PassQueen
a.k.a dumpsqueen.com

Certified Tester Advanced Level Agile Technical Tester

iSQI CTAL-ATT

Version Demo

Total Demo Questions: 13

Total Premium Questions: 139

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, The Agile Technical Tester’s Role in an Agile Team	50
Topic 2, Test Automation	82
Topic 3, Testing Non-Functional Quality Characteristics	7
Total	139

QUESTION NO: 1

“As the leader of the marketing department, I want to have a content management system so that

my employees can edit and provide quality content to the readers”

Which of the following requirements engineering techniques would be the MOST effective for

identifying and prioritizing user stories for the given Epic?

A. Storyboarding

B. Story mapping

C. Defining Personas

D. Class Diagrams

E. Use Cases

ANSWER: A B

Explanation:

Storyboarding and story mapping are especially effective for refining this content-management epic into valuable, implementable user stories. Storyboarding makes the intended user experience concrete by visualizing the sequence in which marketing employees create, edit, review, approve, and publish content. Discussing those visual steps with stakeholders helps uncover user interactions, business rules, and expected outcomes that can be expressed as user stories. It also provides a shared understanding of what “quality content” means in the operational workflow.

Story mapping organizes the discovered user activities and tasks into a structured end-to-end journey. The team can place high-level activities across the map, break them into detailed stories, and then arrange stories by business value, workflow order, and release priority. This makes it practical to identify a smallest useful release—for example, the essential capabilities needed for staff to draft, edit, and publish content—while scheduling later enhancements in subsequent iterations. Together, these techniques support collaborative elicitation and transparent prioritization around real user value, which is central to agile requirements work. The ISTQB Agile Technical Tester certification scope is described at ISTQB Agile Technical Tester; practical story-mapping guidance is available from Agile Alliance.

QUESTION NO: 2

You have been given the following story

As a shopper

I want to scan my membership card

So that I get all the discounts I'm entitled to receive

Which of the following is the correct use of BDD to design test scenarios?

A. Given that the shopper scans their card
When they checkout
Then they should receive alt the quantity discounts for everything they have purchased

B. As a store clerk
I want to scan a customer's card
So that their total includes their discounts

C. Given that I have scanned my card
I expect to receive my discounts
And an itemized list of what I bought

D. Given that a card is scanned
Then discounts should be applied
When the customer checks out

ANSWER: A

Explanation:

“Given that the shopper scans their card
When they checkout
Then they should receive alt the quantity discounts for everything they have purchased” is the appropriate BDD-style test scenario because it expresses an observable business behaviour using the Given–When–Then structure. The scenario starts by establishing a relevant context: the shopper has scanned a membership card. It then identifies the event that triggers the business outcome, checkout. Finally, it states the expected result: eligible discounts are applied to the shopper’s purchase. This directly supports the story’s intended value, namely that a recognised shopper receives the discounts to which they are entitled.

BDD scenarios are designed collaboratively to make examples of expected behaviour precise and testable. In Gherkin, *Given* describes preconditions or context, *When* describes the action or event under test, and *Then* describes a verifiable outcome. This formulation can be discussed by business and technical stakeholders and can later serve as the basis for executable acceptance tests. The scenario therefore connects the user story’s acceptance expectation to a concrete checkout example. See the Cucumber Gherkin reference and the Agile Alliance definition of behaviour-driven development.

QUESTION NO: 3

You are working on a project to develop an application that allows users to collaborate via bespoke, online whiteboards. The first release, delivering core functionality, highlighted misunderstandings of the user stories between testers, developers and the product owner during sprint development Since that release, the developers have agreed to implement a TDD approach for future software development.

Creation of the product backlog for the second release is underway and you have recommended to the project stakeholders that an atdd approach be adopted for the backlog's user stories. What would be a GOOD REASON for making this recommendation?

SELECT ONE OPTION

A. The team wants to start using BDD and therefore atdd must also be used

B. The test strategy states that automation shall be used to minimize effort for execution of acceptance tests

C. TDD can only work effectively when an ATDD approach is used for the user stories

D. Use of ATDD examples will help address the misunderstandings encountered in release 1

ANSWER: D

Explanation:

“Use of ATDD examples will help address the misunderstandings encountered in release 1” is correct because Acceptance Test-Driven Development is specifically intended to build a common, concrete understanding of each user story before implementation begins. In ATDD, the product owner or business representative, developers, and testers collaborate to define examples of expected behaviour and the corresponding acceptance tests. These examples make ambiguous terms, assumptions, business rules, and boundary conditions visible early, while the story is still being refined in the backlog or planned for a sprint.

For an online-whiteboard product, an example could state precisely what should happen when two collaborators edit the same board, invite another user, or lose connectivity. Agreeing such examples gives all participants the same interpretation of the story and provides clear, testable acceptance criteria. The examples can subsequently guide development and acceptance testing, maintaining traceability from the intended business behaviour to verification. This directly addresses the stated problem: differing interpretations among the product owner, developers, and testers during the first release.

ATDD is a collaborative specification and communication practice, commonly described as “three-amigos” collaboration. It complements an agile team’s technical testing practices by ensuring that implementation work starts from jointly understood customer-facing behaviour. See the Agile Alliance definition of ATDD and the ISTQB Agile Technical Tester certification page.

QUESTION NO: 4

A team uses a data-driven automated test to verify discount calculations for an online store.

Which of the following fields would normally be appropriate in the external test-data file?

A. Product category

B. Basket value

C. Customer membership level

D. Expected discount

E. Browser-driver version

ANSWER: A B C D

Explanation:

Product category, basket value, customer membership level, and expected discount are appropriate fields because they represent inputs used by the discount rule and the expected result against which the automated test can compare the actual calculation. Storing these values externally enables one reusable procedure to execute multiple business scenarios without embedding every data variation in the automation code.

The browser-driver version is an execution-environment detail rather than a normal input or expected result for the discount calculation. The test-data file should support the business rule being tested and provide a clear oracle for each row. See the ISTQB Glossary entry for data-driven testing.

QUESTION NO: 5

A team is introducing automated security checks into its continuous integration pipeline.

Which of the following practices support effective use of these checks?

A. Execute static application security testing early in the pipeline

B. Define a process to triage and remediate findings

C. Maintain rules that are relevant to the technologies used by the product

D. Assume a successful automated scan proves that the product has no security vulnerabilities

E. Disable security rules whenever they cause a build to fail

ANSWER: A B C

Explanation:

Executing static application security testing early in the pipeline, defining a process to triage and remediate findings, and maintaining rules that are relevant to the technologies used all support effective automated security checking. Early automated feedback allows developers to address many vulnerabilities close to the change that introduced them. Triage is necessary because findings need to be assessed for validity, severity, exploitability, and ownership.

Security automation is not a substitute for all other security activities. It does not guarantee the absence of vulnerabilities, and disabling rules merely to obtain a green build can hide genuine risks. OWASP provides further information on static code analysis.

QUESTION NO: 6

Which of the following are appropriate uses of service virtualization in an Agile project?

A. Testing an integration while the real dependent service is still under development

B. Simulating rare error responses from a third-party service

C. Avoiding transaction charges during frequent automated test execution

D. Proving that the real provider meets its production service-level agreement

E. Replacing all tests against the real integration permanently

ANSWER: A B C

Explanation:

Service virtualization is appropriate when a dependent service is unavailable, costly to invoke, or needed to simulate conditions that are difficult to create with the real service. A virtualized service can emulate relevant interface behavior, data handling, response conditions, and state transitions under the control of the test team. This enables earlier, repeatable integration testing without waiting for every real dependency.

A virtualized service does not demonstrate that the real provider has met its contractual obligations, and it should not be treated as a replacement for testing the real integration when that becomes possible. Practical background is available from Tricentis' service virtualization overview.

QUESTION NO: 7

A team has introduced a feature toggle to release a new address-validation service gradually. The old validation behavior remains available while the toggle is disabled.

Which of the following should be considered when designing automated tests for this feature?

A. Verify the relevant behavior with the toggle enabled

B. Verify the existing behavior with the toggle disabled while it remains supported

C. Make the toggle state explicit in the test setup and execution report

D. Allow tests to rely on whichever toggle value is configured in the environment

E. Retain tests for the old behavior indefinitely after the toggle is removed

ANSWER: A B C

Explanation:

Automated tests should verify the expected behavior with the toggle both enabled and disabled where both states are supported. This confirms that the new path works as intended and that the existing path remains protected during the transition. The toggle state should be made explicit in test setup and reporting so that a failure can be diagnosed against the correct configuration.

The team should also plan removal or adaptation of obsolete tests when the old behavior and toggle are retired. Leaving tests that depend on a removed path creates automation debt. Relying on the production default without controlling the toggle state makes tests environment-dependent and unreliable. Feature toggles are configuration items whose state can materially affect observed behavior and test results.

QUESTION NO: 8

An organization uses a third-party identity provider in production. The provider has rate limits and cannot reliably supply every error condition needed for automated integration testing.

Which of the following are suitable capabilities of a virtualized identity service for the test environment?

A. Return controlled successful and unsuccessful authentication responses

B. Emulate relevant response delays and service errors

C. Maintain state for token issue and token refresh flows

D. Eliminate the need to test the integration with the real identity provider

E. Automatically correct authentication defects in the application under test

ANSWER: A B C

Explanation:

A virtualized identity service can return controlled successful and unsuccessful authentication responses, emulate relevant response delays or service errors, and maintain the state needed for flows such as token issue and token refresh. These capabilities let the team test important integration behavior repeatedly without consuming third-party quotas or depending on the availability and behavior of the live provider.

The virtualized service should emulate the agreed external contract sufficiently for the intended tests. It is not a substitute for all testing against the real provider, particularly before release, but it allows early and repeatable testing of conditions that are difficult or costly to obtain from the dependency. See the Tricentis service virtualization overview.

QUESTION NO: 9

What is a virtualized service?

A. A stateless mock service that provides simple responses to requests

B. A software service that is developed by another organization but used in the production software as an integral part of a software product

C. A stateful mock service that appears to provide the same behavior and data handling as the real service without actually performing the processing

D. A set of simple stubs used to provide positive acknowledgements for all messages received

ANSWER: C

Explanation:

A virtualized service is a test double that emulates a dependent service closely enough for the system under test to interact with it as though the real dependency were available. “A stateful mock service that appears to provide the same behavior and data handling as the real service without actually performing the processing” captures the essential characteristics: it provides realistic interfaces, responses, data behavior, and, where needed, state transitions, while avoiding calls to the actual external or unavailable system. This enables testing to continue when dependencies are costly, unstable, incomplete, rate-limited, difficult to configure, or unavailable in the test environment. Service virtualization is especially useful for integration-level testing because the virtual service can reproduce relevant interaction scenarios and data conditions deterministically. The Advanced Technical Test Analyst syllabus includes the use of test doubles and virtualization-related techniques as part of testing systems with dependencies and interfaces. See the ISTQB Advanced Technical Test Analyst certification information at ISTQB Technical Test Analyst and iSQI’s certification overview at iSQI Advanced Technical Test Analyst.

QUESTION NO: 10

You are working in a project that developed a product that has reached a stable state and is

deployed on different HW configurations all over Europe.

You management decided to use your project as Proof of Concept for adopting CI as a new way of

working. The POC was implemented on one set of hardware and was successful.

Which of the following actions is a good next step?

A. Enable different test configurations in the CI process to test different configurations that are deployed in the market

B. Speed up test execution by decreasing the amount of User Interface (UI) testing to get faster feedback from the CI tests

C. Reduce the number of tests in the CI test suite, to improve the benefit of the CI approach

D. Implement code to dynamically select CI tests, executing only test cases affected by changes

ANSWER: A

Explanation:

Enable different test configurations in the CI process to test different configurations that are deployed in the market is the appropriate next step because the successful proof of concept established that CI works on one hardware setup, but the product's operational risk exists across the range of hardware configurations used by customers. The next sensible increment is therefore to broaden the pipeline's test environment coverage in a controlled way. This provides evidence that each integrated build remains compatible with representative deployed configurations and helps reveal configuration-dependent failures early, before a release reaches users.

For an Agile Technical Tester, CI is not only about running tests frequently; it is also about obtaining timely, meaningful feedback on the quality risks relevant to the product. Where hardware diversity is a stated characteristic of the production landscape, test configuration is an essential part of that feedback. A CI pipeline can support this through parameterized jobs, configuration matrices, dedicated agents, or access to suitable physical or virtual test environments. The configurations should be selected using risk, market usage, supported platforms, and known integration sensitivities, then expanded as capacity and evidence justify it. This evolves the proof of concept toward a CI implementation that better represents real deployment conditions. See the ISTQB Agile Technical Tester certification overview and the ISTQB Advanced Level Agile Technical Tester certification page.

QUESTION NO: 11

Consider the following section of pseudocode

```
function getPassword() {
var x,
var y,
var z,
var passwordGood = false
 // Get password from user, user is allowed 3 tries
 do until x = 3
   call getPassword (password)
   if password is good
     x = 3
     passwordGood = true
   else
     x = x + 1
     display "Password is not valid, try again"
   endif
 If passwordGood <> true
```

Display " You exceeded the number of tries to enter a password. Your account is now locked. Call customer.

For this section of code, which of the following issues should be identified during a code review?

1. Variables have not been properly defined with meaningful names

2. There are unused variables defined

3. Divisors are not tested for zero

4. Loop counters are not properly initialized

5. There are endless loops

6. There are statements within the loop that should be outside the loop

A. 1, 3, 4, 5

B. 7, 3, 4, 6

C. 2, 3, 5, 6

D. 1, 2, 4, 6

ANSWER: D

Explanation:

1, 2, 4, 6 is correct because the pseudocode presents several maintainability and control-flow concerns that a technical code review should expose. Meaningful variable names are needed to make the password-attempt logic understandable and maintainable; vague identifiers obscure the purpose and state of the code. A declared variable that is never subsequently used is also a review finding, since it creates unnecessary noise and can mislead maintainers about intended behavior. The loop counter must be initialized before it is used to control the attempt loop, otherwise the number of permitted attempts cannot be reliably enforced. Finally, the lockout message belongs after the loop: it should be displayed once, after all permitted attempts have been exhausted, rather than being executed on each loop iteration. These findings fit the Agile Technical Tester emphasis on static testing techniques, including reviews that assess code quality, maintainability, consistency, and control logic early in development. The relevant certification scope is described by ISTQB's Agile Technical Tester certification; iSQI also describes the qualification and its technical-testing focus at iSQI Agile Technical Tester.

QUESTION NO: 12

When using a process-compliant approach to testing a safety-critical project what is an important aspect of test automation?

A. It must provide exhaustive regression testing

B. It must provide traceability back to the requirements and results documentation

C. It must implement automated checklists

D. It must incorporate model-based testing

ANSWER: B

Explanation:

It must provide traceability back to the requirements and results documentation. In a safety-critical, process-compliant project, test automation is not valuable solely because it executes tests quickly; it must also produce objective, reviewable evidence that specified requirements have been verified and that recorded results can be linked to the relevant test cases, procedures, configurations, and requirements. This traceability supports compliance activities, impact analysis when requirements or implementations change, repeatable regression testing, and audits or safety assessments. Automated test tooling should therefore preserve the associations between the requirement being checked, the automated test implementation and version, the execution environment, the test data, and the resulting pass/fail evidence. Such records allow an assessor to determine what was tested, how it was tested, and whether the applicable requirement has evidence of verification. The ISTQB Advanced Level Agile Technical Tester syllabus emphasizes that automation must be selected and implemented with suitable attention to traceability, reporting, maintainability, and the project context, particularly where compliance is relevant. The broader ISTQB glossary also defines traceability as the ability to identify related items in documentation and software artifacts. See the ISTQB Advanced Level Agile Technical Tester certification page and the ISTQB glossary definition of traceability.

QUESTION NO: 13

What is the primary purpose of the debriefing meeting when exploratory testing is used in an Agile setting?

A. To identify defects

B. To define the charier for the test

C. To provide a status of the progress and coverage of the session

D. To review the actions of the tester and determine if there were any errors made during the testing

ANSWER: C

Explanation:

“To provide a status of the progress and coverage of the session” is correct because a debriefing is the structured feedback point following an exploratory test session. Exploratory testing combines learning, test design, and execution, so the team needs a concise account of what was investigated, which areas, risks, and test ideas were covered, what remains uncertain, and how much of the planned session was completed. This information makes the testing work visible to the Agile team and supports timely decisions about follow-up testing, prioritization, and whether additional effort is needed in particular product areas.

In session-based exploratory testing, the tester records session information such as the charter, time spent, coverage, observations, and issues encountered. The debriefing uses these records to communicate an accurate status of the session and establish a shared understanding of the achieved coverage. This fits Agile’s frequent feedback model: the result is not merely a list of activities, but useful evidence about the product and the testing performed. ISTQB defines exploratory testing as an approach in which tests are dynamically designed and executed based on the tester’s knowledge and observations; this makes reporting session progress and coverage especially important. See the ISTQB Glossary: exploratory testing and the ISTQB Agile Technical Tester certification page.