

## AI / ML Related Testing — Crisp One-Liner Answers

---

### **1. In LLM, what is your role, what business problem were you solving, and how did you tackle it?**

☞ I integrated LLM with API automation to reduce manual test creation, improve coverage, generate pytest scripts from Swagger, and embed them into CI/CD for continuous validation.

---

### **2. What kind of testing did you perform using LLM? How did you validate test cases and scripts? Provide examples.**

☞ I used LLM for functional API test generation (positive/negative/edge cases) and validated scripts via pytest using status checks, schema assertions, and Postman cross-verification.

---

### **3. Give examples of AI features you tested, input you provided, and expected output.**

☞ I tested AI-driven test case generation and response validation by feeding API specs and expecting accurate scenarios, assertions, and defect identification aligned with requirements.

---

### **4. What exactly did you work on with AI/ML features using DeepEval automation framework?**

☞ I used DeepEval to automate evaluation of LLM outputs by measuring accuracy, relevance, consistency, and hallucination through defined validation metrics.

---

### **5. How did you approach AI/ML for regression testing, especially via APIs?**

☞ I built automated regression suites that compare current LLM API outputs with baseline responses to detect drift, inconsistencies, and behavioral regressions.

---

### **6. How do you handle anomaly detection in API responses?**

☞ I detect anomalies by validating schema, status codes, field values, and comparing AI outputs with historical baselines to identify unexpected deviations.

---

## **7. API response schema validation using AI/ML approach?**

☞ I combine traditional schema validation with AI-based semantic checks to detect structural mismatches and hidden logical inconsistencies.

---

## **8. What LLM tools and libraries did you use for API automation with Pytest?**

☞ I used OpenAI APIs, LangChain, DeepEval, pytest, requests, JSON schema validation, and Jenkins for AI-powered API automation.

---

## **9. Purpose of hallucination check, and how do you tackle it?**

☞ Hallucination checks ensure factual correctness by validating LLM outputs against API specs, business rules, and trusted baselines using evaluation metrics.

---

## **10. Explain LLM-EB testing. Why is it needed in API?**

☞ LLM-EB testing validates non-deterministic AI outputs using evaluation metrics like accuracy and relevance instead of fixed assertions.

---

## **11. Explain NUMPY, Matplotlib, PANDAS, PPTX, Requests libraries and their usage in API automation.**

☞ NumPy handles numeric data, Pandas processes structured datasets, Requests performs API calls, Matplotlib visualizes test metrics, and Python-PPTX generates automated reports.

---

## **12. When you have prediction models, explain what type of model you used for predictions.**

☞ I used supervised ML models (classification/regression) and fine-tuned LLMs to predict trends, classify inputs, and generate intelligent recommendations.

---

---

## **AI / LLM / ML / MCP – Interview One-Liners**

### **1. What is Artificial Intelligence (AI)?**

☞ AI enables machines to simulate human intelligence for learning, decision-making, and automation.

---

## **2. What is Machine Learning (ML)?**

- 👉 ML is a subset of AI where systems learn patterns from data to improve predictions automatically.
- 

## **3. What is a Large Language Model (LLM)?**

- 👉 An LLM is a deep learning model trained on massive text data to understand and generate human-like language.
- 

## **4. What is MCP (Model Context Protocol)?**

- 👉 MCP is a protocol that lets LLMs securely interact with external tools and files for automated workflows.
- 

## **5. Difference between AI, ML, and LLM?**

- 👉 AI is the broad field, ML is data-driven learning, and LLMs are advanced ML models specialized in language.
- 

## **6. How can AI help in API testing?**

- 👉 AI automates test generation, edge case detection, anomaly analysis, and regression testing.
- 

## **7. What is Prompt Engineering?**

- 👉 Prompt engineering is crafting precise inputs to guide LLMs toward accurate outputs.
- 

## **8. What is an API in AI systems?**

- 👉 AI APIs expose models for applications to access predictions and responses programmatically.
- 

## **9. What is Training Data in ML?**

- 👉 Training data is clean, diverse labeled data used to teach ML models patterns.
- 

## **10. What is Supervised vs Unsupervised Learning?**

- 👉 Supervised uses labeled data; unsupervised discovers hidden patterns without labels.

---

## **11. What is NLP?**

☞ NLP enables machines to understand and process human language.

---

## **12. What is Model Accuracy?**

☞ Accuracy measures how correctly a model predicts expected outcomes.

---

## **13. What is AI Hallucination?**

☞ Hallucination occurs when an LLM generates confident but incorrect information.

---

## **14. What is Fine-Tuning in LLMs?**

☞ Fine-tuning adapts pretrained models using domain-specific data.

---

## **15. What is an AI Pipeline?**

☞ An AI pipeline covers data collection, preprocessing, training, deployment, and monitoring.

---

## **16. What is Model Bias?**

☞ Bias is unfair or skewed predictions caused by imbalanced training data.

---

## **17. What is REST API testing in AI systems?**

☞ It validates AI API inputs, response quality, latency, and error handling.

---

## **18. What is Tokenization in LLMs?**

☞ Tokenization splits text into units for model processing.

---

## **19. What is an Embedding?**

☞ Embeddings convert text into numerical vectors representing semantic meaning.

---

## **20. What is RAG (Retrieval-Augmented Generation)?**

☞ RAG combines LLMs with external knowledge retrieval for improved accuracy.

---

## **21. What is Latency in AI APIs?**

☞ Latency is the response time of an AI API request.

---

## **22. What are Guardrails in AI?**

☞ Guardrails restrict unsafe or invalid AI outputs.

---

## **23. What is Model Deployment?**

☞ Deployment makes AI models accessible in production via APIs.

---

## **24. What is Zero-Shot vs Few-Shot Learning?**

☞ Zero-shot performs tasks without examples; few-shot learns from limited examples.

---

## **25. How do you test AI API reliability?**

☞ By validating consistency, performance, error handling, and security.

---

## **26. What is Explainable AI (XAI)?**

☞ XAI makes AI decisions transparent and understandable.

---

## **27. What are Rate Limits in AI APIs?**

☞ Rate limits restrict API usage to prevent overload.

---

## **28. What is Data Preprocessing?**

☞ Data preprocessing cleans and prepares raw data for training.

---

## **29. What is a Prompt Injection Attack?**

☞ It's a security exploit where malicious prompts manipulate AI behavior.

---

## **30. How would you explain your AI experience as a tester?**

☞ I test AI APIs by validating accuracy, performance, and reliability while using LLMs to accelerate automated test generation.

---

## How do you use LLMs to generate test cases and pytest API scripts using an real API key?

### Overview — What You Are Building

You will use an **LLM API** (like OpenAI/Gemini) to:

1. Send API documentation as input
2. Ask LLM to generate test cases
3. Ask LLM to generate pytest automation script
4. Save output as files

Flow:

Your Python Script → LLM API → Test cases + pytest code → Save to file

---

### Step 1 — Get API Key

You need an API key from an LLM provider:

Example:

- OpenAI
- Google (Gemini)

After signup:

 Copy your API key

---

### Step 2 — Install Required Libraries

pip install openai python-dotenv requests

---

### Step 3 — Store API Key Securely

Create .env file:

OPENAI\_API\_KEY=your\_api\_key\_here

Never hardcode keys in script.

---

### Step 4 — Python Script to Generate Test Cases

## Example: Generate API Test Cases

```
import os

from dotenv import load_dotenv

from openai import OpenAI


load_dotenv()

client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))


api_doc = """
POST /login

Payload: { "username": "string", "password": "string" }

Response: 200 success, 401 unauthorized

"""


prompt = f"""

Generate positive and negative API test cases in table format

for the following API:

{api_doc}

"""

response = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[{"role": "user", "content": prompt},
    temperature=0.3
)

print(response.choices[0].message.content)
```

☞ This prints structured test cases.

---

## ✓ Step 5 — Generate pytest Automation Script

### Python Script to Generate pytest Code

```
prompt = f"""\n\nGenerate pytest API automation script using requests library\nfor the following API:\n\n{api_doc}\n\nInclude:\n- Positive test\n- Negative test\n- Status code validation\n- JSON validation\n\n\nresponse = client.chat.completions.create(\n    model="gpt-4o-mini",\n    messages=[{"role": "user", "content": prompt}]\n)\n\nprint(response.choices[0].message.content)
```

## ✓ Example Output (LLM Generated pytest Script)

LLM will generate something like:

```
import requests\n\n\nBASE_URL = "https://api.example.com"
```

```
def test_login_success():

    payload = {"username": "valid", "password": "valid"}

    response = requests.post(f"{BASE_URL}/login", json=payload)

    assert response.status_code == 200
    assert "token" in response.json()

def test_login_failure():

    payload = {"username": "invalid", "password": "wrong"}

    response = requests.post(f"{BASE_URL}/login", json=payload)

    assert response.status_code == 401
```

---

## Step 6 — Save Generated Code to File

```
with open("test_login.py", "w") as f:
    f.write(response.choices[0].message.content)
```

Now run:

```
pytest test_login.py
```

---

## Advanced Automation Workflow (Real Project)

In real projects:

### 1. Input

- Swagger/OpenAPI JSON
- API docs
- CSV test data

### 2. LLM Generates

- Test cases

- pytest scripts
- Edge cases
- Mock data

### 3. CI/CD Integration

- Auto commit generated scripts
  - Run in Jenkins pipeline
- 

## Best Practices

### Security

- Never expose API keys
- Use environment variables

### Prompt Engineering

Be specific:

 Bad prompt:

Generate test

 Good prompt:

Generate pytest API automation script with fixtures,  
parameterization, and assertions for login endpoint

### Always Validate LLM

LLM code is not perfect.

You must:

- Review generated scripts
  - Fix edge cases
  - Add custom validations
- 

## Interview Answer (Short Version)

If interviewer asks:

 “How do you use LLM to generate pytest scripts?”

You can say:

I use an LLM API with Python by sending API documentation as prompts. The LLM generates structured test cases and pytest automation scripts. I store API keys securely using environment variables, validate generated code, and integrate it into our automation framework. This improves productivity and accelerates test development.

---