DUMPSBOSS.

ISTQB Certified Tester Testing with Generative AI () v1.0

iSQI CT-GenAI

Version Demo

Total Demo Questions: 5

Total Premium Questions: 59

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Introduction to Generative AI	7
Topic 2, Prompt Engineering	18
Topic 3, Generative AI in Software Testing	28
Topic 4, Testing Generative AI Systems	6
Total	59

QUESTION NO: 1

A team uses a vision-language model to review checkout screenshots against visual acceptance criteria. The model reports that all controls are visible and correctly labelled. Which additional test activity is MOST necessary before concluding that the checkout page is accessible?

A. Increase the screenshot resolution and repeat the visual review

B. Verify semantic markup, keyboard operation, and assistive-technology behaviour

C. Accept the result when all visible controls have descriptive text

D. Use screenshots from additional browser window sizes only

ANSWER: B

Explanation:

Verify semantic markup, keyboard operation, and assistive-technology behaviour is correct. A screenshot can provide evidence about visible layout and text, but it does not reliably demonstrate whether controls have appropriate semantic roles, can receive keyboard focus, or are announced correctly by assistive technologies. These properties require inspection and execution-based accessibility testing.

Increasing image resolution may improve visual analysis but does not reveal non-visual interaction behaviour. Repeating the same screenshot review does not provide evidence of keyboard or screen-reader support.

QUESTION NO: 2

Consider applying the meta-prompting technique to generate automated test scripts for API testing. You need to test a REST API endpoint that processes user registration with validation rules. Which one of the following prompts is BEST suited to this task?

A. Role: Act as a test automation engineer with API testing experience. | Context: You are verifying user registration that enforces field and format validation. | Instruction: Generate pytest scripts using requests for both positive (valid) and negative (invalid email, weak password, missing fields) cases. | Input Data: POST /api/register with validation rules for email and password length. | Constraints: Include fixtures, clear assertions, and appropriate response validations. | Output Format: Provide organized pytest scripts.

B. Role: Act as a test automation engineer. | Context: You are creating tests for a registration endpoint. | Instruction: Generate Python test scripts using pytest covering both valid and invalid inputs. | Input Data: POST /api/register with email and password. | Constraints: Follow pytest structure. | Output Format: Provide scripts.

C. Role: Act as an automation tester. | Context: You are validating an API endpoint. | Instruction: Generate Python test scripts that send POST requests and validate responses. | Input Data: User credentials. | Constraints: Include basic scenarios with asserts. | Output Format: Provide organized scripts.

D. Role: Act as a software engineer. | Context: You are testing registration logic. | Instruction: Create Python scripts to verify endpoint behavior. | Input Data: POST /api/register with test users. | Constraints: Add checks for status codes. | Output Format: Deliver functional scripts.

ANSWER: A

Explanation:

Role: Act as a test automation engineer with API testing experience. | Context: You are verifying user registration that enforces field and format validation. | Instruction: Generate pytest scripts using requests for both positive (valid) and negative (invalid email, weak password, missing fields) cases. | Input Data: POST /api/register with validation rules for email and password length. | Constraints: Include fixtures, clear assertions, and appropriate response validations. | Output Format: Provide organized pytest scripts. is the best meta-prompt because it supplies the essential prompt components needed for dependable automated-test generation: a defined expert role, the testing context, explicit tasks, concrete endpoint and validation inputs, implementation constraints, and a requested deliverable format. This specificity guides the model to produce executable pytest code using the appropriate HTTP client library while covering both successful registration and

meaningful validation failures. Naming invalid email formats, weak passwords, and missing fields makes the requested negative testing scope directly traceable to the stated registration rules. Requiring fixtures supports reusable setup, while clear assertions and response validation encourage tests that verify observable API behavior rather than merely submit requests. This structured, constraint-driven approach reflects prompt-engineering practice: providing relevant context, instructions, and output expectations reduces ambiguity and improves the usefulness and consistency of generated artifacts. See the ISTQB Testing with Generative AI certification overview and the pytest fixture documentation.

QUESTION NO: 3

A prompt asks an LLM to classify failed automated tests as product defects, test-script defects, environment failures, or insufficient evidence. Testers observe that the model usually assigns a category even when logs are incomplete. Which prompt improvement BEST addresses this problem?

A. Ask the model to provide the most confident category in every case

B. Define the evidence required for each category and require “insufficient evidence” when that evidence is absent

C. Assign the model the role of a senior test manager

D. Limit the response to one sentence for each failed test

ANSWER: B

Explanation:

Define the evidence required for each category and require “insufficient evidence” when that evidence is absent is correct because it establishes explicit decision criteria and a permitted abstention outcome. This reduces pressure on the model to make a confident classification from incomplete information and makes the generated result easier for a tester to validate.

Asking the model to be more confident reinforces the undesirable behavior. Adding a broad expert role does not define the evidence threshold. Requesting a shorter answer may make review quicker but does not improve the basis for classification.

QUESTION NO: 4

Which competency MOST helps testers steer LLMs to produce useful, on-policy testware?

A. Mastering prompt engineering

B. Configuring network routers

C. Writing low-level device drivers

D. Designing custom CPU instructions

ANSWER: A

Explanation:

Mastering prompt engineering most directly enables a tester to steer an LLM toward useful, on-policy testware. Prompt engineering is the ability to formulate clear, specific, and contextual instructions for a generative AI system. In testing, this means supplying relevant requirements, acceptance criteria, application context, test objectives, constraints, output formats, and organizational quality or security policies. A well-constructed prompt can also establish a role for the model, request traceability to requirements, require assumptions to be stated, and ask for outputs that are suitable for review by a human tester.

This competency is especially important because LLM output is highly dependent on the information and instructions included in its input. Testers use iterative prompting and critical evaluation to refine generated test conditions, test cases, test data, scripts, or documentation so that the result remains aligned with the intended scope and applicable policies. Prompt engineering therefore supports the tester’s responsibility for directing, reviewing, and validating AI-produced artifacts rather than accepting them uncritically. The ISTQB Certified Tester Testing with Generative AI syllabus describes prompt

engineering as a relevant capability for effectively using GenAI in testing. See the ISTQB CT-GenAI certification page and the ISTQB CT-GenAI syllabus resources.

QUESTION NO: 5

What does an embedding represent in an LLM?

A. Tokens grouped into context windows

B. Numerical vectors capturing semantic relationships

C. Logical rules for reasoning

D. A set of test cases for validation

ANSWER: B

Explanation:

Numerical vectors capturing semantic relationships is correct because an embedding converts text, such as a token, sentence, or document chunk, into an ordered list of numerical values in a multidimensional vector space. The values encode patterns learned from language data, so texts with related meanings tend to be located near one another according to a similarity measure such as cosine similarity. This enables an LLM-based system to work with meaning rather than relying only on exact word matches. For example, a query about a software defect can retrieve content referring to a bug, error, or failure even where the wording differs. Embeddings are particularly important in retrieval-augmented generation, where a user query and source-document chunks are embedded and then compared to select relevant context for the model. Microsoft's overview of vector search describes vectors as numerical representations that capture semantic meaning and support similarity search: Azure AI Search vector search overview. The OpenAI API documentation likewise describes embeddings as vectors representing input text, useful for search, clustering, recommendations, and related tasks: OpenAI embeddings guide.