ACD300 Q&As

Appian Certified Lead Developer

Pass Appian ACD300 Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.pass4itsure.com/acd300.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by Appian Official Exam Center

QUESTION 1

You have 5 applications on your Appian platform in production. Users are now beginning to use multiple applications across the platform, and the client wants to ensure a consistent user experience across all applications You notice that some applications use rich text some use section layouts, and others use box layouts. The result is that each application has a different color and size for the header.

What would you recommend to ensure consistency across the platform?

A. Create constants for text size and color, and update each section lo reference these values.

B. In the common application, create a rule that can be used across the platform for section headers, and update each application to reference this new rule

C. In the common application, create one rule for each application, and update each application to reference its respective rule

D. In each individual application, create a rule that can be used lot tot section headers, and update each application lo reference its respective rule

Correct Answer: B

The best way to ensure consistency across the platform is to create a rule that can be used across the platform for section headers. This rule can be created in the common application, and then each application can be updated to reference

this rule. This will ensure that all of the applications use the same color and size for the header, which will provide a consistent user experience.

The other options are not as effective. Option A, creating constants for text size and color, and updating each section to reference these values, would require updating each section in each application. This would be a lot of work, and it would

be easy to make mistakes. Option C, creating one rule for each application, would also require updating each application. This would be less work than option A, but it would still be a lot of work, and it would be easy to make mistakes. Option

D, creating a rule in each individual application, would not ensure consistency across the platform. Each application would have its own rule, and the rules could be different. This would not provide a consistent user experience.

Best Practices:

When designing a platform, it is important to consider the user experience. A consistent user experience will make it easier for users to learn and use the platform.

When creating rules, it is important to use them consistently across the platform. This will ensure that the platform has a consistent look and feel. When updating the platform, it is important to test the changes to ensure that they do not break

the user experience.

QUESTION 2

HOTSPOT

You are reviewing log files that can be accessed in Appian to monitor and troubleshoot platform-based issues.

For each type of log file, match the corresponding Information that it provides. Each description will either be used once, or not at all.

Note: To change your responses, you may deselect your response by clicking the blank space at the top of the selection list.

Hot Area:


design_errors.csv
Select a match:
Inbound requests using HTTP basic authentication
Number of enabled environments
Errors in start forms, task forms, record lists
Metrics such as the number of service accounts
Metrics such as the total time spent evaluating a plug-in function
devops_infrastructure.csv
Select a match:
Inbound requests using HTTP basic authentication
Number of enabled environments
Errors in start forms, task forms, record lists
Metrics such as the number of service accounts
Metrics such as the total time spent evaluating a plug-in function
login-audit.csv
Select a match:
Inbound requests using HTTP basic authentication
Number of enabled environments
Errors in start forms, task forms, record lists
Metrics such as the number of service accounts
Metrics such as the total time spent evaluating a plug-in function

Correct Answer:


design_errors.csv
Select a match:
Inbound requests using HTTP basic authentication
Number of enabled environments
Errors in start forms, task forms, record lists
Metrics such as the number of service accounts
Metrics such as the total time spent evaluating a plug-in function
devops_infrastructure.csv
Select a match:
Inbound requests using HTTP basic authentication
Number of enabled environments
Errors in start forms, task forms, record lists
Metrics such as the number of service accounts
Metrics such as the total time spent evaluating a plug-in function
login-audit.csv
Select a match:
Inbound requests using HTTP basic authentication
Number of enabled environments
Errors in start forms, task forms, record lists
Metrics such as the number of service accounts
Metrics such as the total time spent evaluating a plug-in function

QUESTION 3

On the latest Health Check report from your Cloud TEST environment utilizing a ManaDB add-on. you note the following findings

Category; User Experience Description; # of slow query rules Risk; High

Category; User Experience

Description: U of slow write to data store nodes

Risk: High

Which three things might you do to address this, without consulting the business?

A. Reduce the batch size for database queues to 10.

B. Optimize the database execution use standard database performance troubleshooting methods and tools (such as query execution plans)

C. Reduce the size and complexity of the inputs. If you ore passing in a list, consider whether (he data model can be redesigned lo pass single values instead

D. Optimize the database execution. Replace the new with a materialized view.

E. Use smaller CDTs or limit the fields selected in alqueryEntity()

Correct Answer: BCE

The three things that might help to address the findings of the Health Check report are:

B. Optimize the database execution using standard database performance troubleshooting methods and tools (such as query execution plans). This can help to identify and eliminate any bottlenecks or inefficiencies in the database queries that are causing slow query rules or slow write to data store nodes. C. Reduce the size and complexity of the inputs. If you are passing in a list, consider whether the data model can be redesigned to pass single values instead. This can help to reduce the amount of data that needs to be transferred or processed by the database, which can improve the performance and speed of the queries or writes.

E. Use smaller CDTs or limit the fields selected in a!queryEntity(). This can help to reduce the amount of data that is returned by the queries, which can improve the performance and speed of the rules that use them. The other options are incorrect for the following reasons:

A. Reduce the batch size for database queues to 10. This might not help to address the findings, as reducing the batch size could increase the number of transactions and overhead for the database, which could worsen the performance and speed of the queries or writes.

D. Optimize the database execution. Replace the new with a materialized view. This might not help to address the findings, as replacing a view with a materialized view could increase the storage space and maintenance cost for the database, which could affect the performance and speed of the queries or writes. Verified References: Appian Documentation, section "Performance Tuning".

QUESTION 4

You need to design a complex Appian integration to call a RESTful API. The RESTful API will be used to update a case in a customer\\'s legacy system.

What are three prerequisites for designing the integration?

A. Define the HTTP method that the integration will use.

B. Understand the content of the expected body. Deluding each field type and their limits

C. Understand whether this integration will be used in an interface or in a process model

D. Understand the different error codes managed by the API and the process of error handing m Appall

E. Understand the business rules to be applied to ensure the business logic of the data

Correct Answer: ABD

To design a complex Appian integration to call a RESTful API, you need to have some prerequisites, such as: Define the HTTP method that the integration will use. The HTTP method is the action that the integration will perform on the API, such as GET, POST, PUT, PATCH, or DELETE. The HTTP method determines how the data will be sent and received by the API, and what kind of response will be expected. Understand the content of the expected body, including each field type and their limits. The body is the data that the integration will send to the API, or receive from the API, depending on the HTTP method. The body can be in different formats, such as JSON, XML, or form data. You need to understand how to structure the body according to the API specification, and what kind of data types and values are allowed for each field. Understand the different error codes managed by the API and the process of error handling in Appian. The error codes are the status codes that indicate whether the API request was successful or not, and what kind of problem occurred if not. The error codes can range from 200 (OK) to 500 (Internal Server Error), and each code has a different meaning and implication. You need to understand how to handle different error codes in Appian, and how to display meaningful messages to the user or log them for debugging purposes. The other two options are not prerequisites for designing the integration, but rather considerations for implementing it. Understand whether this integration will be used in an interface or in a process model. This is not a prerequisite, but rather a decision that you need to make based on your application requirements and design. You can use an integration either in an interface or in a process model, depending on where you need to call the API and how you want to handle the response. For example, if you need to update a case in real-time based on user input, you may want to use an integration in an interface. If you need to update a case periodically based on a schedule or an event, you may want to use an integration in a process model. Understand the business rules to be applied to ensure the business logic of the data. This is not a prerequisite, but rather a part of your application logic that you need to implement after designing the integration. You need to apply business rules to validate, transform, or enrich the data that you send or receive from the API, according to your business requirements and logic. For example, you may need to check if the case status is valid before updating it in the legacy system, or you may need to add some additional information to the case data before displaying it in Appian.

QUESTION 5

You are the lead developer for an Appian project, in a backlog refinement meeting You are presented with the following user story.

As a restaurant customer. I need to be able to place my food order online to avoid waiting in line for take out.\\'

Which two functional acceptance criteria would you consider \\'good\\'?

A. The user will click Save, and the order information will be saved in the ORDER table and have audit history

B. The user will receive an email notification when their order is completed.

C. The system mutt handle up to 500 unique orders per day

D. The user cannot submit the form without filling out all required fields.

Correct Answer: BD

Functional acceptance criteria are the conditions that a user story must satisfy to be accepted by the user or stakeholder. They should be specific, measurable, achievable, relevant, and testable. In this case, two functional acceptance criteria that would be considered `good\\' are: The user will receive an email notification when their order is completed. This is a specific, measurable, achievable, relevant, and testable criterion that describes a feature that the user needs to be informed of their order status. The user cannot submit the form without filling out all required fields. This is a specific, measurable, achievable, relevant, and testable criterion that describes afeature that the user needs to provide valid and complete information for their order. The other options are not as good. Option A, the user will click Save, and the order information will be saved in the ORDER table and have audit history, is not a functional acceptance criterion, but rather a technical implementation detail that is not relevant or visible to the user. Option C, the system must handle up to 500 unique orders per day, is not a functional acceptance criterion, but rather a non-functional requirement that describes a performance or quality attribute of the system.