Module 4: Make canvas apps with Power Apps

Scenario

You are a functional consultant for your organization Contoso. You are assigned to work on a project for your client Fabrikam. You have been assigned to continue work on the Fabrikam Knowledge canvas app that we started creating in the prior module. In this practice you will be working with the Microsoft Dataverse connector to filter the list, save the results of the assessment in Microsoft Dataverse and add support for submitting feedback using the Edit Form.

Exercise 1 – Filtering Data

Task 1 – Filter Knowledge Assessments

In this task, you will filter the Knowledge Assessment to show only Active rows that have Start Date in the past and End Date is in the future.

1. Navigate to https://make.powerapps.com.
2. Make sure you are in your environment.
3. Select **Solutions**.
4. Select **Assessment**.
5. Select the **Fabrikam Assessment** Canvas application.
6. Click on the **Edit** button located on the command bar.
7. Wait for the app designer to load.
8. Select the **Knowledge Assessment List** gallery and click the **Advanced** tab in the Properties pane.
9. Set the **Items** Property to the snippet below. This snippet will filter the Knowledge Assessment rows and the list will show only the active rows with Start Date in the past and End Date in the future. *Note:* if you type this instead of pasting it, you will see how the editor helps you build expressions.

Filter('Knowledge Assessments', (Status = 'Status (Knowledge Assessments)'.Active && 'Start Date' <= Today() && 'End Date' >= Today()))

10. Click **File** and **Save** your changes.
11. Click on the **Back** button.
12. Do not close the app designer.

Task 2 – Get Current User

In this task, you will get the current User and save it in a global variable. We are doing this during OnStart so that it only happens once, and the data is available for use elsewhere in the app. We will be using this to retrieve a filtered list of test results submitted by this user.

1. In the tree view, select the **App.**
2. Select the **OnStart** property and set it to snippet below. This snippet will create a global variable **UserPrimaryEmail** that will hold the current user's email.

```
Set(UserPrimaryEmail, User().Email)
```

3. Add the snippet below to the **OnStart** property, after the snippet already there. This snippet will first terminate the first function with semicolon, get the current User and save it in a global variable name **CurrentUser**.

```
;Set(CurrentUser, LookUp(Users, 'Primary Email' = UserPrimaryEmail))
```

4. Add the snippet below to the **OnStart** property, after the snippets already there. Add the following function to work around an existing bug that does not properly load the metadata for related properties. In the future this workaround will not be required.

```
;Set(FirstKABugWorkaround,First('Knowledge Test Results').'Knowledge Assessment')
```

5. Your **OnStart** property should now have the above three snippets. Click **File** and **Save**.
6. Select the **back** button to navigate back to the **Fabrikam Assessment** app designer.

Task 3 – Save Total Points

In this task, you will create a Patch that will create a Knowledge Test Result row. Using the Patch function allows us to on demand create a row with just specific properties being passed. In this case, we will be using the sum function to get a total of our points that are stored in our in-memory collection based on the answers the user provided.

1. Expand the **Take Assessment Screen**.
2. Locate the button and rename it **Score Button**.
3. While you still have the **Score Button** selected, select the **Advanced** tab of the Properties pane. Set the **OnSelect** property to snippet below. This snippet will create a new Knowledge Test Result row. **Note:** For long formulas, you can expand the **fx** bar to format your formula in a larger view.

```
Patch('Knowledge Test Results',Defaults('Knowledge Test Results'),{'Knowledge Assessment':'Knowledge Assessment List'.Selected,Name:'Knowledge Assessment List'.Selected.Title, 'Total Points': Sum(UserAnswers.Points,Points)});
```

Task 4 – Add Feedback Screen

In this task, you will add a new screen to the applications. This screen will let the user submit feedback. This tasks demonstrates how to use the EditForm to create a new row.

1. Click on the ellipses button of the **Take Assessment Screen** and click **Duplicate Screen**. We are creating a duplicate screen because we want all our screens to look the same and we don't want to recreate the header every time we add a new screen.
2. Rename the duplicate screen **Add Feedback Screen**.
3. Rename the Button on the **Add Feedback Screen** to **Submit Feedback Button**.
4. Change the **Text** property of the **Submit Feedback Button** to **"Submit Feedback"**.
5. Click on the ellipses button of the **Assessment Question List** inside the **Add Feedback Screen** and click **Delete**.
6. Select the **Add Feedback Screen**.
7. From the **Insert** tab, click **Forms** and then click **Edit.**
8. In the Property panel on the right, select **Feedback** for **Data Source**. (If you are already on the Advanced tab, you will need to switch to the Properties tab.)
9. Go to the tree view and rename the form **Feedback Form**.
10. Resize and reposition the **Feedback Form** to your liking.
11. From the tree view, expand the **Feedback Form**.
12. Select the **Title** column. In the **Advanced** pane, select **unlock** and go to the **Data** section.
13. Set the **Default** property to the **Title** of the selected **Knowledge assessment**.

```
'Knowledge Assessment List'.Selected.Title
```

14. With the **Title** data card still selected, select the **Properties** tab. Locate the **DisplayMode** property and set it to **View**.

```
DisplayMode.View
```

15. Rename the **Comments_DataCard** to **User Comments**. (You may need to click **Unlock** first.)
16. Expand the **User Comments** data card and rename the **DataCardValue** to **User Comments Text**.
17. Select the **Submit Feedback Button**.
18. Replace the **OnSelect** property value with the snippet below. This snippet will submit the form, reset the form, and navigate back to the previous page.

```
SubmitForm('Feedback Form');ResetForm('Feedback Form');Back(ScreenTransition.None)
```

19. With the **Submit Feedback Button** still selected, return to the **Properties** tab and select the **DisplayMode** property (click on the words **Display mode**).

20. Replace the **DisplayMode** property value with snippet below. This snippet will enable the button if the comments filed has value and disable it if the comments field is blank.

```
If(IsBlank('User Comments Text'), DisplayMode.Disabled, DisplayMode.Edit)
```

Note: The **Source** and **Rating** datacards can be removed from the **Feedback Form** since we will not be using them in this lab. The next two steps will remove those from the **Feedback Form**.

21. Select the ellipses button of the **Rating_Datacard** inside the **Add Feedback Screen >> Feedback Form** and select **Delete**.
22. Select the ellipses button of the **Source_Datacard** inside the **Add Feedback Screen >> Feedback Form** and select **Delete**.
23. Select the **Take Assessment Screen**.
24. Go to the **Insert** tab, navigate to **Icons** in the top menu, and select **Left.**
25. Resize and place the icon on the left side of the header.
26. Set the **Color** property of the icon to **White**.
27. Set the **OnSelect** property of the icon to the snippet below. This snippet will navigate back to the previous page.

```
Back()
```

28. Select the **Take Assessment Screen**.
29. Click **Insert**, navigate to **Icons** in the top menu, and select the **Emoji - Smile** icon.
30. Place the icon in the right side of the header and next to **username.**
31. Set the **Color** property of the icon to **White**.
32. Set the **OnSelect** property of the Emoji icon to the snippet below.

```
NewForm('Feedback Form');Navigate('Add Feedback Screen', ScreenTransition.None)
```

33. Click **File** and **Save** to save your application.