TESTPASSPORT

練 習 問 題



高 品 質 な 学 習 リ ソ ー ス

https://www.testpassport.jp

1年間無料アップデート

Exam : **MS-600**

Title : Building Applications and Solutions with Microsoft 365 Core Services

Version : DEMO

1. Topic 1, ADatum Corporation

This is a case study. **Case studies are not timed separately. You can use as much exam time as you would like to complete each case.** However, there may be additional case studies and sections on this exam. You must manage your time to ensure that you are able to complete all questions included on this exam in the time provided.

To answer the questions included in a case study, you will need to reference information that is provided in the case study. Case studies might contain exhibits and other resources that provide more information about the scenario that is described in the case study. Each question is independent of the other questions in this case study.

At the end on this case study, a review screen will appear. This screen allows you to review your answers and to make changes before you move to the next section of the exam. After you begin a new section, you cannot return to this section.

To start the case study
To display the first question in this case study, click the **Next** button. Use the buttons in the left pane to explore the content of the case study before you answer the questions. Clicking these buttons displays information such as business requirements, existing environment, and problem statements. If the case study has an **All Information** tab, note that the information displayed is identical to the information displayed on the subsequent tabs. When you are ready to answer a question, click the **Question** button to return to the question.

Overview
ADatum Corporation develops a software as a service (SaaS) application named E-invoicing.

Existing Environment
Application Architecture
E-invoicing consists of a single-page application (SPA) and a backend web service that provides invoice management and processing functionality.

E-invoicing stores all the details of each invoicing operation in a backend cloud database. E-invoicing generates invoices in PDF format and provides users with the ability to download the PDF after it is generated. Each invoice has a unique identifier named invoiceid.

The users have a common workflow where they sign in to E-invoicing, and then open E-invoicing in multiple tabs of a web browser so they can use different parts of the application simultaneously.

Security Architecture
ADatum uses the principle of least privilege whenever possible. ADatum always uses the latest libraries and integration endpoints.

Requirements

Business Goals
ADatum wants to integrate E-invoicing, Azure Active Directory (Azure AD), and Microsoft Graph so that their customers can leverage Microsoft Office 365 services directly from within E-invoicing.

Planned Changes
ADatum plans to add the following capabilities to E-invoicing:
- Email the generated invoices to customers on behalf of the current signed-in user. Any emails generated by the system will contain the invoiced.
- Perform as many operations as possible in the browser without having to leave the E-invoicing application.
- Use Azure AD to manage identities, authentication, and authorization.
- Display all emails that contain a specific invoiceid.

Technical Requirements
ADatum identifies the following technical requirements for the planned E-invoicing capabilities:
- Ensure that all operations performed by E-invoicing against Office 365 are initiated by a user. Require that the user authorize E-invoicing to access the Office 365 data the first time the application attempts to access Office 365 data on the user's behalf.
- Send scheduled reminders to customers before a payment due date. Create an administration user interface to enable the scheduled reminders.
- Implement Microsoft Graph change notifications to detect emails from vendors that arrive in a designated mailbox.
- Implement single sign-on (SSO) and minimize login prompts across browser tabs.
- Secure access to the backend web service by using Azure AD.
- Ensure that all solutions use secure coding practices.

Backend Security Planned Changes
ADatum wants to use custom application roles to map user functionality to permissions granted to users. E-invoicing will have internal logic that will dynamically identify whether the user should be allowed to call the backend API.

SSO JavaScript Script
You plan to implement SSO with Microsoft Authentication Library (MSAL) by using the following code:

```
01 const config ={
02   auth: {
03      clientId: '3b41e6eb-29a1-44cc-8473-e8abfe5c4e07'
04   },
05   cache: {
06    ...
07   }
08 }
09 const myMSALObj = new UserAgentApplication(config);
10 ...
```

Access Token JavaScript Script

You have the following JavaScript code to obtain an access token.

```
01 ...
02 userAgentApplication.acquireTokenSilent(accessTokenRequest).then(
03  function(accessTokenResponse) {
04    let accessToken = accessTokenResponse.accessToken;
05  }).catch(function (error) {
06    if (error.errorMessage.indexOf('interaction_required') !== -1) {
07      userAgentApplication.acquireTokenPopup(accessTokenRequest).then(
08          function(accessTokenResponse) {
09           ...
10          }).catch(function(error) {
11            console.log(error);
12          });
13    }
14    console.log(error);
15 });
```

Change Notification JSON

You have the following JSON message that will be sent by the Microsoft Graph service to detect the vendor emails.

```
01 {
02    "value": [
03      {
04          "subscriptionId":"<subscription_guid>"
05          "subscriptionExpirationDateTime":"<expirationdate>",
06          "clientState":"<secret>",
07          "changeType":"created",
08          "resource":
09            "users/<user_guid>@<tenant_guid>/messages/<long_id_string>",
10          "resourceData":
11     {
12      "@odata.type":"#Microsoft.Graph.Message",
13      "@odata.id":"Users/<user_guid>@<tenant_guid>/Messages/<long_id_string>",
14      "@odata.etag":"W/\"CQAAABYAAADkrWGo7bouTKlsgTZMr9KwAAAUWRHf\"",
15      "id":<long_id_string>"
16     }
17    }
18  ]
19 }
```

Which type of authentication flow should you recommend for the planned integration with Office 365?

A. device code

B. implicit grant

C. authorization code

D. client credentials

Answer: C
Explanation:
To use Microsoft Graph to read and write resources on behalf of a user, your app must get an access token from the Microsoft identity platform and attach the token to requests that it sends to Microsoft Graph.
One common flow used by native and mobile apps and also by some Web apps is the OAuth 2.0 authorization code grant flow.
Scenario: Email the generated invoices to customers on behalf of the current signed-in user. Any emails generated by the system will contain the invoiced. Use Azure AD to manage identities, authentication, and authorization.
Reference: https://docs.microsoft.com/en-us/graph/auth-v2-user

2.Which URI should you use to query all the email that relate to an invoice?
A. https://graph.microsoft.com/v1.0/me/messages?$filter=contains(subject, {invoiceid})
B. https://graph.microsoft.com/v1.0/me/messages?$subject eq {invoiceid}
C. https://graph.microsoft.com/v1.0/me/messages?$search="{invoiceid}"
D. https://graph.microsoft.com/v1.0/me/messages?${invoiceid}
Answer: C
Explanation:
Reference: https://docs.microsoft.com/en-us/graph/search-query-parameter

3.You need to implement the role functionality for the backend web service calls.
Which two actions should you perform? Each correct answer presents part of the solution. NOTE: Each correct selection is worth one point.
A. Upload a certificate for the application registration of the backend web service.
B. Modify the manifest that defines the application roles and set Allowed Member Types to Apps.
C. Modify the manifest that defines the application roles and set Allowed Member Types to Users.
D. Assign the application roles to the Azure AD group that contains the users who are mapped to the roles.
E. Create a new client secret in the application registration of the backed web service.
Answer: B,D
Explanation:
Reference: https://docs.microsoft.com/en-us/azure/active-directory/develop/howto-add-app-roles-in-azure-ad-apps

4.How can you validate that the JSON notification message is sent from the Microsoft Graph service?
A. The ClientState must match the value provided when subscribing.
B. The user_guid must map to a user ID in the Azure AD tenant of the customer.
C. The tenant ID must match the tenant ID of the customer’s Office 365 tenant.
D. The subscription ID must match the Azure subscription used by ADatum.
Answer: A
Explanation:
clientState specifies the value of the clientState property sent by the service in each notification. The maximum length is 128 characters. The client can check that the notification came from the service by

comparing the value of the clientState property sent with the subscription with the value of the clientState property received with each notification.
Note: A subscription allows a client app to receive notifications about changes to data in Microsoft Graph.
Reference: https://docs.microsoft.com/en-us/graph/api/resources/subscription

5.You need to complete the MSAL.js code for SSO.
Which code segment should you insert at line 06?
A. storeAuthStateInCookie: false
B. storeAuthStateInCookie: true
C. cacheLocation: 'localStorage'
D. cacheLocation: 'sessionStorage'
Answer: C
Explanation:
Scenario: Implement single sign-on (SSO) and minimize login prompts across browser tabs.
When your application is open in multiple tabs and you first sign in the user on one tab, the user is also signed in on the other tabs without being prompted. MSAL.js caches the ID token for the user in the browser localStorage and will sign the user in to the application on the other open tabs.
By default, MSAL.js uses sessionStorage which does not allow the session to be shared between tabs.
To get SSO between tabs, make sure to set the cacheLocation in MSAL.js to localStorage.
Reference: https://docs.microsoft.com/bs-latn-ba/Azure/active-directory/develop/msal-js-sso