**Level 1: Brief Introduction**

1. Introduce your self
2. Ask candidate to introduce himself/herself

**Level 2: Project specific discussion**

1. Previous experience
2. Technologies used
3. Achievements
4. Extra activities

**Level 3: JavaScript/C# coding**

1. Can you explain fundamentals of object oriented programming languages?
2. Which one of these fundamentals you have used recently?
3. Write a JavaScript function which will show a notification in the form when form is loaded. e.g. User is loding an Account form. A message, "Welcome {accountName} !", should be shown in the form notification section.
4. Write a C# code to design a simple calculator.

It should have below methods.

a. Add

b. Subtract

c. Multiply

d. Divide

Declare 2 variables at the start of the program and call these methods one after other.

The program should display the operation name and the output.

Consider exception handling wherever applicable.

**Level 4: CRM basics**

1. Can you explain the security model in CRM?
2. What type of customizations you have worked on?
3. How plugin is different from the Workflow and power automate flow
4. How do you handle exceptions in the plugin code?
5. What are deployment models for plugins
6. How do you debug the plugin code
7. What is the difference between JavaScript and Business Rule
8. **I have a business rule and a JavaScript on a CRM form. Which one execute first Java-Script or Business Rule**

* In Microsoft Dynamics CRM, the JavaScript code executes first, then the business rule. Business rules are server side scripts that get executed on the server, while JavaScript is client-side and gets executed on the client's browser. So, when a form containing both is loaded, the browser runs the JavaScript first, then the form data and business rules are sent to and processed by the server.

1. What are the different ways to fetch the data from a data verse table ? what are the limitations of one over the other?
2. There are certain fields on the CRM form. I would like to hide that field from certain users. Can we do this in the CRM ? If yes, how?
3. If you delete a record from dynamics CRM UI, what happens in database? Can you bring it back?

- When you delete a record from Dynamics CRM UI, that record is marked for deletion in the database; it doesn't get permanently deleted immediately. The process that handles the physical deletion of records is called "cascade delete operation."

All the related records will also be marked for deletion due to the cascading rule defined in the system. Please note that once a record is deleted through the UI, it cannot be directly restored.

To restore it, you would need to have a backup of the database from which you can restore the deleted data. Restoring data from a backup should only be done by a database administrator, given the high complexity of this task.

Another preventive measure might be to use Dynamics 365 CRM's Audit feature. This allows you to track changes made to CRM data, including record deletion, so that you can refer back to it when needed.

Always ensure to take regular backups of your CRM database to prevent unintended data loss.

1. What are the types of Relationship Behavior in MS Dynamics CRM
   1. Referential: With this type of behavior, all referenced records can be accessed from the primary record.
   2. Referential (Restrict Delete): Similar to referential, but the primary record cannot be deleted while it has related records in referenced entities.
   3. Parental: In this type, any operation performed on the parent record will automatically be applied to the child records. For example, if a user is changing the owner for a record, all related (child) records will also change their owner.
   4. Cascading: The actions taken on the parent record, such as deletion or assignment, are cascaded down to any child records.
   5. Configurable Cascading: Works like cascading, but gives users the ability to configure what actions to perform on related entities.
   6. Custom: This allows users to specify the behavior of actions such as Assign, Share, Unshare, Reparent, etc.
   7. None: There's no action performed on the related (referential) entity record when an operation such as Delete, Assign, Share, or Unshare is carried out on the primary entity record.

**Level 5: Power Apps**

1. How canvas app is different from model driven app
2. What are some common scenarios where you would suggest the use of model-driven apps over canvas apps?
3. I have a canvas app with a gallery control. I want to show the item in the right side panel when a particular item is selected from the gallery. How to do that?

* **IsSelected** is a Boolean value, evaluating to "true" or "false", for whether this item in the Gallery is selected or not. Only one item in our Gallery can be selected at any time, so this value can only be "true" for one item. If you put your app in **Preview** mode. Try selecting some different coffee makers and notice how the Rectangle control only appears to the one that you've selected. The visible rectangle also corresponds to the item displayed in the form next to the gallery.

1. **Different types of variables in canvas app? How do you declare those and when to use them?**

* There are three types of variables that you can define within a canvas app in Power Apps: global, context, and collections.

1. Global Variables: As the name suggests, these can be accessed from anywhere within the app across all screens. You declare a global variable using the Set function. For example, 'Set(varName, "value")' would set a global variable called 'varName' with a value of "value". You can then reference this variable anywhere in your app by using 'varName'. Use global variables when you need to carry the information across different screens.

2. Context Variables: These are restricted to the screen on which they're used, and they are created using the UpdateContext function or Navigate function. For example, 'UpdateContext({varName:"value"})' sets a context variable 'varName' to "value". Similarly, 'Navigate(Screen2, None, {varName :"value"})' passes the context variable to Screen2. Use context variables to carry data from one screen to another screen or to maintain the screen state.

3. Collections: Collections are used to store data as a table. You can build and manipulate collections using multiple functions like ClearCollect, Collect, and Remove. For instance, the function 'Collect(MyCollection, {Name: "John", Age: 30})' will create a collection called 'MyCollection' with two columns (Name, Age) and one entry. Use collections to manage and operate on groups of objects/values.

Remember: Variables and collections are held in memory and are discarded when you close the app, so it's usually best to use them for temporary storage. If you need permanent storage, consider using a data source instead.

1. **How to show in app notification for error, information and success**

* In Power Apps, you can show in-app notifications for errors, information, and success messages using the `Notify` function. Here's an example of how you can implement it:

1. Success Notification:```Notify("Operation Successful", NotificationType.Success)```

* The above statement will display a success notification with the message "Operation Successful".
* 2. Error Notification: ```Notify("An error has occurred", NotificationType.Error)```
* The above statement will display an error notification with the message "An error has occurred".
* 3. Information Notification:```Notify("Please check the details", NotificationType.Information)```
* The above statement will display an information notification with the message "Please check the details".
* The `Notify` function also allows you to customize the duration of the notification by adding a third parameter that represents the duration in milliseconds (the default is 10 seconds = 10000 milliseconds).
* ```Notify("Operation Successful", NotificationType.Success, 5000)```
* This will display a success notification that lasts for 5 seconds.
* 4. Warning Notification:
* ```Notify("Warning! Check the input details", NotificationType.Warning)```
* The above statement will display a warning notification.

1. **What are DLP policies in Power Apps**

* To safeguard data in your organization, you may use Power Apps to define and enforce policies that restrict which consumer connectors can share with certain business data. These policies are known as data loss prevention (DLP) policies.
* DLP policies ensure that data is controlled uniformly across your organization and prevent essential company data from being accidentally shared to connections such as social networking sites.DLP policies can be defined and managed at the tenant or environment level using the Power Platform admin center.

1. **What is a DataCard in the form?**

* Inside of the form, when you select a Field, that field is portrayed as a **DataCard** in the Tree view. Notice how each **DataCard** contains four different controls. The two most important controls in your **DataCard** are the label control, which will initially include the name **DataCardKey**, and the text input control, which will initially include the name **DataCardValue**. The other two controls are optional and depend on whether the data is required or not. The **StarVisible** is a text label with an asterisk in it that appears if you designate that field as required (either in your form or in the data). The **ErrorMessage** is another text label that appears if you try to submit a form and that input is missing. You can look at the Visible properties for these controls. **Visible** evaluates to: "true" or "false". The **And()** function you'll see in the **StarVisible** control means that both conditions, separated by a comma, must be true for the function to evaluate as "true".

1. **Write a pseudocode/steps to create a Power automate flow. Get all the accounts available in the dataverse table. Assign account numbers(AccountNumber is a custom column) sequentially.**
2. **What is Power Apps delegation**

* Delegation is the point at which the expressiveness of Power Apps formulas reaches the need to limit network data transfer. In short, rather than transmitting data to the app for local processing, Power Apps will outsource data processing to the data source.
* Working with huge data sets necessitates the use of delegated data sources and formulae. It's the only way to keep your app running smoothly and guarantee users have access to all of the information they require.

1. **How can you import data into a model-driven app?**

* Data can be imported into a model-driven app using the built-in 'Get Data' feature. You can also use external services like Power Automate or Azure Data Factory to automate data import.

1. **How to handle errors in the power apps?**

* To do so, first, we will have to get information about any errors through the Errors function. And then, through Validation and DataSourceInfo, some of the errors can be ignored even before they take place.
* The Canvas App introduced the 'IfError' and 'isError' functions, to manage errors and display the appropriate message.
* Formula-level error management must be enabled in order to use these features. To enable it, follow the steps below:  
  To begin, start the Canvas App and choose File.  
  Under Settings, select Advanced Settings.  
  Allow for formula-level error management.

1. **How to embed a canvas app on the form and making it context aware**

* Here are the steps to embed a canvas app on a form and make it contextually aware in Microsoft Dynamics 365:
* 1. Navigate to the model-driven form designer, and open the form you want to add the canvas app to.
* 2. Select "Insert"-> "Canvas app".
* 3. Configure your canvas app.
* 4. To make your canvas app context aware, you need to set the 'ModelDrivenFormIntegration' parameter.
* Here is a step-by-step explanation:

- Open the canvas app in the Power Apps studio.

- On the 'Insert' tab, click on 'New screen' to add a screen to your app.

- Choose 'ModelDrivenFormIntegration' from the Data panel on the right-hand side.

- In the formula bar, type 'ModelDrivenFormIntegration'. You will see a set of properties under it like 'Data', 'Display Name', 'Errors', 'Item', etc.

- Click on the 'Item' property. Here you can retrieve any field from the current entity record that the form is displaying.

- Finally, use the information to display the relevant content or perform actions in the canvas app as per the context of the form record.

* By using the 'ModelDrivenFormIntegration' object you can make your embedded canvas app contextually aware and thereby interact with the form in which it is embedded.

1. **how to make a power app responsive**

* You can make a Power Apps responsive by using the App's properties and formulas, and taking advantage of a feature called "containers". Here are the steps:
* 1. Set properties: Open the Power Apps Studio, click on `File`, then `App settings`, and then `Screen size + orientation`. Change the screen size by setting `Width` and `Height` to the maximum expected size, and then switch on the `Scale to fit` option.
* 2. Use Relative Positioning: If you want your app to be responsive, you want controls to move and adjust based on screen size. Instead of using hard-coded X & Y positioning, use controls like Parent.Height or Parent.Width, which change dynamically based on the parent control or screen.
* 3. Use ‘Flexible height gallery’: Flexible height galleries have property wrap count. You can use the formula like: If(App.Width < App.Height, 2, 4) which simply means if the orientation is portrait use wrap count as 2 else 4 – this will auto adjust the number of items seen in a gallery based on orientation.
* 4. Use Containers: In Modern App Designer, containers are the recommended control to use for grouping and arranging other controls in your app. The Canvas is a free-form surface, but the controls inside containers can be arranged either in Flexible width or Flexible height format which helps in building responsive interfaces.
* 5. Apply relative sizes to controls: Like positioning, the height and width of the controls should also be relative. You can use parent control height, screen width etc. to adjust the height and width of controls.
* 6. Use media queries: Media queries help in delivering different styles to different devices based on their characteristics or parameters like device-width, device-height, device-dpi, etc.
* Remember, making an app truly responsive requires careful design and consideration of the various screen sizes and orientations your app users might have.

1. **Is it possible to use the canvas app in a model-driven app?**

* Canvas apps can be added to model-driven forms in the same way that other custom controls can. Rich data integration capabilities in an embedded canvas app integrate contextual data from the host model-driven form into the embedded canvas app.

**Level 6: Project Process**

1. Which process is followed by your team for SDLC? What is your role in it?
2. What is agile methodology?
3. What is scrum? What is the team size?
4. What is the sprint length?

**Level 7: Azure concepts**

1. **I have a power app which provides media upload facility to the end user. What** considerations you have to make in designing the app in terms of storing the media files?

* Up to 200 MB of media for each app can be uploaded to Power Apps. However, what is majorly recommended is using media/blog storage services, such as Azure Media or Azure Storage, and embedding the media URL to the app.

1. **What is the difference between Power Apps and logic apps?**

* Azure Logic Apps is a service that allows you to integrate apps, build workflows, and more. Power Apps are used to create graphical user interfaces. Since almost everything can be referred to as an application or an app, it's not logical to suppose that all applications are similar in some manner.

1. **What are Azure functions?**

* Azure Functions is a serverless compute service that enables developers to build, deploy, and manage small pieces of code without worrying about server infrastructure. These functions can be triggered by various events, such as HTTP requests, timer-based schedules, or data changes.

1. **How the Azure functions are invoked?**

* Azure Functions are typically invoked or triggered based on specific events or conditions. Here are some ways Azure Functions can be invoked:

1. HTTP Trigger: This allows the function to be invoked upon receiving an HTTP request. This is ideal for creating APIs or web-hooks.

2. Timer Trigger: With this, the function is executed at set intervals, based on a time schedule that you define (for example, every 15 minutes or every Sunday at 8AM).

3. Blob Trigger: This trigger runs the function whenever a new item is added to a Azure Blob Storage, or when an existing blob gets updated.

4. Queue Trigger: The function is executed when a new item is added to an Azure Queue Storage.

5. Cosmos DB Trigger: This trigger allows the function to execute when there are changes to data in an Azure Cosmos DB.

6. Event Grid Trigger: With this, the function gets invoked whenever an event is fired in Azure Event Grid.

7. Event Hub Trigger: This is similar to Event Grid trigger, but specific to Azure Event Hub.

8. Service Bus Trigger: This enables your function to be executed when a message lands in a specific Azure Service Bus Queue or Topic.

9. IoT Hub Trigger: This trigger runs the function whenever an Azure IoT Hub receives a message.

Each of these triggers can be configured when you create a new Azure Function, and the parameters of the trigger (like blob name, queue message, schedule, etc.) can be used as inputs for your function.

1. **Can you write an Azure function with HttpTrigger? The function should return the message as,**

**"Hello World!"**

[FunctionName("HttpTriggerExample")]

public static async Task<IActionResult> Run(

[HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpRequest req,

ILogger log)

{

log.LogInformation("C# HTTP trigger function processed a request.");

// Your code here

return new OkObjectResult("Hello, Azure Functions!");

}

1. **How does Azure Functions handle scaling automatically?**

Azure Functions scale automatically based on the number of incoming requests or events. For example, if you have a function handling incoming messages from a queue, as the queue load increases, Azure Functions will provision additional instances to process the messages in parallel, ensuring timely processing.

1. **Explain Azure Durable Functions.**

Azure Durable Functions allow you to create complex, stateful workflows by orchestrating multiple Azure Functions. For instance, you can create a durable workflow that involves multiple functions to process an order, update inventory, and notify customers when their order is complete.

**Level 8: Additional topics**

1. **Have you worked on any ETL tool?**