Nabila Hannania

Engineering Knowledge AI Agent Test

1. Describe differences between REST API, MCP in the context of AI.

REST API (Representational State Transfer API) is a type of API (Application Programming Interface) that allows communication between different systems over the internet. REST API works by sending requests and receiving responses, typically in JSON format, between the client and server. In the context of AI, REST API is commonly used to call inference endpoints, where input data is sent and predictions are returned.

Meanwhile, MCP (Model Context Protocol) is a standard way to make information available to large language models (LLMs). Unlike REST API, MCP acts as a standardization layer for AI applications, allowing them to communicate effectively with external services such as tools, databases, and predefined templates. Its main focus is on enabling AI agents to call external tools, retrieve knowledge, and operate effectively within structured environments.

The key different of REST API and MCP:

- Purpose
  REST API is general-purpose. It was designed for software-to-software communication, not specifically for AI. MCP, on the other hand, was built from the ground up for large language models. It standardizes how LLMs fetch context and use tools.
- Discovery
  REST API doesn't tell you what it can do, you have to read the documentation. Meanwhile an MCP server can be queried directly with a question like "What tools do you offer?" and it will return a machine-readable list of available functions, along with their required inputs and expected outputs. This allows AI agents to discover and adapt to new capabilities without the need for manual pre-programming.
- Standardization
  Every REST API is different. Every API has its own auth, error handling, and data formats. This inconsistency creates extra work for developers. MCP removes this variability because all MCP servers communicate using the same protocol. As a result, once an AI learns how to interact with one MCP server, it can seamlessly interact with any of them.
- Adaptability
  When a REST API changes, whether by adding a new endpoint or modifying a response format, clients often break until they are manually updated to match the changes. MCP avoids this problem by enabling capability discovery at runtime. If a server introduces a new tool, the AI can immediately recognize and begin using it without requiring updates or reprogramming.

2. How REST API, MCP, can improve the AI use case.

REST API to improve AI use case:

- Model Deployment: Simplify deployment of AI/ML models by exposing them as APIs, making inference accessible across different systems.
- Data Access & Integration: Provide a standardized way for AI systems to retrieve, update, and manipulate data from various sources. This enables real-time insights, database updates, and the ability to trigger actions in other applications.
- Service Integration: Allow AI agents to interact with external services, such as payment gateways, communication platforms, or analytics tools. It enables complex, multi-step workflows.
- Scalability & Compatibility: Well suited for scaling AI services and integrating with existing web or mobile applications.

MCP to Improve AI use case:

- Tool Access: Improves AI reasoning by enabling direct, structured interactions with external tools and services.
- Reduced Hallucinations: Grounds AI responses in real data, improving accuracy and trustworthiness.
- Enhanced Contextual Understanding: Provides richer context in exchanges, helping AI models better interpret meaning, intent, and task requirements, reducing ambiguity.
- Dynamic Discovery & Invocation: Enables AI agents to discover and understand available tools at runtime. This allows them to autonomously adapt, selecting the right tools as needed without manual updates.

3. How do you ensure that your AI agent answers correctly?

Ensuring an AI agent provides accurate and reliable answers requires attention across the entire lifecycle (development - evaluation - monitoring)

- Development
    - Use High-Quality Training Data: Train the agent on a comprehensive and accurate knowledge base built from trusted sources. Regularly clean and update the data to keep it relevant and reliable.
    - Clear Instructions & Prompts: Use concise, unambiguous language to guide the agent. Define the agent's persona and purpose clearly to shape responses. Break complex tasks into smaller steps and include examples to demonstrate the desired outcomes.
    - Controlled Output Behavior: Adjust the model's temperature to balance creativity and consistency.
        - Low temperature (0.0–0.3): more predictable, consistent responses.
        - High temperature (0.8–1.0): more creative, diverse responses.
        - For accuracy-focused tasks, a lower temperature is often preferable.
- Evaluation and testing.
  Because LLMs may generate slightly different outputs even under controlled conditions, validation requires flexible strategies:
    - Normalize Responses: Standardize outputs (e.g., remove extra whitespace, unify casing, ignore punctuation) before comparing with expected results.
    - Validate Key Elements: Focus on whether the response contains the essential information rather than matching word-for-word.
    - Systematic Testing: Run benchmark tests and edge-case scenarios to check consistency and correctness across different inputs.
- Monitoring
    - Input/Output Logging: Continuously log interactions to track unexpected variations and detect drift.
    - Performance Tracking: Regularly assess the agent in real-world scenarios to ensure reliability and accuracy over time.
    - Manual Review: Complement automated checks with periodic human inspection for deeper evaluation and quality assurance.

4. Describe what can you do with Docker / Containerize environment in the context of AI

Docker / containerize environment is a powerful tool for building, shipping, and running AI services. By packaging models, tools, and dependencies into containers, you can streamline development, deployment, and scaling across environments. Some of the things you can do with docker / containerize environment include:

- Docker makes it straightforward to build and run AI agents and agentic applications. With a simple compose.yaml file, developers can define models, agents, and MCP-compatible tools, then launch the entire agentic stack using docker compose up. This approach ensures that from development to production, agents remain consistently wired, connected, and ready to run.
- Good reproducibility. Containers encapsulate the entire runtime environment, including code, libraries, dependencies, and specific versions of frameworks such as TensorFlow or PyTorch. This guarantees that AI models behave consistently across machines and environments, eliminating the “works on my machine” problem and enabling more reliable research and production workflows.
- Enables scalability. Using orchestration tools such as Kubernetes or Docker Swarm, developers can deploy inference services that automatically scale to meet demand. This allows AI applications to handle heavy workloads efficiently while providing features such as load balancing and fault tolerance.
- Provide isolation, preventing dependency conflicts between projects or applications running on the same host. This makes it easy to test different versions of models or frameworks in parallel without interference. For AI developers, this means cleaner experimentation and simpler environment setup.
- Simplifies continuous integration and continuous deployment (CI/CD) pipelines for AI projects. Because containers provide a consistent environment, automated testing, packaging, and deployment are more reliable, ensuring smooth transitions across development, testing, and production environments.

5. How do you finetune the LLM model from raw?

Fine-tuning is the process of taking a pre-trained language model and adapting it to a domain-specific task or dataset. Instead of training a model from zero, fine-tuning updates the model's parameters using new, specialized data, making it more accurate and useful in a given context.

Step by step how to fine tune LLM include:

- Choose a Dataset
  Select the dataset that will be used for fine-tuning. This is usually domain-specific, such as company documents, customer support transcripts, or any specialized text corpus. If working outside of text (VLM), the dataset might include images or multimodal data.
- Choose a Base Model
  Pick a pre-trained LLM that suits your task. Consider model size, architecture, licensing, and resource requirements when choosing the base model.
- Preprocess the Data
  Clean the dataset so it matches the model's input format. For text, this involves tokenization, sentence splitting, and removing noise. For images, it could mean tasks like detection, cropping, or resizing. The dataset should also be divided into training, validation, and test subsets to support proper evaluation.
- Select a Framework and Fine-Tuning Method
  Use popular frameworks such as Hugging Face Transformers, DeepSpeed, or PyTorch. Depending on available compute resources, choose a fine-tuning method:
  - Full fine-tuning: Update all model parameters.
  - Parameter-efficient methods: Techniques like LoRA, PEFT, or QLoRA train fewer parameters, making the process faster and more resource-efficient.
- Load Model and Data, Then Train
  Load the pre-trained model along with your dataset and fine-tune using the chosen framework and method. Monitor training closely to ensure that the model is learning without overfitting.
- Validate During Training
  Regularly evaluate the model on the validation set to measure accuracy and generalization. This step helps track progress, identify overfitting, and decide whether to adjust training strategies.
- Optimize with Hyperparameter Tuning
  Experiment with different configurations of learning rate, batch size, optimizer, and training epochs. Proper hyperparameter tuning can significantly improve performance.
- Test on Unseen Data
  Once the model is trained, evaluate it on the test set. This ensures that the fine-tuned model performs well on data it has never seen before, giving a more accurate measure of real-world performance.
- Deploy the Fine-Tuned Model

Export the final model for production use. It can be containerized with Docker and exposed as a service via REST API or MCP for seamless integration with applications and AI agents.