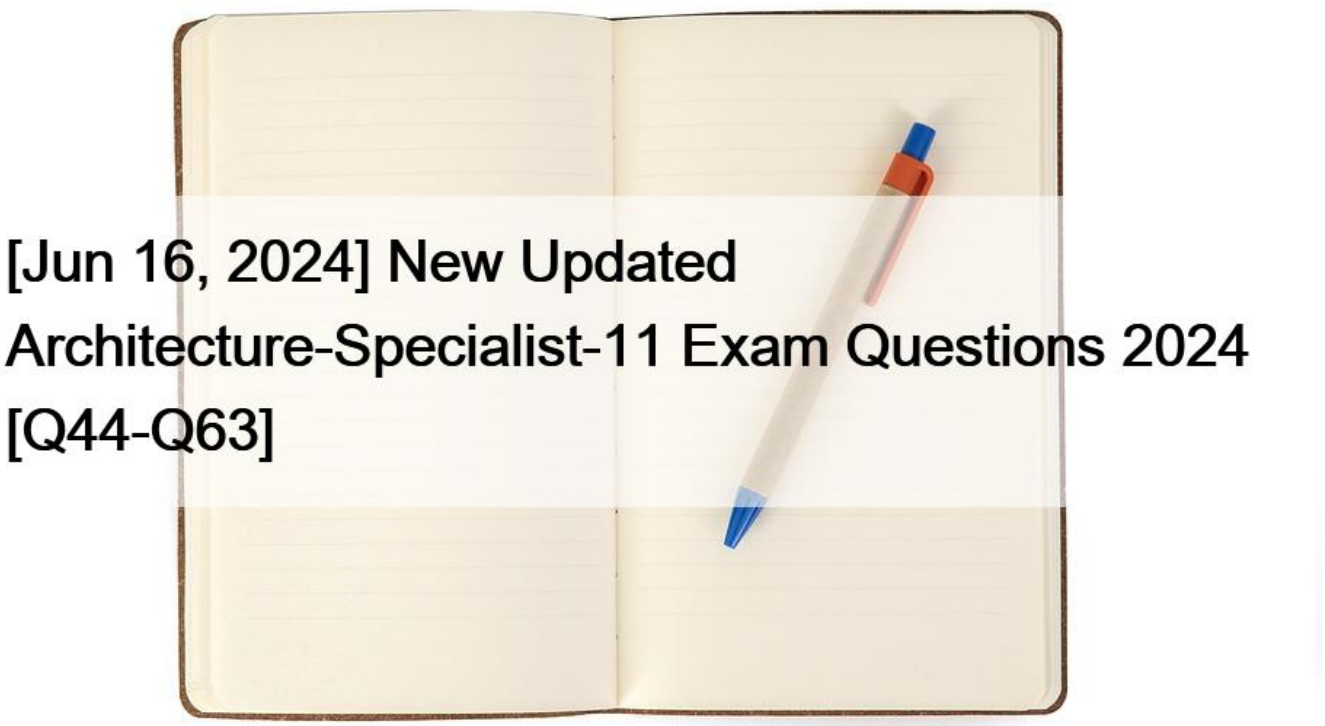
[Jun 16, 2024 New Updated Architecture-Specialist-11 Exam Questions 2024 [Q44-Q63



[Jun 16, 2024] New Updated Architecture-Specialist-11 Exam Questions 2024
Updated Free OutSystems Architecture-Specialist-11 Test Engine Questions with 85 Q&As

Q44. _CW module is for
* Logic to Synchronize data in CS’s with an external system. Isolating this logic makes the CS completely system agnostic and it’s easier to decouple or replace the external system.
* A BL becomes a Calculation Engine if it performs complex calculations, (e.g. an invoice calculation engine or an insurance simulator). Engines are usually subject to versions.
* Technical wrapper to expose an API to External consumers, keeping core services system agnostic and supporting multiple versions of the API.
* Core Widgets (blocks), to manage complexity, composition or to have its own lifecycle.
* Reusable Core Services with public entities, actions, and blocks.

Q45. What is the new name for Architecture Dashboard?
* Architecture Validation
* Architecture Canvas
* Architecture Framework

* AI Mentor

Q46. Considering Architecture Dashboard(AI Mentor), which of the following sentences is FALSE?
* Architecture Dashboard allows you to automatically apply a solution to solve a found code pattern.
* Architecture Dashboard performs a code analysis that uncovers patterns related to performance, security, architecture and maintainability.

Q47. Which of the below is NOT part of the three step process of Architecture Design Process?
* Assemble
* Shape
* Disclose
* Organize

Q48. In which Architecture Canvas layer do you expect to have a higher reusability rate?
* End-User layer
* Core layer
* Foundation layer

Q49. What is the common naming convention for a mobile version of a CS module?
* M_CS
* _CS
* mobile_CS
* m_cs
* m_CS

Q50. Concepts are assembled into Modules in the Assembl are not principles in the Assemble step.
* Don’t join concepts with different lifecycles
* Apply known design patterns
* N/A
* Join conceptually-related concepts
* Isolate reusable logic from Integration logic

Q51. There are 3 common scenarios for Sharing a Style Guide. Which of the below is not part of th scenario.
* Intranet (Single Sign On): Own Menu, Common Login Flow. Menu is defined in the Custom Template, but Login is defined in the Custom Theme. Application reference the Custom Them which picks up the Login.
* Enterprise Apps: Common Menu, Common Login Flow. Menu is defined in the Custom Theme, but Login is defined in the Custom Template.
* Independent Apps : Own Menu, Own Login Flow. Login and Menu is defined in the Custom Template. Applications reference to its own Application Theme thus do not use the Login and Menu in the Custom Template.
* Portal : Common Menu, Common Login Flow. Login and Menu is defined in the Custom Them Application Theme reference the Custom Theme thus have a shared menu and login flow.

Q52. The Architecture Canvas is a …
* framework to support Architecture Conventions for Modules in Outsystems
* framework to support application architecture design in Outsystems
* framework to support Architecture Design Process in Outsystems

Q53. Consider the common style guide scenarios presented in this course. Which of the following statements is true?
* The “specialize a built-in Style Guide” scenario should be used when the changes to the base theme are not extensive.

* The “clone a built-in Style Guide” should be used when you want to extend an existing theme.
* Build-your-own Style Guide should be used when the changes in the theme are not extensive.

Q54. Which of the below matches the most to Core Module Pattern – ECS with Isolated Synchronization Logic Pattern…
* Same as ECS with local replica but synchronization logic is separated. Pro: Code independence. Consumers of CS is not affected by Sync. Sync can orchestrate several CS
* … Entity is exposed as read-only and API is available to centralize business logic for entity creation/update
* … a wrapper used to contain the logic, actions and data that will expose code that is inside of O external library or to inspect external database and import the data structures so they can be used as entities inside of OS
* Same as Base ECS pattern, but have a local replica. Store data to serve as a local cache. Pro: Leverage Entity Use, Simpler Integration API. Con: Less impact on source system
* … caches only summary data that is frequently lister, joined or searched. Full detail for a O single entry is fetched directly from external system. Use when whole database too big or costly to synchronize. Details are only required for single entities (not lists)
* … Entity is not in Outsystems but in an external ERP system. IS just makes remote call to p external system/database. No data is being kept inside OS. Data retrieval may not be optimized as it needs to traverse two different systems to get the information back. Con: Integration API must support all use cases
* … tries to fetch data from local cache entity, if not there, get single entry from the external v system. Cache only that record (read-through caching) Use when whole database too big or costly to synchronize. Integration only touches a small portion of the database. Avoid if access to lists of data is needed up front
* … is a pattern with two modules, a connector module that can be used to encapsulate an) external API with the input/output structures and a wrapper module to expose the normalized API to the consumers.
* Same as ECS with local replica but API module is provided. So any changes to the external system can notify OS, which OS then gets update from the ERP system (subscription system)
* … is needed if data is coming from MULTIPLE external systems. IS will decide which driver to use depending on the data.

Q55. What is NOT a best practice for Mobile Application Architecture: transactions & granularity?
* Have long synchronizations in a single transaction. Better UX as app does not need to sync all the time. Is prepared for constant offline or device standby
* Ensure order and sync granularity. Sync incrementally by entity with partial commit. This way O synchronizations is prepared for constant interruptions and allow retries without repeating the entire synchronization from the start.

Q56. Which of the following recommendations should be applied to the Core layer?
* Core modules should have Front-end Screens for testing purposes.
* Core modules should not have Core Entities.
* Core modules should have public Read-only Entities.
* Core modules should not have business logic.

Q57. ISO/IEC 25010:2011 adds two more aspect to ISO/IEC 9126:1991. Which of the below is not part of it?
* Compatibility : Compatibility was also added as a way to ensure that a piece of software can work together with other systems
* Security : Security ensure software solutions can protect information and data
* Integration : Integration to allow data exchange with other software

Q58. Which of the following is NOT a benefit of having well-defined application architecture?
* Poor service abstraction
* Reduces costs
* Reduces risk
* Supports planning

Q59. Of the options below, which one is a benefit of adopting the Architecture Canvas?
* It’s an automatic way to find and fix architecture issues.
* It’s a systematic approach to architecture design
* It promotes the business users’ collaboration and understanding
* It’s a faster architecture design

Q60. Which of the below matches the most to Core Module Pattern – ECS with Local Replica Pattern…
* … Entity is not in Outsystems but in an external ERP system. IS just makes remote call to p external system/database. No data is being kept inside OS. Data retrieval may not be optimized as it needs to traverse two different systems to get the information back. Con: Integration API must support all use cases
* … is a pattern with two modules, a connector module that can be used to encapsulate an O external API with the input/output structures and a wrapper module to expose the normalized API to the consumers.

p Same as ECS with local replica but synchronization logic is separated. Pro: Code independence. Consumers of CS is not affected by Sync. Sync can orchestrate several CS
* … a wrapper used to contain the logic, actions and data that will expose code that is inside of external library or to inspect external database and import the data structures so they can be used as entities inside of OS
* … caches only summary data that is frequently lister, joined or searched. Full detail for a single entry is fetched directly from external system. Use when whole database too big or costly to synchronize. Details are only required for single entities (not lists)
* … Entity is exposed as read-only and API is available to centralize business logic for entity creation/update
* Same as Base ECS pattern, but have a local replica. Store data to serve as a local cache. Pro: Leverage Entity Use, Simpler Integration API. Con: Less impact on source system
* … tries to fetch data from local cache entity, if not there, get single entry from the external system. Cache only that record (read-through caching) Use when whole database too big or costly to synchronize. Integration only touches a small portion of the database. Avoid if access to lists of data is needed up front
* … is needed if data is coming from MULTIPLE external systems. IS will decide which driver to use depending on the data.
* Same as ECS with local replica but API module is provided. So any changes to the external system can notify OS, which OS then gets update from the ERP system (subscription system)

Q61. “Spaghetti Architecture” is also known as ‘tightly coupled architecture’ or ‘brittle architecture’. Which is NOT a reason why is “Spaghetti Architecture” bad?
* One small change in a component cause a series of cascading effects on other components
* Hinder future changes as they become less flexible and difficult to manage
* Any changes is maintainable and easy to deploy

Q62. Which of the below is NOT a weak dependency as of OS11?
* Screens
* Static Entities
* Server Actions
* Database Entities
* Service Actions
* Local Storage Entities
* Structures

Q63. Which of the below is NOT a suitable advice for designing a Parallel mobile local storage?
* Allow table dependency : Normalize tables to promote parallel data fetch
* Use Fetch Data : Avoid cascading aggregates in Onlnitialize and OnReady
* Avoid generic tables : Contains too much data, not all are relevant

Try 100% Updated Architecture-Specialist-11 Exam Questions [2024:
https://www.vceprep.com/Architecture-Specialist-11-latest-vce-prep.html]