DUMPSBOSS.

Certified Pega Lead System Architecture (LSA) Exam 23

Pegasystems PEGACPLSA23V1

Version Demo

Total Demo Questions: 6

Total Premium Questions: 65

Buy Premium PDF

https://dumpsboss.co

support@dumpsboss.co

support@dumpsboss.co
dumpsboss.co

Topic Break Down

Topic	No. of Questions
Topic 1, Application Design	19
Topic 2, Case Management	7
Topic 3, Data Model	10
Topic 4, Integration	5
Topic 5, Security	9
Topic 6, Deployment	8
Topic 7, Performance	4
Topic 8, System Administration	3
Total	65

QUESTION NO: 1

Dynamic class references (DCR) are one approach to creating a data model in a Pega implementation. Which one of the following best describes the DCR approach?

A. Associating objects with classes at runtime based on user input or certain conditions, allowing for flexibility in the structure of the data model.

B. Linking different classes together in a data model to establish relationships and associations between various entities.

C. Dynamically adjusting the attributes of a class without modifying its structure, facilitating agile data management.

D. Referencing classes that are predefined and unchangeable, ensuring consistency and reliability in the data model.

ANSWER: A

Explanation:

Associating objects with classes at runtime based on user input or certain conditions, allowing for flexibility in the structure of the data model, correctly describes dynamic class referencing. This approach is appropriate when the concrete class of a page or object cannot be determined at design time because it depends on information available only while the application is processing a case. For example, a common business process can select an appropriate specialized class after a user identifies a product, service, or claim category. The application can therefore work with a shared conceptual model while using the class that is appropriate for the specific runtime context.

Dynamic class referencing helps keep a design adaptable where multiple related object types must be handled through common processing. Rather than hard-coding every possible class into a single static structure, the implementation determines the relevant class reference from data or processing conditions. The selected class can then provide the properties, rules, and behavior required for that particular object type. This supports a scalable data model when new specialized types may be introduced over time, while retaining shared abstractions and reuse. Pega's data-modeling learning material discusses modeling data and selecting appropriate class structures; see Pega Academy: Data Modeling and the Pega Documentation portal.

QUESTION NO: 2

A bank application creates a Dispute case when a customer challenges a card transaction. The dispute investigation must retain the merchant name, transaction amount, and authorization details exactly as they were when the case was created, even if the card-processing system later corrects or enriches the transaction record. Which data access approach is the most appropriate?

A. Use the System of Record data access pattern so that the case always retrieves the latest transaction details from the card-processing system.

B. Use the snapshot data access pattern and copy the required transaction information into the Dispute case when it is created.

C. Store the transaction information as properties in the work pool class so that all dispute cases inherit the same values.

D. Use a node-level data page so that every dispute case shares one cached transaction record.

ANSWER: B

Explanation:

Use the snapshot data access pattern and copy the required transaction information into the Dispute case when it is created. The decisive requirement is that the investigation must retain the original transaction context even when the external system of record subsequently changes its data. Storing a snapshot in the case preserves the historical values used for the dispute decision and audit trail.

Using a System of Record pattern would retrieve the current external record when data is accessed, which could replace the original transaction context with amended values. A data page can support either pattern, but the required business behavior is determined by whether the case needs current data or a point-in-time copy.

QUESTION NO: 3

A financial-services application contains investigation cases that include sensitive customer details. Regional supervisors must be able to identify that a case exists in their own region and view limited case context, while only authorized investigators in the same region can open the full case. Which two security configurations best meet this requirement? (Choose Two)

A. Create an ABAC Discover policy that permits supervisors to discover cases in their assigned region.

B. Create an ABAC Read policy that permits full case access only to authorized investigators in the matching region.

C. Grant all supervisors an access role that allows unrestricted read access to investigation cases.

D. Route all investigation cases to a regional work queue and rely on work-queue membership for security.

E. Store the sensitive customer details only in a hidden section of the case user interface.

ANSWER: A B

Explanation:

An ABAC Discover policy can allow regional supervisors to discover cases that meet the regional condition and view limited, non-sensitive case information. Discover access is suitable when a user requires operational awareness without full access to the case contents.

An ABAC Read policy can then control access to the complete case by evaluating attributes such as the operator's region, role, and investigator authorization. This separates limited discovery from permission to read sensitive details. Granting all supervisors broad read access or relying only on work-queue membership would not enforce the required attribute-based regional and role conditions.

QUESTION NO: 4

An application uses a queue processor for asynchronous notification processing and a Job Scheduler for a nightly reconciliation task. The deployment architect must ensure that the appropriate node types are available in the production cluster. Which two node types are required? (Choose Two)

A. BackgroundProcessing node

B. JobScheduler node

C. WebUser node

D. Search node

E. BIX node

ANSWER: A B

Explanation:

A BackgroundProcessing node is required to process queue processor items. Queue processors execute asynchronous work in the background and depend on nodes configured for background processing.

A JobScheduler node is required to run Job Scheduler rules. Job Scheduler processing is performed by nodes with the Job Scheduler node type enabled. WebUser nodes support interactive browser requests, but they are not the required processing nodes for these two background-processing capabilities.

QUESTION NO: 5

A Pega Platform application designed for customer onboarding in a bank involves multiple subprocesses and extensive data entry and retrieval. With over 15 screens to navigate, the Quality Assurance team faces challenges in thoroughly testing the

application. Each bug fix requires the team to restart testing from the beginning, adding to the difficulty. What suggestion do you adopt as a Lead System Architect to assist the team?

A. Suggest that the team begins recording with the screen recording feature of the browser, and then replay the recording when testing the application after performing bug fixes.

B. Encourage the team to write utilities to speed up the testing process.

C. Advise the team to create Pega test suites to run the bulk test cases in one go and speed up the testing process.

D. Advise the team to begin recording the testing performed using the scenario testing feature, and then replay the recording when testing the application after performing bug fixes.

ANSWER: D

Explanation:

Advise the team to begin recording the testing performed using the scenario testing feature, and then replay the recording when testing the application after performing bug fixes. Scenario testing is designed to automate end-to-end user journeys through a Pega application. A tester records interactions with the application, including navigation, entering values, and validating expected outcomes across the screens in a case workflow. The resulting scenario test can then be run again after a rule change or bug fix, providing a repeatable regression test for the complete onboarding flow.

This is especially appropriate when the quality-assurance challenge is repeated manual navigation through many screens and subprocesses. Recording the representative customer-onboarding journey converts that effort into an automated UI-level test, enabling the team to verify that the process still behaves as expected after changes without manually restarting the entire journey. Scenario tests can also be maintained as the application evolves and incorporated into a broader regression-testing practice. Pega documents scenario testing as a capability for validating application behavior by recording and replaying user interactions; see Pega Platform documentation: Scenario testing and Pega Academy: Scenario testing.

QUESTION NO: 6

As a Lead System Architect, one of your principal duties involves advocating for the reuse of assets. You have the authority to harness the features of the Library in App Studio. Which two of the following statements best encapsulate the advantages of the Library? (Choose Two)

A. Update the existing components to make it more specific to the current application.

B. Perform create, read, update, and delete operations on the library components.

C. Manage or add components to your application to enable new features.

D. Use the library to optimize your application creation by reusing assets from all the existing applications developed in your Pega ecosystem.

ANSWER: C D

Explanation:

The Library in App Studio supports asset reuse by making reusable components available for application teams to discover, add, and manage while designing an application. “Manage or add components to your application to enable new features” is correct because the Library is an App Studio capability for incorporating packaged, reusable assets into an application. This allows teams to introduce supported functionality without rebuilding an equivalent component from scratch.

“Use the library to optimize your application creation by reusing assets from all the existing applications developed in your Pega ecosystem” is also correct in the sense that a governed Library enables reuse of assets that have been made available across applications in the Pega environment. Reuse reduces duplicate implementation effort and promotes consistent experiences and implementation patterns. An LSA should use appropriate governance when publishing and evolving shared assets so that consuming applications can benefit from stable, maintainable components. Pega’s low-code application-development approach emphasizes composability and reuse as ways to accelerate delivery while maintaining consistency. See the Pega Academy application development topics and the Pega Documentation portal for current App Studio and reusable-component guidance.