Demo Questions

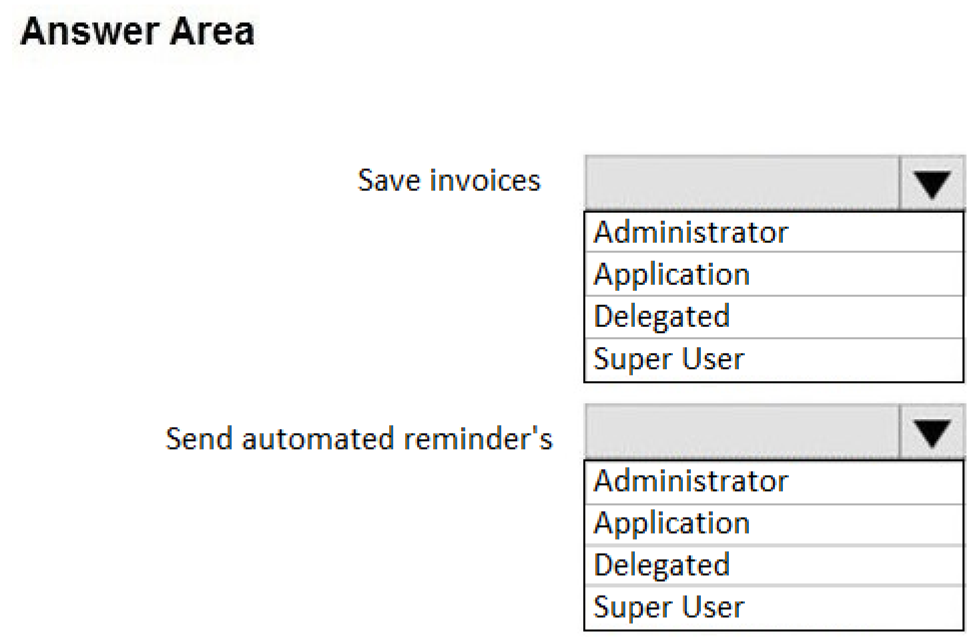
Microsoft MS-600 Exam

Building Applications and Solutions with Microsoft 365 Core Services (beta)

Question #1 ***Topic 1***

HOTSPOT -
You are developing an interactive invoicing application that will be used by end users. The application will have the following features:
✑ Save invoices generated by a user to the user's Microsoft OneDrive.
✑ Email daily automated reminders.
You need to identify which permissions to grant for the application features. The solution must use the principle of least privilege.
Which permission should you grant for each feature? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

Hot Area:



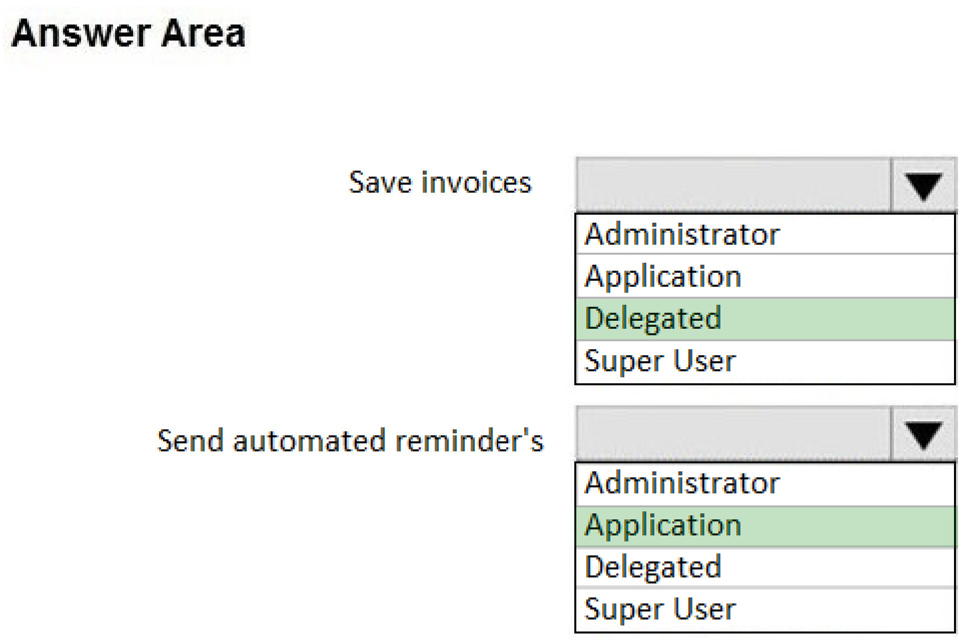
Answer Area

Save invoices

- Administrator
- Application
- Delegated
- Super User

Send automated reminder's

- Administrator
- Application
- Delegated
- Super User



Answer Area

Save invoices

- Administrator
- Application
- [✓ Delegated]
- Super User

Send automated reminder's

- Administrator
- [✓ Application]
- Delegated
- Super User

Correct Answer:
Microsoft identity platform supports two types of permissions: delegated permissions and application permissions.

Box 1: Delegated -
⚯ Delegated permissions are used by apps that have a signed-in user present. For these apps, either the user or an administrator consents to the permissions that the app requests, and the app

is delegated permission to act as the signed-in user when making calls to the target resource.

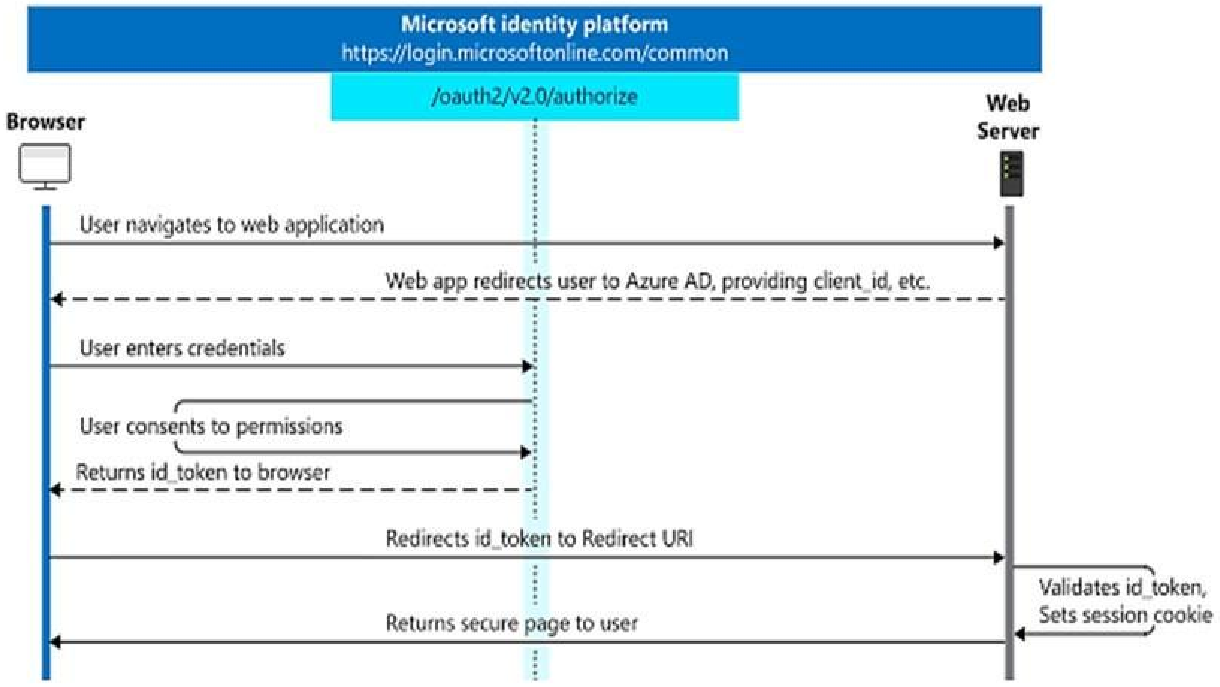
Box 2: Application -
⊸ Application permissions are used by apps that run without a signed-in user present; for example, apps that run as background services or daemons.
Application permissions can only be consented by an administrator.
Reference:
https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-permissions-and-consent

Question #2*Topic 1*

You need to develop a server-based web app that will be registered with the Microsoft identity platform. The solution must ensure that the app can perform operations on behalf of the user.
Which type of authorization flow should you use?

- [✓ A. authorization code]
- B. refresh token
- C. resource owner password
- D. device code

Correct Answer: *A*
In web server apps, the sign-in authentication flow takes these high-level steps:

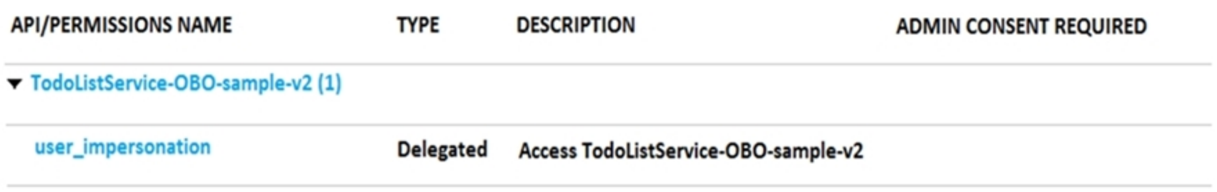

Microsoft identity platform
https://login.microsoftonline.com/common
/oauth2/v2.0/authorize
Browser
Web Server
User navigates to web application
Web app redirects user to Azure AD, providing client_id, etc.
User enters credentials
User consents to permissions
Returns id_token to browser
Redirects id_token to Redirect URI
Validates id_token, Sets session cookie
Returns secure page to user

You can ensure the user's identity by validating the ID token with a public signing key that is

received from the Microsoft identity platform endpoint. A session cookie is set, which can be used to identify the user on subsequent page requests.
In addition to simple sign-in, a web server app might need to access another web service, such as a REST API. In this case, the web server app engages in a combined OpenID Connect and OAuth 2.0 flow, by using the OAuth 2.0 authorization code flow.
Reference:
https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-app-types

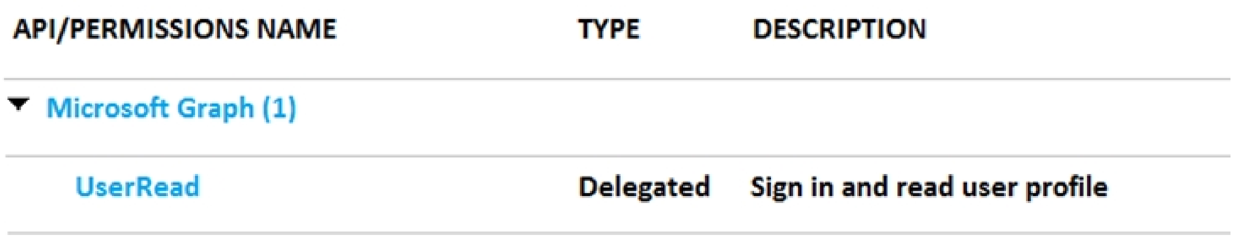
Question #3*Topic 1*

You have a single-page application (SPA) named TodoListSPA and a server-based web app named TodoListService.
The permissions for the TodoList SPA API are configured as shown in the TodoList SPA exhibit. (Click the TodoListSPA tab.)



API/PERMISSIONS NAME	TYPE	DESCRIPTION	ADMIN CONSENT REQUIRED
▼ TodoListService-OBO-sample-v2 (1)			
user_impersonation	Delegated	Access TodoListService-OBO-sample-v2	

The permissions for the TodoListService API are configured as shown in the TodoListService exhibit. (Click the TodoListService tab.)



API/PERMISSIONS NAME	TYPE	DESCRIPTION
▼ Microsoft Graph (1)		
UserRead	Delegated	Sign in and read user profile

You need to ensure that TodoListService can access a Microsoft OneDrive file of the signed-in user. The solution must use the principle of least privilege.
Which permission should to grant?

- A. the Sites.Read.All delegated permission for TodoListService
- B. the Sites.Read.All delegated permission for TodoListSpa
- C. the Sites.Read.All application permission for TodoListSPA
- D. the Sites.Read.All application permission for TodoListService

Correct Answer: *A*
A client application gains access to a resource server by declaring permission requests. Two types are available:

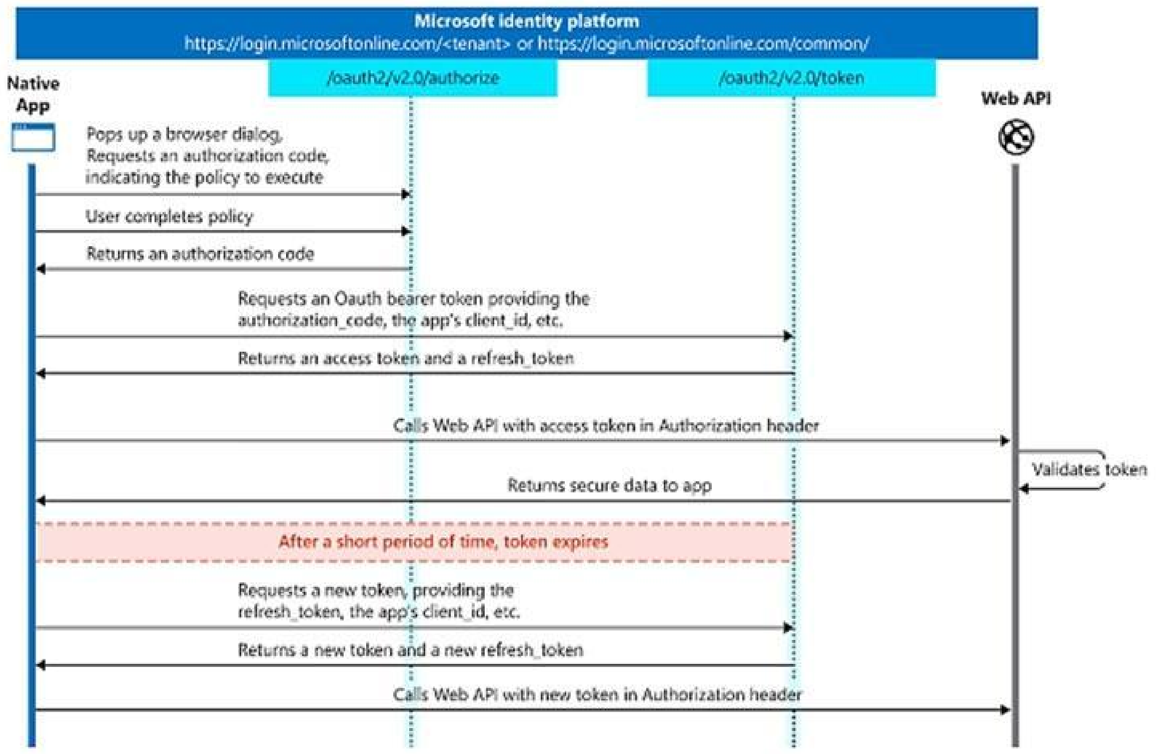
"Delegated" permissions, which specify scope-based access using delegated authorization from the signed-in resource owner, are presented to the resource at run-time as "scp" claims in the client's access token.
"Application" permissions, which specify role-based access using the client application's credentials/identity, are presented to the resource at run-time as "roles" claims in the client's access token.
Reference:
https://docs.microsoft.com/en-us/azure/active-directory/develop/developer-glossary#permissions

Question #4*Topic 1*

You are building a server-based web app that will use OAuth2 and will be registered with the Microsoft identity platform.
Which two values does the app require to obtain tokens from the Azure Active Directory (Azure AD) authorization endpoint? Each correct answer presents part of the solution.
NOTE: Each correct selection is worth one point.

- A. the tenant ID
- B. the context token
- [✓ C. the application ID]
- D. the application secret
- [✓ E. the authorization code]

Correct Answer: *CE*
C: The required client_id is the Application (client) ID that the Azure portal "" App registrations experience assigned to your app.
E: The authorization code flow begins with the client directing the user to the /authorize endpoint.

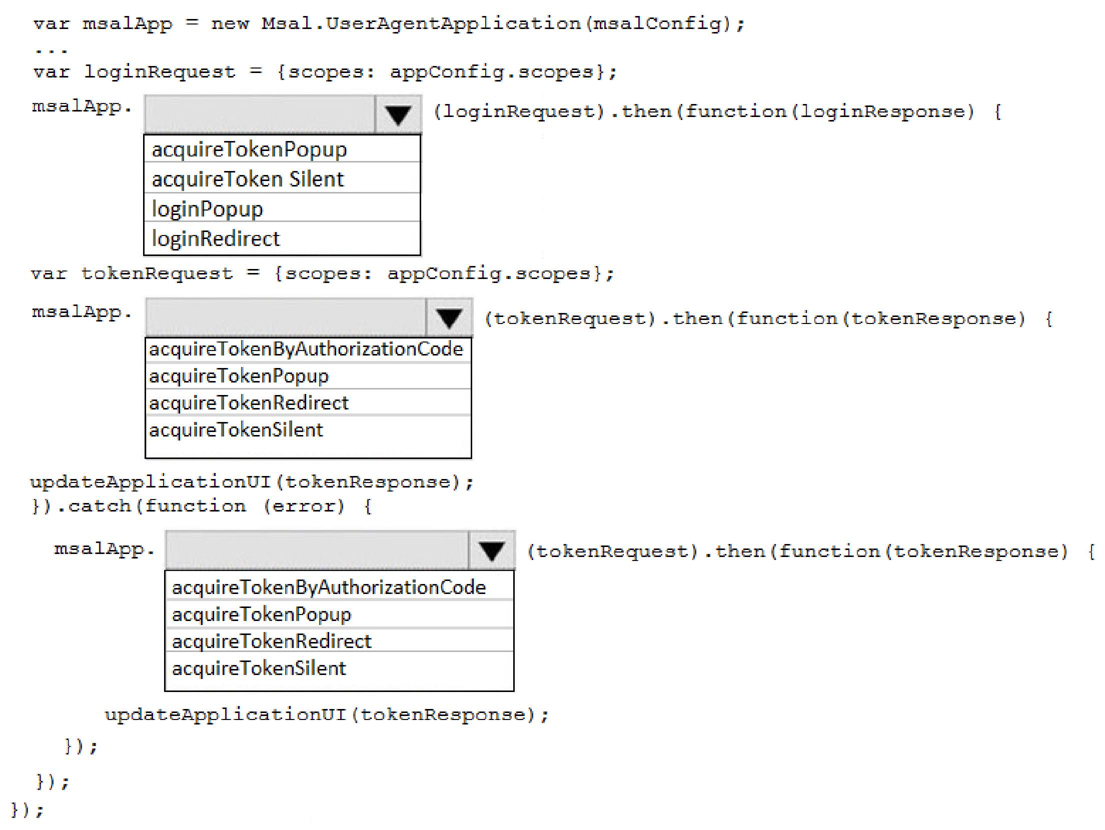

Microsoft identity platform
https://login.microsoftonline.com/<tenant> or https://login.microsoftonline.com/common/
/oauth2/v2.0/authorize
/oauth2/v2.0/token
Native App
Web API
Pops up a browser dialog, Requests an authorization code, indicating the policy to execute
User completes policy
Returns an authorization code
Requests an Oauth bearer token providing the authorization_code, the app's client_id, etc.
Returns an access token and a refresh_token
Calls Web API with access token in Authorization header
Validates token
Returns secure data to app
After a short period of time, token expires
Requests a new token, providing the refresh_token, the app's client_id, etc.
Returns a new token and a new refresh_token
Calls Web API with new token in Authorization header

Reference:
https://docs.microsoft.com/en-us/azure/active-directory/develop/v2-oauth2-auth-code-flow

Question #5*Topic 1*

HOTSPOT -
You are developing a single-page application (SPA).
You plan to access user data from Microsoft Graph by using an AJAX call.
You need to obtain an access token by the Microsoft Authentication Library (MSAL). The solution must minimize authentication prompts.
How should you complete the code segment? To answer, select the appropriate options in the answer area.
NOTE: Each correct selection is worth one point.

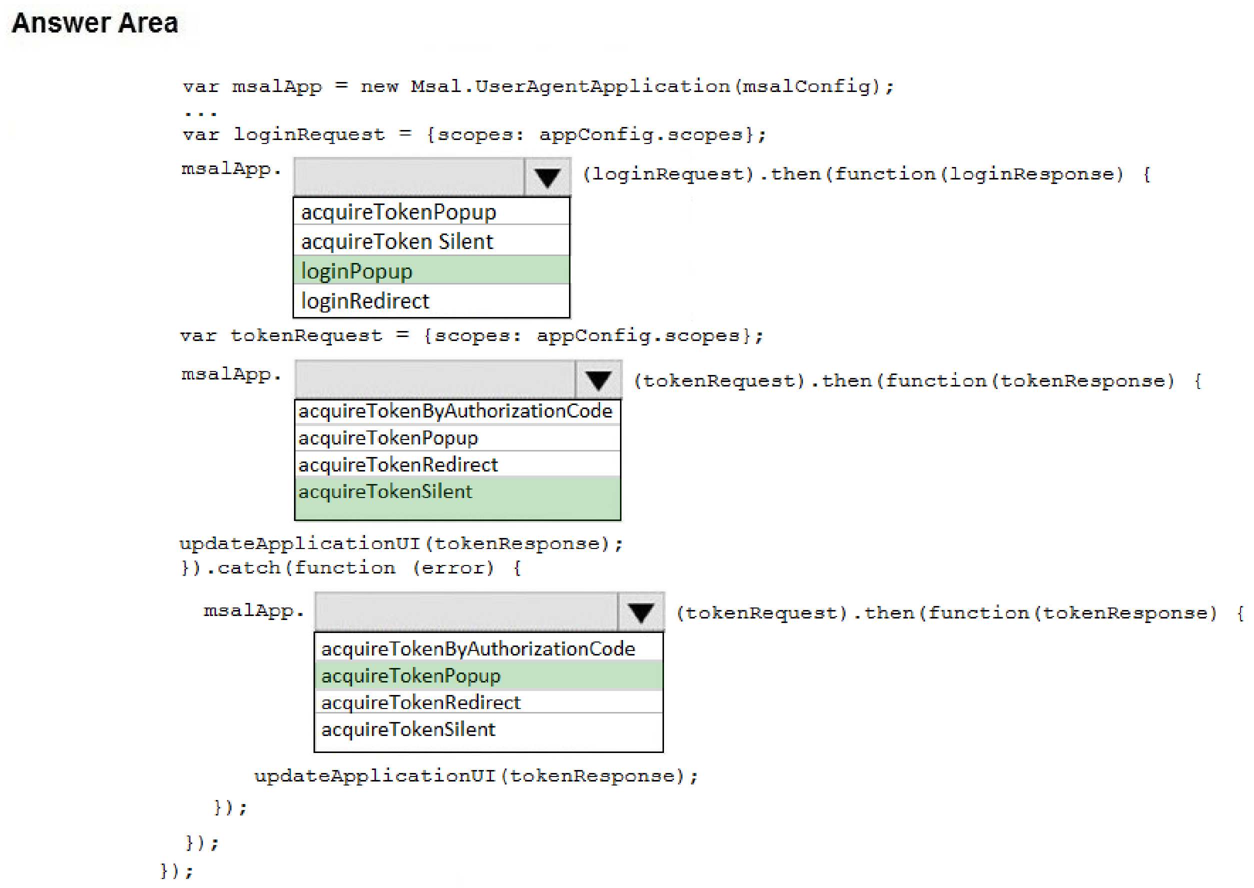
Hot Area:



Answer Area

```
var msalApp = new Msal.UserAgentApplication(msalConfig);
...
var loginRequest = {scopes: appConfig.scopes};
msalApp.[                    ▼] (loginRequest).then(function(loginResponse) {
        acquireTokenPopup
        acquireToken Silent
        loginPopup
        loginRedirect
var tokenRequest = {scopes: appConfig.scopes};
msalApp.[                    ▼] (tokenRequest).then(function(tokenResponse) {
        acquireTokenByAuthorizationCode
        acquireTokenPopup
        acquireTokenRedirect
        acquireTokenSilent
updateApplicationUI(tokenResponse);
}).catch(function (error) {
  msalApp.[                    ▼] (tokenRequest).then(function(tokenResponse) {
          acquireTokenByAuthorizationCode
          acquireTokenPopup
          acquireTokenRedirect
          acquireTokenSilent
      updateApplicationUI(tokenResponse);
    });
  });
});
```

Correct Answer:



Answer Area

```
var msalApp = new Msal.UserAgentApplication(msalConfig);
...
var loginRequest = {scopes: appConfig.scopes};
msalApp.[          ▼] (loginRequest).then(function(loginResponse) {
```

- acquireTokenPopup
- acquireToken Silent
- [✓ loginPopup]
- loginRedirect

```
var tokenRequest = {scopes: appConfig.scopes};
msalApp.[          ▼] (tokenRequest).then(function(tokenResponse) {
```

- acquireTokenByAuthorizationCode
- acquireTokenPopup
- acquireTokenRedirect
- [✓ acquireTokenSilent]

```
updateApplicationUI(tokenResponse);
}).catch(function (error) {
  msalApp.[          ▼] (tokenRequest).then(function(tokenResponse) {
```

- acquireTokenByAuthorizationCode
- [✓ acquireTokenPopup]
- acquireTokenRedirect
- acquireTokenSilent

```
      updateApplicationUI(tokenResponse);
    });
  });
});
```

Box 1: loginPopup -

Box 2: acquireTokenSilent -
The pattern for acquiring tokens for APIs with MSAL.js is to first attempt a silent token request by using the acquireTokenSilent method. When this method is called, the library first checks the cache in browser storage to see if a valid token exists and returns it. When no valid token is in the cache, it sends a silent token request to Azure Active Directory (Azure AD) from a hidden iframe. This method also allows the library to renew tokens.

Box 3: acquireTokenPopup -
//AcquireToken Failure, send an interactive request.
Example:
userAgentApplication.loginPopup(applicationConfig.graphScopes).then(function (idToken) {
//Login Success
userAgentApplication.acquireTokenSilent(applicationConfig.graphScopes).then(function (accessToken) {
//AcquireToken Success
updateUI();
}, function (error) {

```
//AcquireToken Failure, send an interactive request.
userAgentApplication.acquireTokenPopup(applicationConfig.graphScopes).then(function
(accessToken) { updateUI();
}, function (error) {
console.log(error);
});
})
}, function (error) {
console.log(error);
});
```

Reference:
https://github.com/AzureAD/microsoft-authentication-library-for-js/issues/339