DUMPSARENA

UiPath Test Automation Engineer Professional v1.0 Exam

UiPath UiPath-TAEPv1

Version Demo

Total Demo Questions: 13

Total Premium Questions: 133

Buy Premium PDF

https://dumpsarena.co

sales@dumpsarena.co

sales@dumpsarena.co
dumpsarena.co

Topic Break Down

Topic	No. of Questions
Topic 1, Test Automation Concepts	13
Topic 2, Test Automation Project Design	30
Topic 3, Test Automation Implementation	28
Topic 4, Test Automation Execution	55
Topic 5, Test Automation Maintenance	7
Total	133

QUESTION NO: 1

What is the difference between RPA testing and application testing?

A. RPA testing and application testing are the same thing, the term used simply depends on the specific software involved.

B. RPA testing involves checking the quality of robotic processes, whereas application testing checks the functionality of software applications.

C. RPA testing checks the functionality of a software application, but application testing evaluates the performance of robots.

D. RPA testing evaluates human interactions with software, while application testing checks robotic processes.

ANSWER: B

Explanation:

RPA testing involves checking the quality of robotic processes, whereas application testing checks the functionality of software applications. In UiPath, test automation can validate both an application and an automation workflow, but these are distinct testing targets. Application testing focuses on whether the underlying software behaves as required: for example, whether screens, services, business rules, and user-facing functions produce expected results. RPA testing focuses on the automation that performs a business process through applications, including whether the robot follows the intended workflow, handles data correctly, interacts reliably with target interfaces, and produces the expected process outcome. This distinction matters because a stable application can still be automated incorrectly, while a well-designed automation can encounter failures caused by changes or defects in the application it uses. UiPath Test Suite supports testing through reusable test cases, test data, and execution reporting, enabling teams to validate automations and the applications or processes involved in their delivery lifecycle. UiPath describes Test Suite as a platform for testing automations and applications, with support for managing, executing, and analyzing tests. See the UiPath Test Suite introduction and UiPath automation project documentation.

QUESTION NO: 2

Which are the types of coded automations in UiPath?

A. Workflows, Test cases, Source files.

B. Coded workflows, Coded test cases, Coded functions.

C. Coded workflows, Coded test cases, Code source files.

D. Coded workflows, Coded databases, Code source files.

ANSWER: B

Explanation:

Coded workflows, Coded test cases, Coded functions. is correct because UiPath Studio supports these three coded-automation templates for implementing automation and testing logic in C#. A coded workflow is the code-based counterpart to a workflow and can use UiPath services and activities through the coded-automation APIs. A coded test case provides the structure for automated testing: it can define test logic, use assertions, and be executed as part of a UiPath test project. A coded function is intended to encapsulate reusable coded logic that can be invoked from other automations, helping teams organize common implementation into maintainable units. These templates allow developers to work with code while retaining integration with UiPath projects, dependencies, execution, and testing capabilities. UiPath documents the available coded-automation templates and their intended uses in its Studio documentation: Coded automations. For details on creating and using reusable code-based functionality, see Coded functions.

QUESTION NO: 3

How can activity coverage scenarios for different automation needs be created in UiPath?

A. Create one comprehensive test case that covers all scenarios in one execution.

B. Alter the workflow to ensure a single test case covers all scenarios.

C. Create separate test cases to cover each scenario during the execution.

D. Use the same test case, but execute it in different workflows.

ANSWER: C

Explanation:

Create separate test cases to cover each scenario during the execution. UiPath Test Suite uses test cases as independent, maintainable units for validating automation behavior. Defining a distinct test case for each relevant scenario lets the automation be exercised with the inputs, conditions, and expected outcomes that apply specifically to that scenario. This provides focused activity coverage, including coverage of different workflow paths and business conditions, while keeping test results easy to identify and analyze.

Separate test cases can be organized and run together in test sets when a broader execution is needed, without combining all validation logic into one difficult-to-maintain test. They can also be data-driven where multiple input variations are appropriate, helping teams expand coverage in a controlled and traceable way. This structure supports clear reporting in Test Manager and makes it easier to update the validation for one scenario without affecting the tests for unrelated scenarios. UiPath describes test cases as the fundamental artifacts used to test automations and supports grouping them into test sets for execution. See UiPath Test Cases documentation and UiPath Test Sets documentation.

QUESTION NO: 4

What type of data can be added to the Tester Comment field?

A. Text and Screenshot

B. Text and documents

C. Text and Video recording

D. Only text

ANSWER: D

Explanation:

The correct answer is **Only text**. In UiPath Test Manager, the Tester Comment field is intended for a tester to enter written context about a test execution, such as observations, execution notes, clarifications, defect-related details, or information that helps other project members understand the outcome. The field stores a textual comment associated with the test activity; it is not an upload control or a container for embedded media. Test Manager distinguishes written execution information from artifacts that may be managed through the relevant test result, defect, or evidence workflows. This makes comments easy to review in the execution history and supports traceability by preserving the tester's narrative alongside the recorded execution status. When completing manual tests, testers should use concise but meaningful text in this field to document what occurred, including relevant values, environmental conditions, or reproduction notes where appropriate. UiPath's Test Manager documentation describes the execution and result-management capabilities in its Test Manager user guidance: UiPath Manual Tests documentation and UiPath Test Results documentation.

QUESTION NO: 5

What programming language can be used in Coded Workflows?

A. VB.Net

B. C#

C. C# and VB.Net

D. VBScript

ANSWER: B

Explanation:

C# is the language used for UiPath Coded Workflows. A coded workflow is a class-based automation artifact created in a C# UiPath project, where the workflow class inherits from `CodedWorkflow`. This provides a code-first way to create automation logic while retaining access to UiPath services, activities, object repository elements, and testing capabilities through the Studio coding APIs. Developers write methods, use standard C# syntax and .NET features, and can call activities or other workflow components from the coded workflow. The project language is significant: UiPath's documentation identifies coded automations, including Coded Workflows, as a capability for C# projects. Therefore, selecting C# accurately reflects the supported programming language for this Studio feature in the context of the exam. See UiPath's Coded Workflows documentation and its automation project documentation for the project-language and coded-automation requirements.

QUESTION NO: 6

What does the Run All in View action in the Test Explorer panel do?

A. It runs tests that are currently in view through all filters.

B. It runs only failed test cases.

C. It runs all test cases and workflows.

D. It runs only selected test cases and workflows.

ANSWER: A

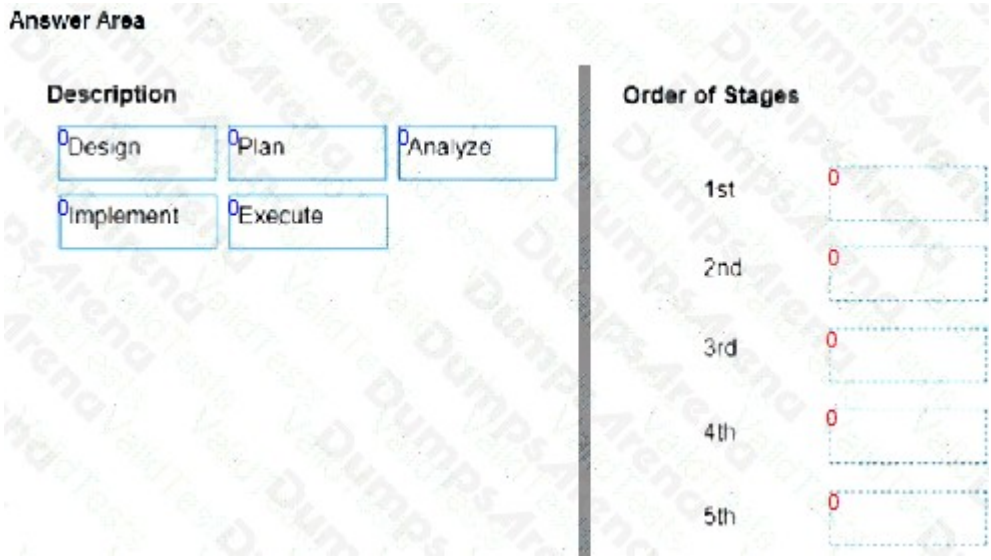
Explanation:

It runs tests that are currently in view through all filters. In Test Explorer, the visible test list reflects the current view, including any filtering and navigation context that has been applied. Using *Run All in View* starts execution for that displayed set, allowing a tester to run a focused group of test cases without having to select every item individually. This is especially useful when the project contains many test cases but the tester wants to execute only those returned by a particular search, status filter, tag-related view, or selected location in the Test Explorer hierarchy. The action is therefore based on what Test Explorer is presently showing, rather than being a command to execute every test asset in the entire project. UiPath documents Test Explorer as the Studio panel for organizing, filtering, and running test cases, with commands that support execution from the current testing context. For details on working with the panel and its execution actions, see the UiPath Test Explorer documentation. Additional guidance on creating and running automated tests is available in the UiPath automated testing overview.

QUESTION NO: 7 - (SIMULATION)

What is the correct order of stages of the Testing process?

Instructions: Drag the Description found on the left and drop on the correct order of stages found on the right.



ANSWER: See the explanation for the answer

Explanation:

The correct order of the testing process stages is:

1. **Plan**
2. **Design**
3. **Implement**
4. **Execute**
5. **Analyze**

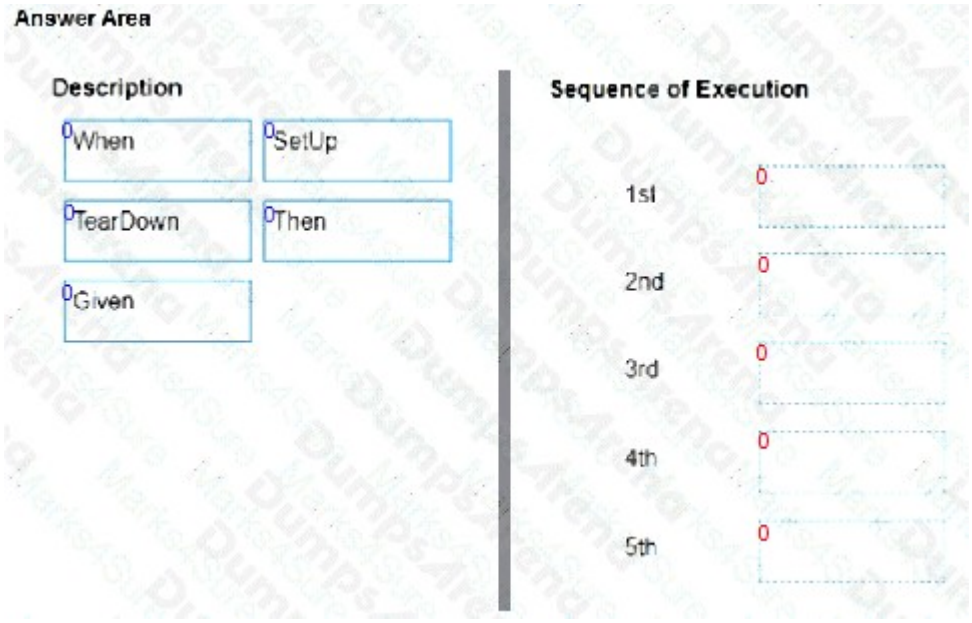
Drag the descriptions to the order fields as: `1st Plan`, `2nd Design`, `3rd Implement`, `4th Execute`, `5th Analyze`.

QUESTION NO: 8 - (DRAG DROP)

DRAG DROP

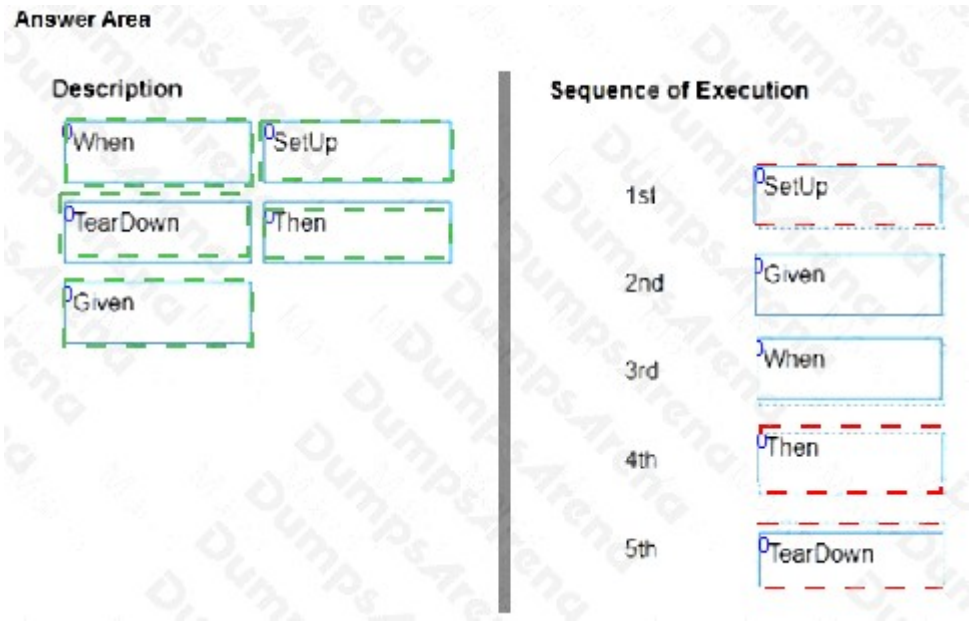
What is the sequence of execution of the main blocks of a Test Case that has Test Automation Framework attached?

Instructions: Drag the Description found on the left and drop on the correct Sequence of Execution found on the right.



ANSWER:

Answer:



Answer:

Explanation:

A UiPath Test Automation Framework test case follows the familiar behavior-driven flow from preparing the test through validation and cleanup. The sequence starts with **SetUp**, which prepares the test environment and any shared prerequisites needed before the test scenario begins. This can include opening an application, initializing data, navigating to an initial state, or creating required test resources. After preparation, **Given** establishes the specific starting context or precondition for the scenario being tested. The next block, **When**, contains the action or event that is being exercised. This is the part of the test that performs the business operation under test, such as submitting a form or triggering a workflow. **Then** follows the action and verifies the expected outcome through validations or assertions. Finally, **TearDown** runs after the scenario to clean up resources and leave the environment in an appropriate state, such as closing applications or removing test data. Thus, the correct execution sequence is SetUp, Given, When, Then, and TearDown. UiPath documents test automation concepts and test case execution in its Test Automation documentation, while the broader given-when-then structure is described by Cucumber's Gherkin reference.

QUESTION NO: 9

What is the BDD test case template used for in UiPath?

A. To handle Global Exception Handlers in behaviour driven development approach.

B. To create test cases based on If-Then-Else containers.

C. To structure the test around If-Then-Else statements.

D. To structure the test around Given-When-Then containers.

ANSWER: D

Explanation:

The BDD test case template is used to structure the test around Given-When-Then containers. Behavior-driven development expresses an expected behavior in a readable, business-oriented sequence: *Given* defines the initial context or preconditions, *When* captures the action or event being tested, and *Then* verifies the expected outcome. In UiPath Studio, selecting the BDD Test Case template provides this organization directly in the workflow, helping teams create tests that clearly communicate both the scenario and its acceptance criteria. This format is particularly useful when automation developers, testers, and business stakeholders need a shared and easily understandable description of what the automation should do. Each part of the scenario can contain the relevant UiPath activities, while the overall test remains aligned with the

established BDD pattern. UiPath documents BDD test cases as a template that uses Given, When, and Then annotations/containers to organize test logic and support behavior-focused testing. See UiPath Test Case Templates documentation and UiPath Studio Test Automation documentation.

QUESTION NO: 10

Which of the following descriptions matches the concept of Integration Testing?

A. Evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

B. Validates the complete and fully integrated system. The purpose of a system test is to evaluate the end-to-end system specifications.

C. Individual units of a system are tested. The purpose is to validate that each unit of the system code performs as expected.

D. Individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

ANSWER: D

Explanation:

Individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. This accurately describes integration testing because this test level examines the interfaces, data flow, dependencies, and communication behavior that occur when separately developed or separately tested components work together. A component may behave correctly in isolation, yet failures can arise after integration because of incorrect assumptions about input or output formats, API contracts, timing, error handling, or shared resources. Integration testing is therefore concerned with verifying that the connections between units produce the expected behavior, rather than evaluating a unit independently or assessing the whole solution against end-to-end requirements. In UiPath test automation, this principle applies when automations, workflows, applications, APIs, queues, or other dependent components are connected and their interaction must be validated as a combined process. UiPath's testing capabilities support organizing and executing tests across different test levels as part of a quality-assurance strategy. See UiPath's Test Automation documentation and the ISTQB definition of integration testing.

QUESTION NO: 11

What happens once a Test Set execution starts in Test Manager?

A. A test execution container is created without a name, and an empty test result entry, called a test case log. is added for each Test Case.

B. A test execution container with the same name as the Test Set is created, and for each Test Case, an empty test result entry, called a test case log. is added.

C. An empty test result, called a Test Case log. is created, but no test execution container is formed.

D. A Test Case log is created for each executed Test Case, and an empty test result entry is added to an unnamed test execution container.

ANSWER: B

Explanation:

Once execution is initiated for a Test Set, UiPath Test Manager creates a test execution container named after that Test Set. This container acts as the parent record for that particular execution run, allowing Test Manager to organize the run's status, traceability information, and results in one place. At the same time, Test Manager creates an initially empty result record for every Test Case included in the Test Set. These individual records are called test case logs and are subsequently populated with execution details as the run proceeds, such as the outcome, timestamps, and available diagnostic information. Creating the container and all associated test case logs at the start gives the execution a complete, auditable structure even before every test has completed. This behavior supports clear reporting at both the overall Test Set level and the individual Test

Case level. UiPath documents the execution tracking model in its Test Set executions documentation and provides broader context for execution reporting in the Test results documentation.

QUESTION NO: 12

Consider the following snippet:

What should be improved to ensure the Test Case will not fail with exception?

A. In the Verification point '_autogenerated_TextString' should be removed.

B. The ObjectRepository should be changed so all the elements fall into Home screen.

C. The ApplicationPath should be provided dynamically.

D. Click 'OK' should be removed.

ANSWER: C

Explanation:

The ApplicationPath should be provided dynamically. An application executable path is environment-specific: it can differ between developer workstations, test machines, virtual desktops, or CI agents, and it can change after an application upgrade or relocation. Supplying the path through a Test Case or workflow argument, a configuration value, or an environment-specific asset lets the same automation resolve the appropriate executable at runtime rather than relying on one fixed local path. This makes the test portable and reduces the risk that the application-launch activity throws an exception because its target file cannot be found. UiPath arguments are designed to pass values into and out of workflows, making them a suitable way to inject a deployment-specific application path while keeping the test logic reusable. The path can also be maintained as external test/configuration data so it can be changed without editing the workflow itself. See UiPath's guidance on workflow arguments and its overview of Studio projects for reusable, configurable automation design.

QUESTION NO: 13

What are the tasks that can be executed from the UiPath command-line interface (CLI) while creating a CI/CD pipeline?

A. Job, Run, Deploy, Delete, Analyze.

B. Asset Job, Package, Test, Help, Version.

C. Analyze, Delete, Deploy, Pack.

D. Run, Test, Deploy, Pack, Help, Version.

ANSWER: D

Explanation:

Run, Test, Deploy, Pack, Help, Version. is correct because UiPath CLI supports the key automated lifecycle actions needed in a CI/CD pipeline. The *pack* command creates a deployable UiPath automation package from a project, allowing a build agent to produce a versioned artifact. The *analyze* capability, where used in pipeline design, performs workflow analysis against configured rules. The *test* command executes automated tests and can publish test results, supporting quality gates. The *deploy* command publishes a package to UiPath Orchestrator, while *run* starts executions or test runs through the configured Orchestrator connection. In addition, CLI help output enables users and build agents to discover supported command syntax, and the version command identifies the installed CLI release, which is useful for reproducible pipeline environments and troubleshooting. Together, these commands cover package creation, validation and testing, publishing, execution, and operational CLI support. UiPath documents the available command-line tasks and their usage in its UiPath CLI documentation, with pipeline-oriented packaging and deployment guidance available in the CI/CD integrations documentation.