencies in deploying Llama Nemotron models across A100 and H100 clusters, which approaches help identify optimal resource allocation strategies? (Choose two.)

A. Allow Nemotron variants to profile actual workload characteristics and allocate resources based on observed demands.
B. Profile resource utilization for each Nemotron variant and match models to appropriate GPU tiers.
C. Allocate all agents to Hl00 GPUs, allowing resource profiles to automatically adjust for model size and computational requirements.
D. Assess concurrent execution capabilities by employing multi-instance GPU partitioning for varying workload types.

Answer: BD
Explanation:
The selected design maps to Profile resource utilization for each Nemotron variant and match models to appropriate GPU tiers and Assess concurrent execution capabilities by employing multi-instance GPU partitioning for varying workload types, which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. The deployment logic aligns with NVIDIA NIM for containerized inference, TensorRT-LLM for optimized engines, and Triton for batching, scheduling, and Prometheus-visible inference metrics. Performance comes from matching workload shape to serving topology: small requests, large reasoning calls, embeddings, rerankers, and multimodal models should scale on separate resource signals. GPU utilization, queue depth, dynamic batching, model precision, and container lifecycle are therefore first-class design variables, not after-the-fact tuning knobs.

QUESTION 13
A financial services company is deploying a multi-agent customer service system consisting of three specialized agents: a reasoning LLM for complex queries, an embedding agent for document retrieval, and a re-ranking agent for result optimization. The system experiences significant traffic variations, with peak loads during business hours (10x normal traffic) and minimal usage overnight. The company needs a deployment solution that can handle these fluctuations cost-effectively while maintaining sub-second response times during peak periods. Which NVIDIA infrastructure approach would provide the MOST cost-effective and scalable

deployment solution for this variable-load multi-agent system?

A. Deploy agents directly on individual NVIDIA RTX workstations without containerization or orchestration, relying on load balancers with round-robin for traffic distribution.
B. Deploy each agent on dedicated NVIDIA DGX systems with manual scaling based on previous days traffic predictions and static resource allocation for peak loads.
C. Deploy NVIDIA NIM microservices on Kubernetes with auto-scaling capabilities, utilizing NVIDIA NIM Operator for lifecycle management and horizontal pod autoscaling based on custom metrics.
D. Deploy all agents on a single large GPU instance without containerization, scaling compute by upgrading to larger GPU instances when needed.

Answer: C
Explanation:
The selected design maps to Deploy NVIDIA NIM microservices on Kubernetes with auto-scaling capabilities utilizing NVIDIA NIM Operator for lifecycle management and horizontal..., which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. The deployment logic aligns with NVIDIA NIM for containerized inference, TensorRT-LLM for optimized engines, and Triton for batching, scheduling, and Prometheus-visible inference metrics. Performance comes from matching workload shape to serving topology: small requests, large reasoning calls, embeddings, rerankers, and multimodal models should scale on separate resource signals. GPU utilization, queue depth, dynamic batching, model precision, and container lifecycle are therefore first-class design variables, not after-the-fact tuning knobs.

QUESTION 14
Your team notices a spike in failed tool calls from a deployed workflow agent after a recent API schema update. The agent still returns outputs, but many are irrelevant or incomplete. Which maintenance task should be prioritized to restore accurate behavior?

A. Reset the agent's long-term memory and reinitialize logs.
B. Update the tool function specifications and re-test action sequences.
C. Increase model temperature to encourage tool exploration.
D. Reduce tool retrieval vector similarity threshold to broaden context.

Answer: B
Explanation:
The selected design maps to Update the tool function specifications and re-test action sequences, which is the highest-control path for this scenario rather than a prompt-only or single-service shortcut. For tool-using agents, the durable pattern is schema-bound function invocation with timeouts, typed outputs, retry policy, and traceable execution rather than free-form endpoint guessing. The evaluation target is the full agent workflow: planning quality, tool selection, intermediate state, latency, retries, user feedback, and final task completion. Instrumentation must expose where degradation starts so remediation can focus on prompts, tool schemas, retrieval, model parameters, or infrastructure rather than random retuning.

Thank You for Trying Our Product

Lead2pass Certification Exam Features:

★ More than **99,900** Satisfied Customers Worldwide.

★ Average **99.9%** Success Rate.

★ **Free Update** to match latest and real exam scenarios.

★ **Instant Download** Access! No Setup required.

★ Questions & Answers are downloadable in **PDF** format and **VCE** test engine format.

★ Multi-Platform capabilities - **Windows, Laptop, Mac, Android, iPhone, iPod, iPad**.

★ **100%** Guaranteed Success or **100%** Money Back Guarantee.

★ **Fast**, helpful support **24x7**.

View list of all certification exams: http://www.lead2pass.com/all-products.html

10% Discount Coupon Code: **ASTR14**