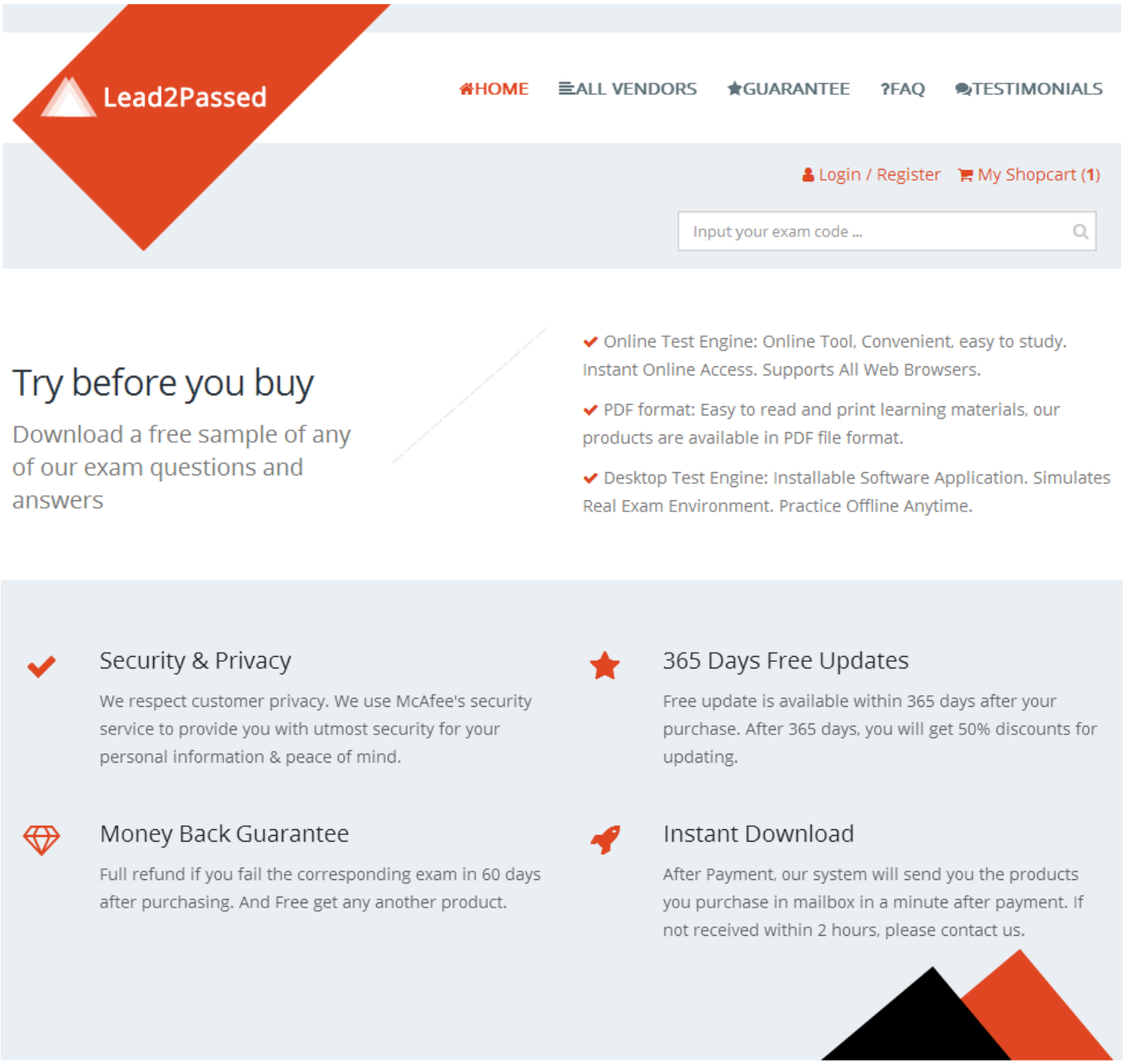
Lead2Passed



http://www.lead2passed.com

Valid Certification Exam Dumps Materials and Study Guide - Lead2Passed

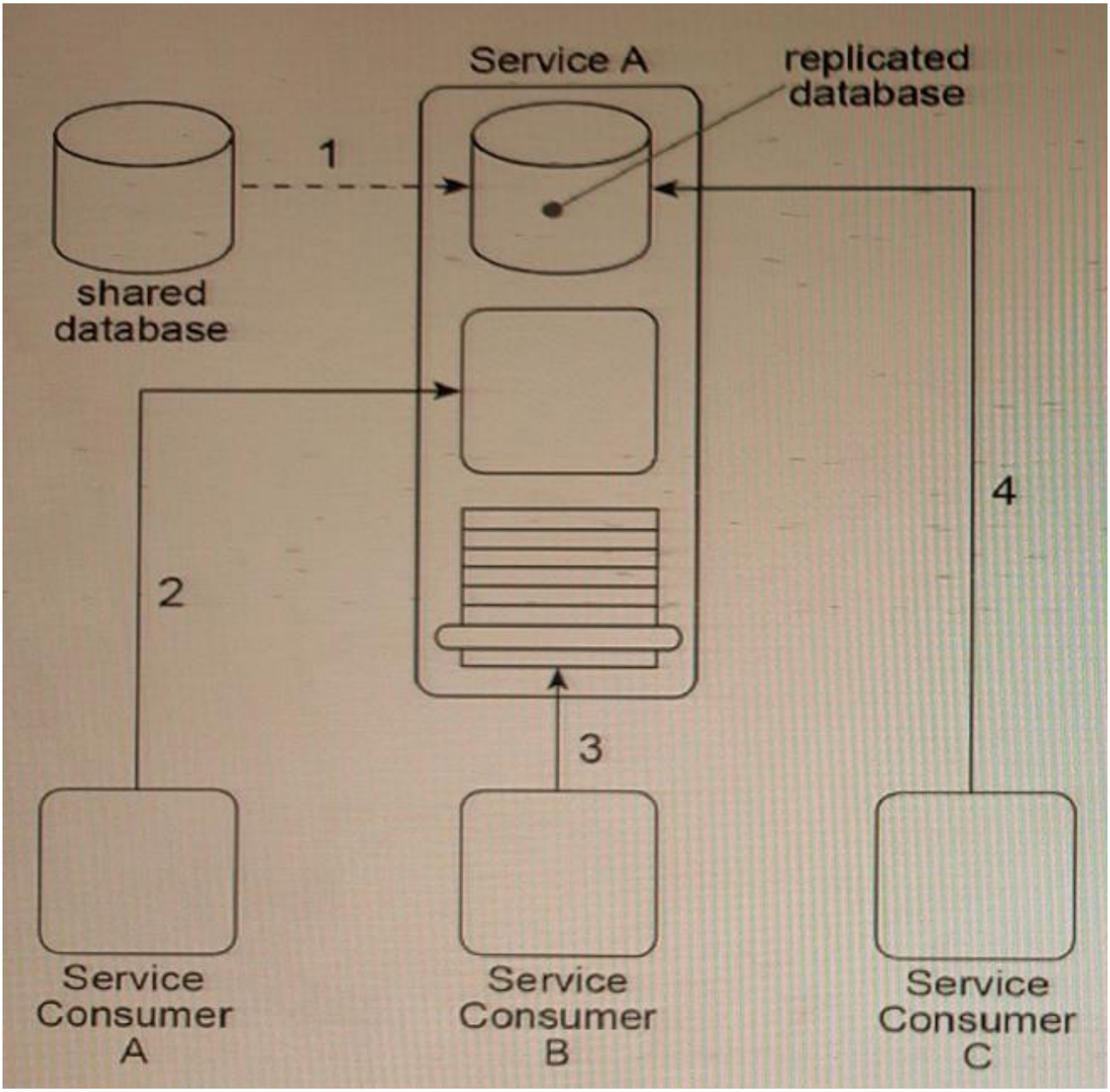
Exam : **S90.08B**

Title : SOA Design & Architecture Lab with Services & Microservices

Vendor : SOA

Version : DEMO

NO.1 Refer to Exhibit.



Service A is a utility service that provides generic data access logic to a database containing data that is periodically replicated from a shared database (1). Because the Standardized Service Contract principle was applied to the design of Service A, its service contract has been fully standardized.
The service architecture of Service A Is being accessed by three service consumers. Service Consumer A accesses a component that is part of the Service A Implementation by Invoking it directly (2). Service Consumer B invokes Service A by accessing Its service contract (3). Service Consumer C directly accesses the replicated database that Is part of the Service A Implementation (4).
You've been told that the reason Service Consumers A and C bypass the published Service A service contract is because, for security reasons, they are not allowed to access a subset of the capabilities in the API that comprises the Service A service contract. How can the Service A architecture be changed to enforce these security restrictions while avoiding negative forms of coupling?

A. The Contract Centralization pattern can be applied to force service consumers to access the Service A architecture via its published service contract only. The Service Loose Coupling principle can

then be applied to ensure that the centralized service contract does not contain any content that is dependent on or derived from the underlying service implementation.

B. The Contract Centralization pattern can be applied to force all service consumers to access the Service A architecture via its published service contract. This will prevent negative forms of coupling that could lead to problems when the database is replaced. The Service Abstraction principle can then be applied to hide underlying service architecture details so that future service consumers cannot be designed to access any part of the underlying service implementation.

C. The Contract Centralization pattern can be applied to force service consumers to access the Service A architecture via its published service contract only. The Idempotent Capability pattern can be applied to Service A to establish alternative sets of service capabilities for service consumers with different levels of authorization.

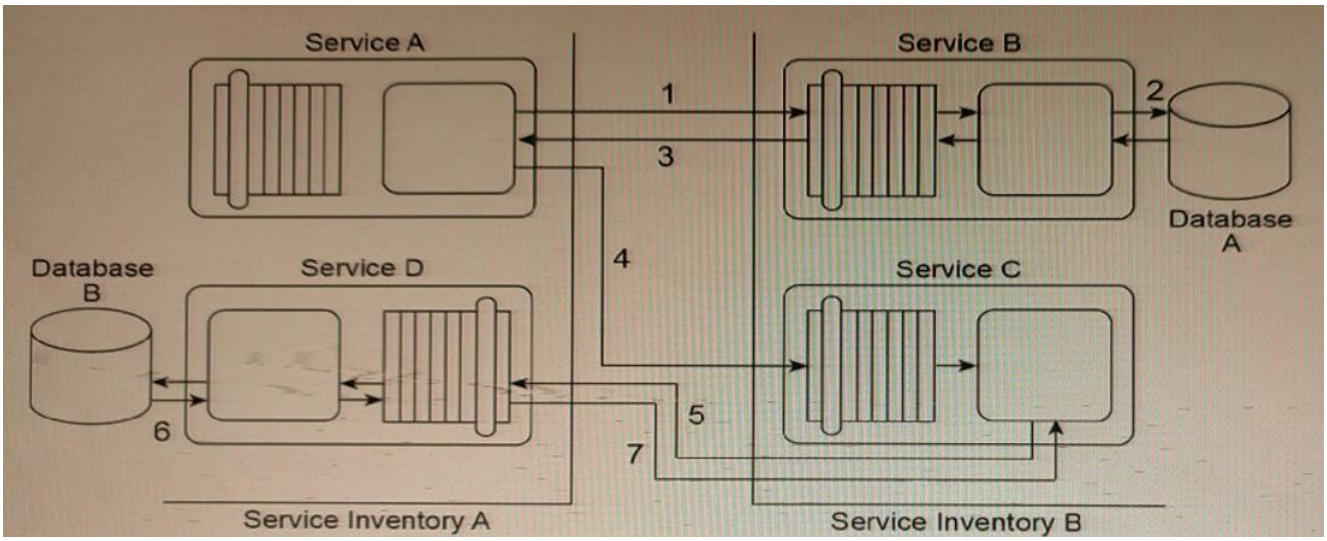
D. The Contract Centralization pattern can be applied to force service consumers to access the Service A architecture via its published service contract only. The Concurrent Contracts pattern can be applied to Service A in order to establish one or more alternative service contracts. This allows service consumers with different levels of authorization to access different types of service logic via Service A's published service contracts.

Answer: D

Explanation:

The Contract Centralization pattern can be applied to force service consumers to access the Service A architecture via its published service contract only. The Service Loose Coupling principle can then be applied to ensure that the centralized service contract does not contain any content that is dependent on or derived from the underlying service implementation. This will enforce the security restrictions while avoiding negative forms of coupling. By ensuring loose coupling, changes to the implementation of Service A will not require changes to its published service contract, making it easier to maintain and evolve the service.

NO.2 Refer to Exhibit.



Service A sends a message to Service B (1). After Service B writes the message contents to Database A (2), it issues a response message back to Service A (3). Service A then sends a message to Service C (4). Upon receiving this message, Service C sends a message to Service D (5), which then writes the message contents to Database B (6) and issues a response message back to Service C (7).

Service A and Service D are located in Service Inventory A. Service B and Service C are located in

Service Inventory B.
You are told that In this service composition architecture, all four services are exchanging invoice-related data in an XML format. However, the services in Service Inventory A are standardized to use a different XML schema for invoice data than the services in Service Inventory B. Also, Database A can only accept data in the Comma Separated Value (CSV) format and therefore cannot accept XML-formatted data. Database B only accepts XML-formatted data. However, it is a legacy database that uses a proprietary XML schema to represent invoice data that is different from the XML schema used by services in Service Inventory A or Service Inventory B.
What steps can be taken to enable the planned data exchange between these four services?

A. The Protocol Bridging pattern can be applied so that protocol conversion logic is positioned between the Service B logic and Database A. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between Service A and Service B, between Service A and Service C, between Service C and Service D, and between the Service D logic and Database

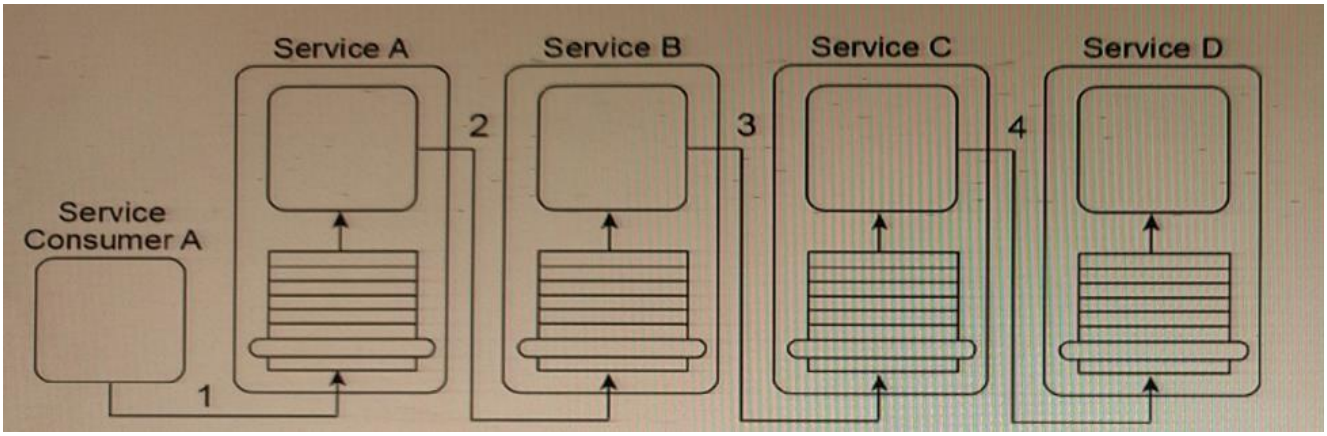
B. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service B, between Service C and Service D, and between the Service D logic and Database B. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between Service A and Service C, and between the Service B logic and Database

C. The Data Model Transformation pattern can be applied so that data model transformation logic is positioned between Service A and Service B, between Service A and Service C, between Service C and Service D, and between the Service D logic and Database B. The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between the Service B logic and Database A.

D. The Protocol Bridging pattern can be applied so that protocol conversion logic is positioned between Service A and Service B, between Service A and Service C, and between Service C and Service The Data Format Transformation pattern can be applied so that data format transformation logic is positioned between the Service B logic and Database A and between the Service D logic and Database B.

Answer: C
Explanation:
This solution addresses the two main challenges in the service composition architecture: the different XML schema used by services in Service Inventory A and Service Inventory B, and the incompatible data formats of the two databases.
By applying the Data Model Transformation pattern, data model transformation logic can be inserted to map the invoice-related data between the different XML schemas used by the services in Service Inventory A and Service Inventory B. This can be done at the appropriate points in the message flow: between Service A and Service B, between Service A and Service C, between Service C and Service D, and between the Service D logic and Database B.
By applying the Data Format Transformation pattern, data format transformation logic can be inserted to convert the XML-formatted data used by the services to the CSV format required by Database A, and to convert the proprietary XML schema used by Database B to the XML schema used by the services. This can be done between the Service B logic and Database A.
The Protocol Bridging pattern is not necessary in this case because all services are already communicating using the same protocol (presumably HTTP or a similar protocol).

NO.3 Refer to Exhibit.



Service Consumer A sends a message to Service A (1), which then forwards the message to Service B (2). Service B forwards the message to Service C (3), which finally forwards the message to Service D (4). However, Services A, B and C each contain logic that reads the contents of the message to determine what intermediate processing to perform and which service to forward the message to. As a result, what is shown in the diagram is only one of several possible runtime scenarios.
Currently, this service composition architecture is performing adequately, despite the number of services that can be involved in the transmission of one message. However, you are told that new logic is being added to Service A that will require it to compose one other service to retrieve new data at runtime that Service A will need access to in order to determine where to forward the message to. The involvement of the additional service will make the service composition too large and slow.
What steps can be taken to improve the service composition architecture while still accommodating the new requirements and avoiding an increase in the amount of service composition members?

A. The Intermediate Routing pattern can be applied together with the Service Agent pattern by removing Service B or Service C from the service composition and replacing it with a service agent capable of intercepting and forwarding the message at runtime based on pre-defined routing logic. The Service Discoverability principle can be further applied to ensure that Service A can be found by any future service consumers.

B. The Asynchronous Queuing pattern can be applied together with a Routing service that is invoked by messages read from a messaging queue. This new Routing service can replace Service B and can be accessed by Service A and Service C so they can determine where to forward messages to at runtime. The Service Loose Coupling principle can be further applied to ensure that the new Routing service remains decoupled from other services so that it can perform its routing functions independently from service contract invocation.

C. The Service Instance Routing pattern can be applied to introduce a Routing service to provide a centralized service to contain routing-related business rules. This new Routing service can be accessed by Service A and Service C so they can determine where to forward messages to at runtime. The Service Reusability principle can be further applied to ensure that the logic in all remaining services is designed to be multi-purpose and reusable.

D. The Intermediate Routing pattern can be applied together with the Service Agent pattern to establish a service agent capable of intercepting and forwarding the message at runtime based on pre-defined routing logic. The Service Composability principle can be further applied to ensure that all services are designed as effective service composition participants.

Answer: D

Explanation:

This solution addresses the issue of the service composition becoming too large and slow by introducing a new Routing service that is invoked by messages read from a messaging queue. This allows Service A and Service C to determine where to forward messages to at runtime without the need for additional services in the composition. The Service Loose Coupling principle is applied to ensure that the new Routing service remains decoupled from other services so that it can perform its routing functions independently from service contract invocation.