

KillTest

높은 품질, 더 나은 서비스



연습 문제

<https://www.killtest.kr>

1년간 무료 업데이트 제공

Exam : AZ-202

**Title : Microsoft Azure Developer
Certification Transition**

Version : DEMO

1. Topic 1, Litware Inc.Overview

Background

You are a developer for Litware Inc., a SaaS company that provides a solution for managing employee expenses. The solution consists of an ASP.NET Core Web API project that is deployed as an Azure Web App.

Overall architecture

Employees upload receipts for the system to process. When processing is complete, the employee receives a summary report email that details the processing results. Employees then use a web application to manage their receipts and perform any additional tasks needed for reimbursement

Receipt processing

Employees may upload receipts in two ways:

- Uploading using an Azure Files mounted folder
- Uploading using the web application

Data Storage

Receipt and employee information is stored in an Azure SQL database.

Documentation

Employees are provided with a getting started document when they first use the solution. The documentation includes details on supported operating systems for Azure File upload, and instructions on how to configure the mounted folder.

Solution details

Users table

Column	Description
UserId	unique identifier for and employee
ExpenseAccount	employees expense account number in the format 1234-123-1234
AllowedAmount	limit of allowed expenses before approval is needed
SupervisorId	unique identifier for employee's supervisor
SecurityPin	value used to validate user identity

Web Application

You enable MSI for the Web App and configure the Web App to use the security principal name,

Processing

Processing is performed by an Azure Function that uses version 2 of the Azure Function runtime.

Once processing is completed, results are stored in Azure Blob. Storage and an Azure SQL database.

Then, an email summary is sent to the user with a link to the processing report. The link to the report must remain valid if the email is forwarded to another user.

Requirements

Receipt processing

Concurrent processing of a receipt must be prevented.

Logging

Azure Application Insights is used for telemetry and logging in both the processor and the web application. The processor also has Trace Writer logging enabled. Application Insights must always contain all log messages.

Disaster recovery

Regional outage must not impact application availability. All DR operations must not be dependent on application running and must ensure that data in the DR region is up to date.

Security

Users' SecurityPin must be stored in such a way that access to the database does not allow the viewing of SecurityPins. The web application is the only system that should have access to SecurityPins.

All certificates and secrets used to secure data must be stored in Azure Key Vault. You must adhere to the Least Privilege Principal. All access to Azure Storage and Azure SQL database must use the application's Managed Service Identity (MSI).

Receipt data must always be encrypted at rest.

All data must be protected in transit, User's expense account number must be visible only to logged in users. All other views of the

expense account number should include only the last segment, with the remaining parts obscured. In the case of a security breach, access to all summary reports must be revoked without impacting other parts of the system.

Issues

Upload format issue

Employees occasionally report an issue with uploading a receipt using the web application. They report that when they upload a receipt using the Azure File Share, the receipt does not appear in their profile. When this occurs, they delete the file in the file share and use the web application, which returns a 500 Internal Server error page.

Capacity issue

During busy periods, employees report long delays between the time they upload the receipt and when it appears in the web application.

Log capacity issue

Developers report that the number of log messages in the trace output for the processor is too high, resulting in lost log messages-

Application code

Processing.cs

Processing.cs

```

PC01 public static class Processing
PC02 {
PC03     public static class Function
PC04     {
PC05         [FunctionName ("IssueWork")]
PC06         public static async Task Run ([TimerTrigger("0 */5 * * * *")] TimerInfo timer, ILogger log)
PC07         {
PC08             var container = await GetCloudBlobContainer();
PC09             foreach (var fileItem in await ListFiles())
PC10             {
PC11                 var file = new CloudFile (fileItem.StorageUri.PrimaryUri);
PC12                 var ms = new MemoryStream();
PC13                 await file.DownloadToStreamAsync(ms);
PC14                 var blob = container.GetBlockBlobReference (fileItem.Uri.ToString());
PC15                 await blob.UploadFromStreamAsync(ms);
PC16             }
PC17         }
PC18     }
PC19     private static CloudBlockBlob GetDRBlob (CloudBlockBlob sourceBlob)
PC20     {
PC21         . . .
PC22     }
PC23     private static async Task<CloudBlobContainer> GetCloudBlobContainer()
PC24     {
PC25         var cloudBlobClient = new CloudBlobClient (new Uri(" . . ."), await GetCredentials());
PC26
PC27         await cloudBlobClient.GetRootContainerReference().CreatIfNotExistAsync();
PC28         return cloudBlobClient.GetRootContainerReference();
PC29     }
PC30     private static async Task<StorageCredentials> GetCredentials()
PC31     {
PC32         . . .
PC33     }
PC34     private static async Task<List<IListFileItem>> ListFiles()
PC35     {
PC36         . . .
PC37     }
PC37     private KeyVaultClient _keyVaultClient = new KeyVaultClient(" . . .");
PC38 }
PC39 }

```

Database.cs

```
DB01 public class Database
DB02 {
DB03     private string ConnectionString =
DB04
DB05     public async Task<object> LoadUserDetails(string userId)
DB06     {
DB07
DB08     return await policy.ExecuteAsync (async () =>
DB09     {
DB10         using (var connection = new SqlConnection (ConnectionString))
DB11         {
DB12             await connection.OpenAsync();
DB13             using (var command = new SqlCommand("_", connection))
DB14             using (var reader = command.ExecuteReader())
DB15             {
DB16                 -
DB17             }
DB18         }
DB19     } };
DB20 }
DB21 }
```

ReceiptUploader.cs

```
RU01 public class ReceiptUploader
RU02 {
RU03     public async Task UploadFile(string file, byte[] binary)
RU04     {
RU05         var httpClient = new HttpClient();
RU06         var response = await httpClient.PutAsync( "...", new ByteArrayContent(binary));
RU07         while (ShouldRetry (response))
RU08         {
RU09             response = await httpClient.PutAsync ( "...", new ByteArrayContent(binary));
RU10         }
RU11     }
RU12     private bool ShouldRetry(HttpResponseMessage response)
RU13     {
RU14
RU15     }
RU16 }
```

ConfigureSSE.ps1

```

CS01 $storageAccount = Get-AzureRmStorageAccount -ResourceGroupName "$_" -AccountName "$_"
CS02 $keyVault = Get-AzureRmKeyVault -VaultName "$_"
CS03 $key = Get-AzureKeyVaultKey -VaultName $keyVault.VaultName -Name "$_"
CS04 Set-AzureRmKeyVaultAccessPolicy'
CS05 -VaultName $keyVault.VaultName'
CS06 -ObjectId $storageAccount.Identity.PrincipalId'
CS07
CS08
CS09 Set-AzureRmStorageAccount"
CS10 -ResourceGroupName $storageAccount.ResourceGroupName'
CS11 -AccountName $storageAccount.StorageAccountName'
CS12 -EnableEncryptionService File '
CS13 -KeyvaultEncryption'
CS14 -KeyName $key.Name
CS15 -KeyVersion $key.Version'
CS16 -KeyVaultUri $keyVault.VaultUri

```

DRAG DROP

You need to ensure that the upload format issue is resolved.

What code should you add at line RU14? To answer, drag the appropriate code fragments to the correct locations. Each code fragment may

be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content. NOTE: Each correct selection is worth one point.

Values

- SMBDeletePending
- ShareBeingDeleted
- HttpStatusCode.Conflict
- CannotDeleteFileOrDirectory
- HttpStatusCode.InternalServerError

Answer Area

```

return
response.StatusCode = = 
&&
response.ReasonPhrase = = "

```

Answer:**Values**

- SMBDeletePending
- ShareBeingDeleted
- HttpStatusCode.Conflict
- CannotDeleteFileOrDirectory
- HttpStatusCode.InternalServerError

Answer Area

```

return
response.StatusCode = = HttpStatusCode.InternalServerError
&&
response.ReasonPhrase = = " CannotDeleteFileOrDirectory "

```

Explanation:

Box 1: HttpStatusCode.InternalServerError HttpStatusCode.InternalServerError is equivalent to HTTP status 500. InternalServerError indicates that a generic error has occurred on the server.

Box 2: CannotDeleteFileOrDirectory HttpResponseMessage.ReasonPhrase Property gets or sets the

reason phrase which typically is sent by servers together with the status code.

Scenario: Upload format issue Employees occasionally report an issue with uploading a receipt using the web application. They report that when they upload a receipt using the Azure File Share, the receipt does not appear in their profile. When this occurs, they delete the file in the file share and use the web application, which returns a 500 Internal Server error page.

References: <https://docs.microsoft.com/en-us/dotnet/api/system.net.httpstatuscode?redirectedfrom=MSDN&view=netframework-4.7.2>

2.HOTSPOT

You need to add the Supporting Operating Systems section to the Getting Started document.

How should you complete the section? To answer, select the appropriate options in the answer area.

NOTE: Each correct selection is worth one point.

Operating system	Supported
Windows 7	No Yes
Windows 8.1	No Yes
Windows 10	No Yes

Answer:

Operating system	Supported
Windows 7	No Yes
Windows 8.1	No Yes
Windows 10	No Yes

Explanation:

Scenario: Employees are provided with a getting started document when they first use the solution. The documentation includes details on supported operating systems for Azure File upload, and instructions on how to configure the mounted folder.

You can use Azure file shares on a Windows installation that is running either in an Azure VM or on-premises. The following table illustrates which OS versions support accessing file shares in which environment:

References: <https://docs.microsoft.com/en-us/azure/storage/files/storage-how-to-use-files-windows>

3.HOTSPOT

You need to ensure that security requirements are met.

What value should be used for the ConnectionString field on line DB03 in the Database class? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.

`"Data Source=datastore.database.windows.net;Initial Catalog=expense;`

▼	;
Integrated Security = SSPI	
Trusted_Connection = False	
Network Library = DBNSSOCN	
MultipleActiveResultSets = True	
▼	;"
Encrypt = True	
Integrated Security = True	
Failover Partner = False	
Named Pipes = True	

Answer:

`"Data Source=datastore.database.windows.net;Initial Catalog=expense;`

▼	;
Integrated Security = SSPI	
Trusted_Connection = False	
Network Library = DBNSSOCN	
MultipleActiveResultSets = True	
▼	;"
Encrypt = True	
Integrated Security = True	
Failover Partner = False	
Named Pipes = True	

Explanation:

Box 1: Integrated Security=SSPI

Integrated security: For all data source types, connect using the current user account. For SqlConnection you can use Integrated Security=true; or Integrated Security=SSPI; Scenario: All access to Azure Storage and Azure SQL database must use the application's Managed Service Identity (MSI)

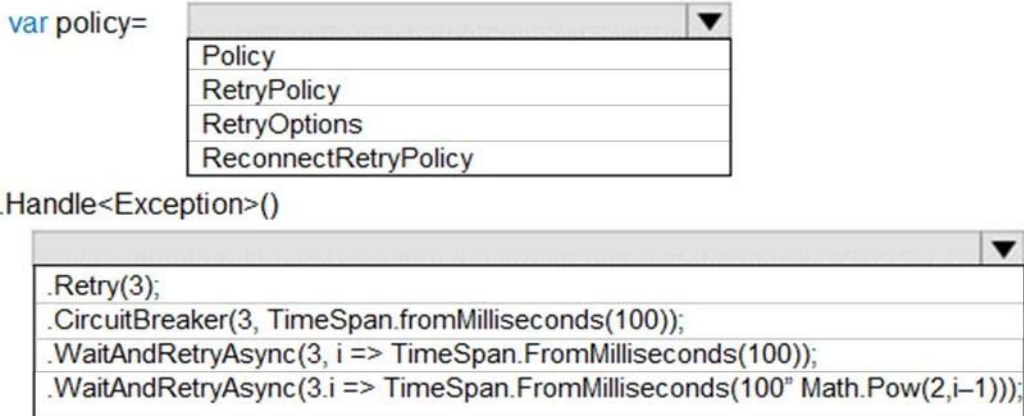
Box 2: Encrypt = True Scenario: All data must be protected in transit. References:

<https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/connection-string-syntax>

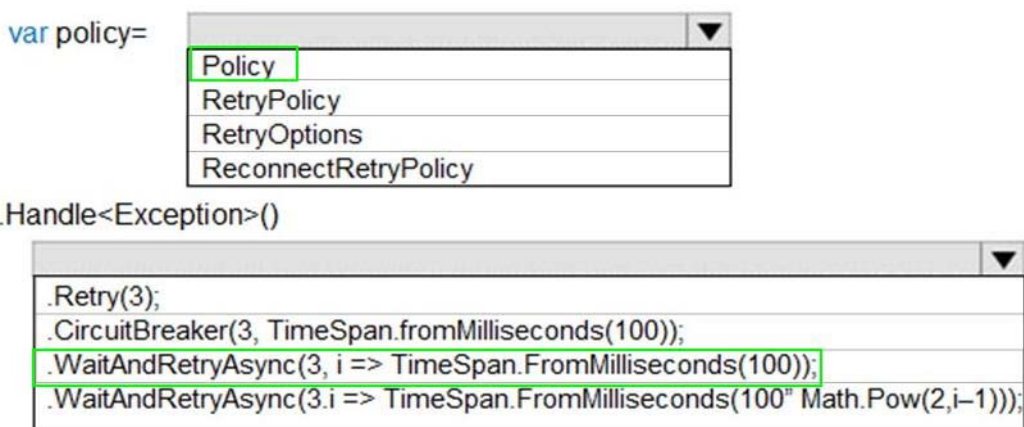
4.HOTSPOT

You need to configure retries in the LoadUserDetails function in the Database class without impacting user experience.

What code should you insert on line DB07? To answer, select the appropriate options in the answer area. NOTE: Each correct selection is worth one point.



Answer:



Explanation:

Box 1: Policy

RetryPolicy retry = Policy.Handle<HttpRequestException>().Retry(3);

The above example will create a retry policy which will retry up to three times if an action fails with an exception handled by the Policy.

Box 2: WaitAndRetryAsync(3, i => TimeSpan.FromMilliseconds(100 * Math.Pow(2, i-1))); A common retry strategy is exponential backoff: this allows for retries to be made initially quickly, but then at progressively longer intervals, to avoid hitting a subsystem with repeated frequent calls if the subsystem may be struggling.

Example: Policy.Handle<SomeExceptionType>()

WaitAndRetry(3, retryAttempt => TimeSpan.FromSeconds(Math.Pow(2, retryAttempt)));

References: <https://github.com/App-vNext/Polly/wiki/Retry>

5.Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution Determine whether the solution meets the stated goals.

You need to ensure that the SecurityPin security requirements are met.

Solution: Enable Always Encrypted for the SecurityPin column using a certificate based on a trusted certificate authority. Update the Getting Started document with instruction to ensure that the certificate is installed on user machines.

Does the solution meet the goal?

A. Yes

B. No

Answer: B