Free Questions for MS-600

Shared by Olsen on 12-12-2023

For More Free Questions and Preparation Resources

Check the Links on Last Page

Question 1

Question Type: MultipleChoice

Case Study: ADatum Corporation

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 invoiced.

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 }
```

You have an Azure Static Web Apps app named Web1.

You build a Microsoft Teams app named App1.

You need to deploy App1 to Web1 by using GitHub Actions.

Which two credentials can you use? Each correct answer presents a complete solution.

NOTE: Each correct selection is worth one point.

Options:

A- a service principal
B- the identity of the GitHub Actions workflow
C- a Static Web Apps deployment token
D- a client secret

Answer:

A, C

Question 2

Question Type: MultipleChoice

Case Study: ADatum Corporation

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 invoiced.

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 }
```

Note: This question is part of a series of questions that present the same scenario. Each question in the series contains a unique solution that might meet the stated goals. Some question sets have more than one correct solution, while others might not have a correct solution.

After you answer a question in this section, you will NOT be able to return to it. As a result, these questions will not appear in the review screen.

You need to deploy a Microsoft Teams app. The app will create a Microsoft Teams personal tab that displays a website.

Solution: You manually create a Microsoft Teams manifest file that includes a configurableTabs section containing the name, ID, and content URL of the website.

Does this meet the goal?

Options:

A- Yes

B- No

Answer:

B

Explanation:

https://docs.microsoft.com/en-us/microsoftteams/platform/tabs/how-to/create-personal-tab?tabs=nodejs

Question 3

Question Type: OrderList

Case Study: ADatum Corporation

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 invoiced.

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.

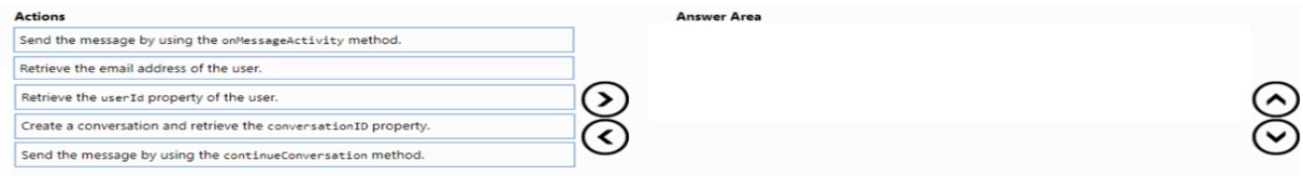
```
01 {
02    "value": [
03      {
04          "subscriptionId":"<subscription_guid>"
05          "subscriptionExpirationDateTime":"<expirationdate>",
06          "clientState":"<secret>",
07          "changeType":"created",
08          "resource":
09            "users/<user_guid>@<tenant_guid>/messages/<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 }
```

You are building a Microsoft Teams bot by using the Microsoft Bot Framework SDK.

You need to configure the bot to send proactive messages to users with whom the bot has NOT interacted.

Which three actions should the bot perform m sequence? To answer, move the appropriate

actions from the list of actions to the answer area and arrange them in the correct order.



Answer:

Retrieve the userId property of the user.Create a conversion and retrieve the conversationID property.Send the message by using the continueConversation method.

Question 4

Question Type: MultipleChoice

Case Study: ADatum Corporation

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 invoiced.

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.

```
{
  "value": [
    {
      "subscriptionId":"<subscription_guid>"
      "subscriptionExpirationDateTime":"<expirationdate>",
      "clientState":"<secret>",
      "changeType":"created",
      "resource":
        "users/<user_guid>@<tenant_guid>/messages/<long_id_string>",
      "resourceData":
   {
    "@odata.type":"#Microsoft.Graph.Message",
    "@odata.id":"Users/<user_guid>@<tenant_guid>/Messages/<long_id_string>",
    "@odata.etag":"W/\"CQAAABYAAADkrWGo7bouTKlsgTZMr9KwAAAUWRHf\"",
    "id":<long_id_string>"
   }
  }
 ]
}
```

You are building a custom API.

Client applications will use access tokens to authenticate to the API.

You need to validate the integrity of the tokens.

Which three elements should you verity? Each correct answer presents part o(the solution.

NOTE: Each correct selection is worth one point.

Options:

A- the let claim
B- the aud claim
C- the tld claim
D- the JSON Web Token (JWT) signature
E- the exp laim

Answer:

B, D, E

Question 5

Question Type: MultipleChoice

Case Study: ADatum Corporation

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 invoiced.

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 }
```

You need to publish build status messages to a Microsoft Teams channel.

Solution: You create a bot that monitors the build pipeline and pushes the messages to the channel by using the Microsoft Bot Framework.

Does this meet the goal?

Options:

A- Yes
B- No

Answer:

A

Explanation:

This solution does not meet the goal because creating a bot that monitors the build pipeline and pushes the messages to the channel by using the Microsoft Bot Framework requires additional coding and configuration.

A simpler and more efficient solution is to useAzure Pipelines app for Microsoft Teams1. This app allows you to easily subscribe to build status messages from Azure DevOps and receive them in your Teams channel without writing any code.

Alternatively, you can also useIncoming Webhook2orProject or Roadmap app3to send notifications to a Teams channel from other sources.

In addition, there are some recent news about Microsoft Teams that you might be interested in:

Microsoft Teams Allows Users To Mark Messages As Important Or Urgent4

Microsoft Teams is getting big performance improvements next month5

Question 6

Question Type: MultipleChoice

Case Study: ADatum Corporation

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 invoiced.

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 }
```

You plan to develop a SharePoint Framework (SPFx) web part by using the ReactJS framework.

You need to ensure that all the web part fields and controls adopt the theme of the site when you deploy the web part. The solution must minimize development effort.

What should you include in the solution?

Options:

A- Fluent UI
B- HTML5andCSS
C- Microsoft Fluid Framework
D- Material-UI

Answer:

A

Question 7

Question Type: MultipleChoice

Case Study: ADatum Corporation

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 invoiced.

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 }
```

You are building an app that will use the Microsoft Graph API and the Microsoft identity platform to enable users to perform the following tasks:

* Sign in to Azure AD.

* View all the Microsoft 365 groups that they own.

Each week, the app will also email the users a list of the Microsoft 365 groups to which they belong.

You need to identify which permissions to assign to the app. The solution must use the principle of least privilege.

What should you identify?

Options:

A- User.Read delegated, Group.Read application, and Mail.Send delegated permissions
B- User.Read delegated. Group.Read delegated, and Mail.Send delegated permissions
C- User.Read delegated, User.Read application, Group.Read application, and Mail.Send application permissions
D- User.Read delegated. Group.Read delegated, Group.Read application, and Mail.Send application permissions

Answer:

B

To Get Premium Files for MS-600 Visit
https://www.p2pexams.com/products/ms-600

For More Free Questions Visit
https://www.p2pexams.com/microsoft/pdf/ms-600

