Mastering Power Apps

Lab Manual

Contents

Lab 0 – Preparation

Summary

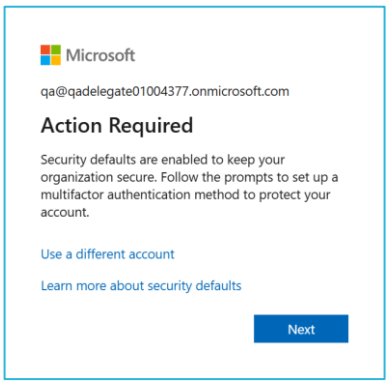
In this lab, you will prepare your browser environment, download student files, and then set up your Office 365 account. This will allow each student to work in a safe and isolated environment while completing the rest of the course labs.

Task 1 – Sign into your course Office 365 account

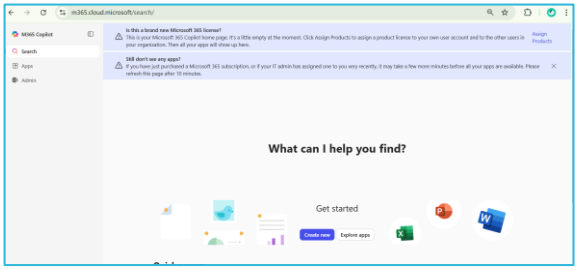
1. Navigate to https://m365.cloud.microsoft/
2. Sign in using the Microsoft 365 credentials supplied to you by your instructor.

Note: For all further labs – these credentials are referred to as you Global Administrator/Global Admin.

3. Note: As part of this you will need to set up Multi Factor Authentication (MFA) on the account. This is due to Security Defaults being enabled. We will disable Security defaults in the next task.
4. In the Action Required prompt, select Next.



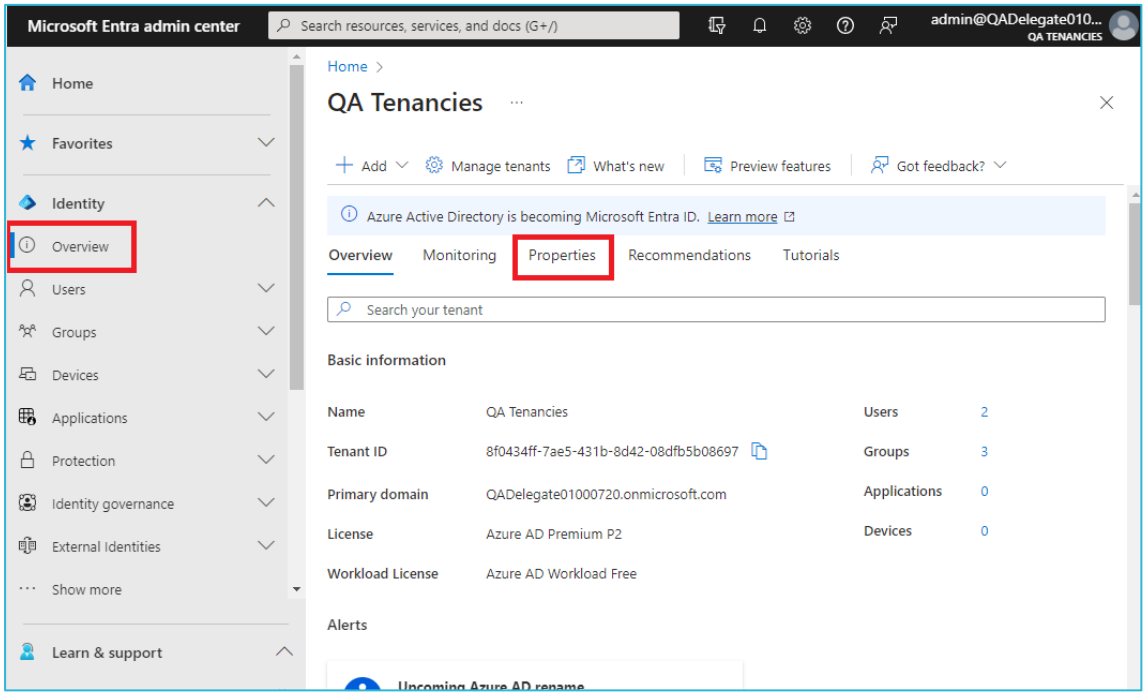
5. Follow the prompts to set up MFA.
6. At the Stay signed in? prompt, click Yes.
7. Close any first-run experiences. You should now be on the home page of M365 Copilot.



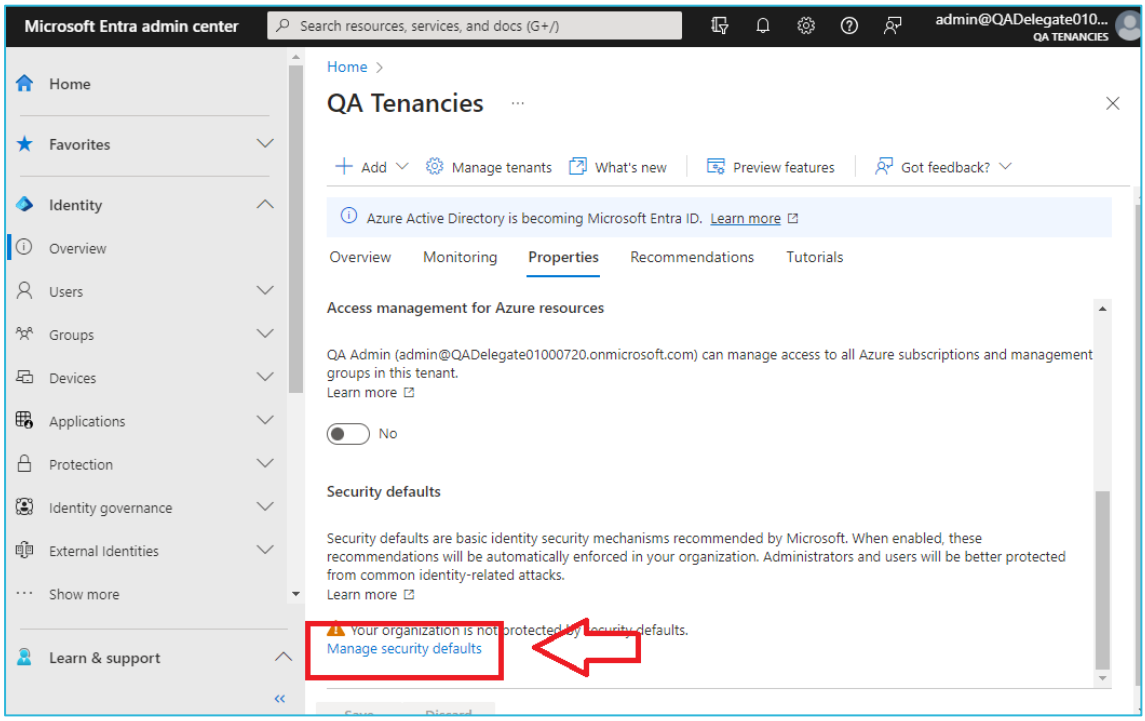
Task 2 – Disable Entra Identity Security Defaults

Note: This task is not part of normal setup and is not recommended in a production environment. However, completion of this task will make the learner experience easier by removing the requirement to setup multi-factor authentication.

1. In the web browser, open a new tab and navigate to the Microsoft Entra admin center. This admin center can be found at the following URL. https://entra.microsoft.com/.
2. If necessary, sign in with your global admin account.
3. In the navigation pane, ensure that Overview is selected in the Identity group.
4. In the main pane, select Properties.

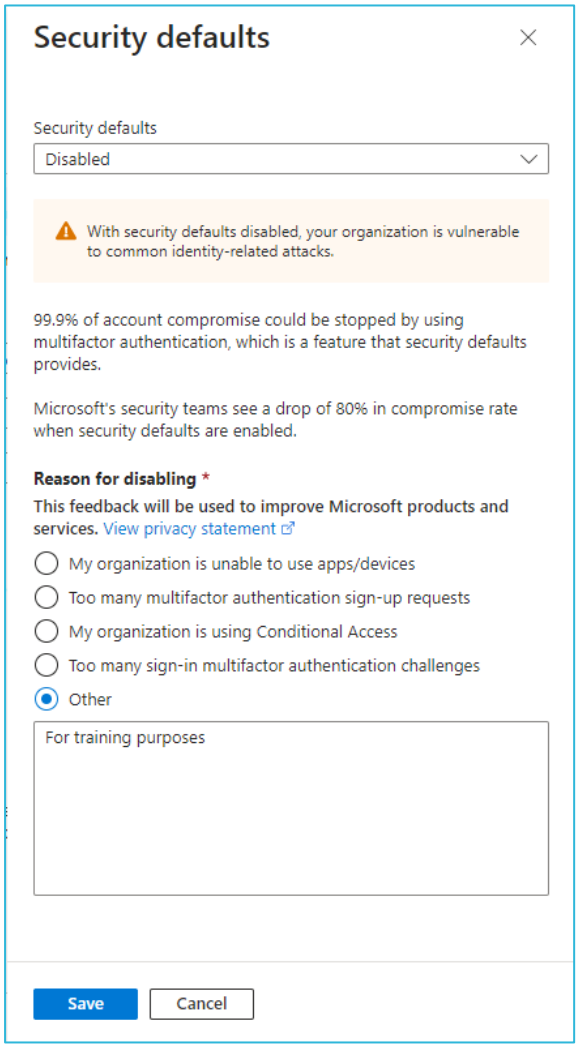


5. Scroll to the bottom of the pane and then select the Manage Security defaults link.



6. In the Enable Security defaults pane that opens, select Disabled for the Security defaults setting.

7. In the Reason for disabling section, select the Other radio button, and then type the message For training purposes in the text box.



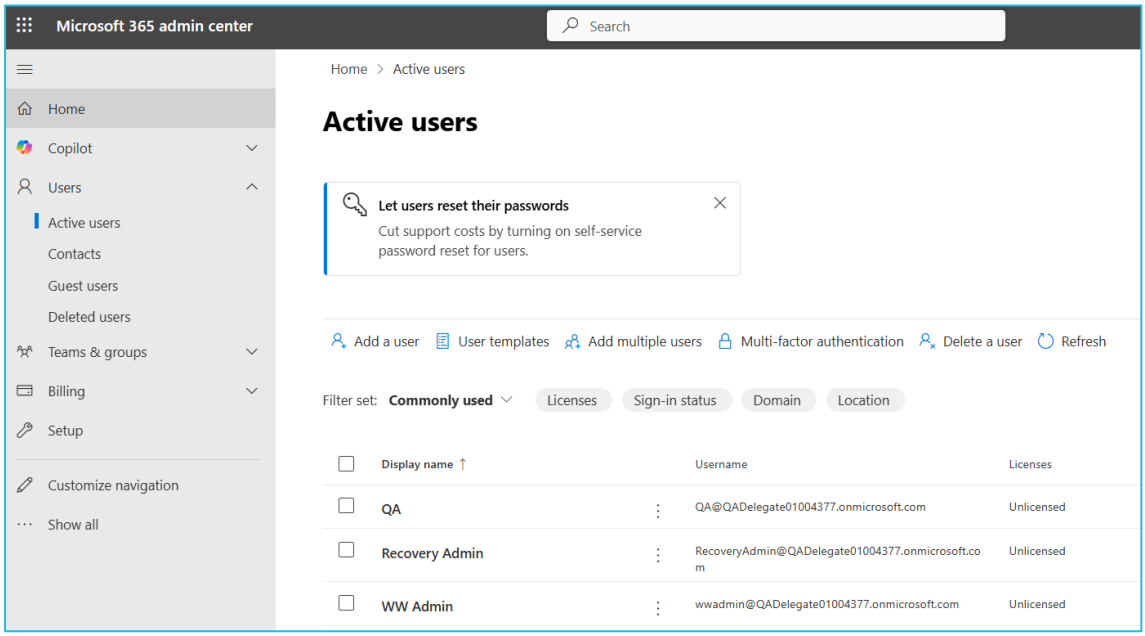
8. Click Save.
9. In the confirmation box that appears, select Disable.
10. Close the browser tab for Microsoft Entra admin center.

Task 3 – Assigning a Microsoft 365 License to the Global Administrator account

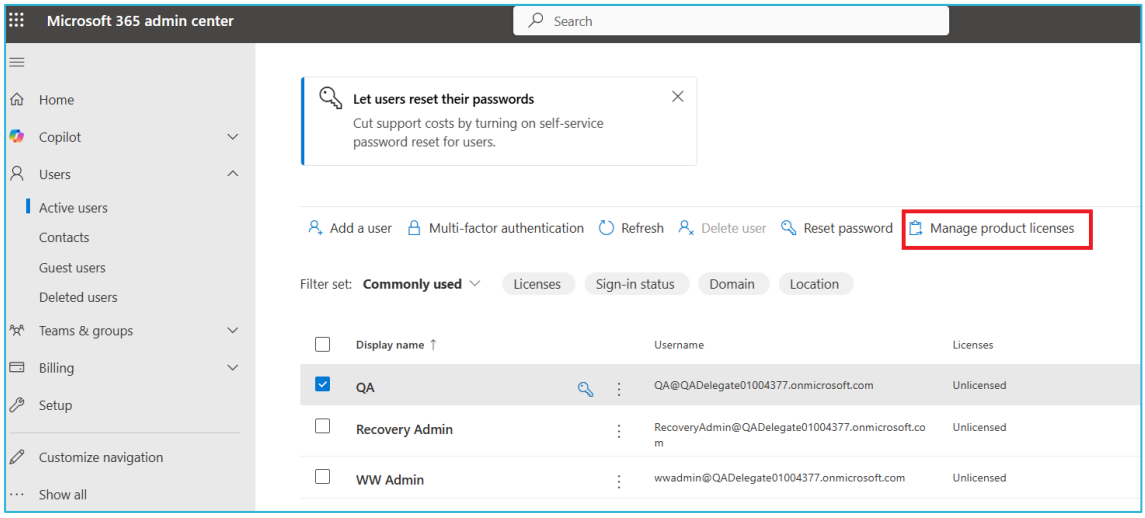
1. From a web browser, open https://admin.microsoft.com
2. Expand Users – Active Users.

3. Select your Global Admin Account – QA.

Important – Please do not change or remove the Recovery other accounts during the course.



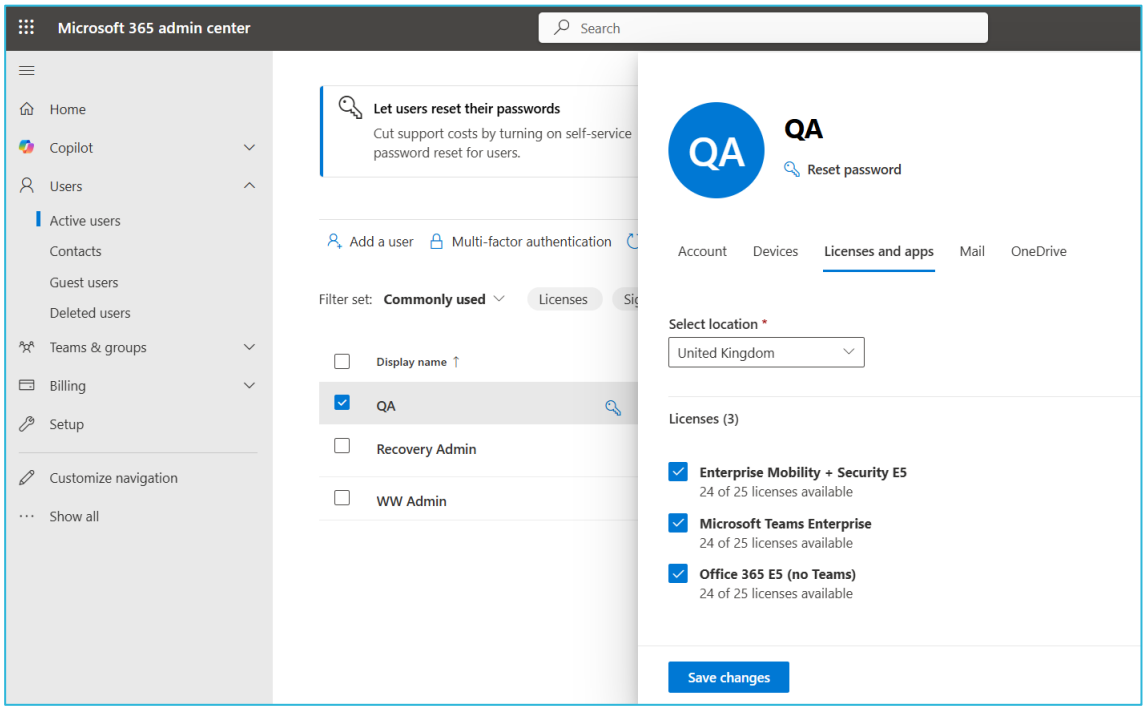
4. Click Manage Product Licenses.



5. Select All available licenses.

Note – The exact licenses may differ depending on the course, options could include Enterprise Mobility + Security E5, Microsoft Teams Enterprise, Office 365 E5 (no Teams) and others.

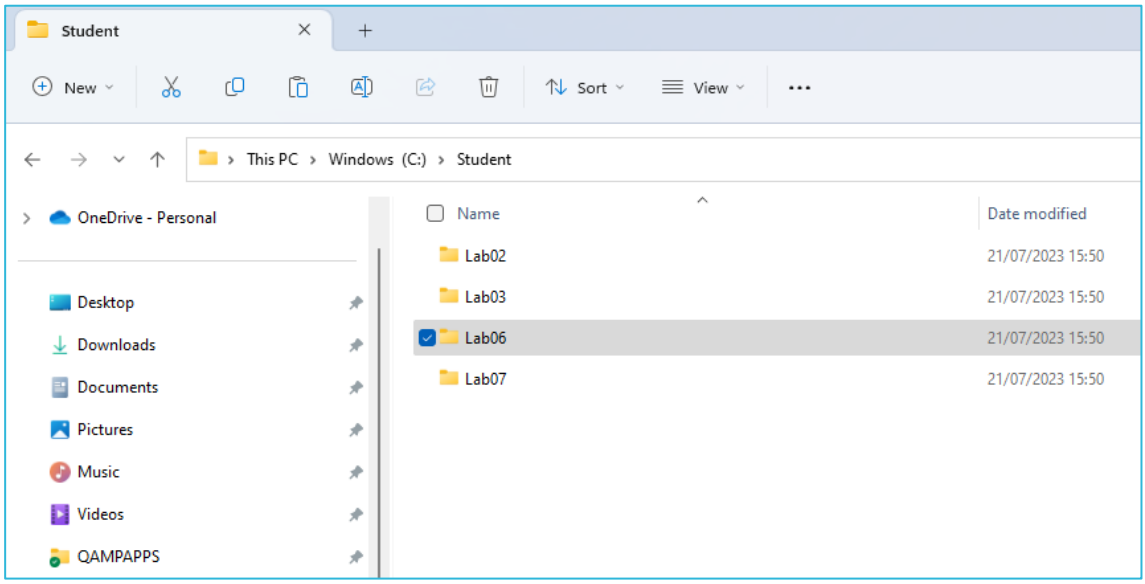
6. Click Save changes.



7. Close the properties pane and then close the Microsoft 365 Admin center tab.

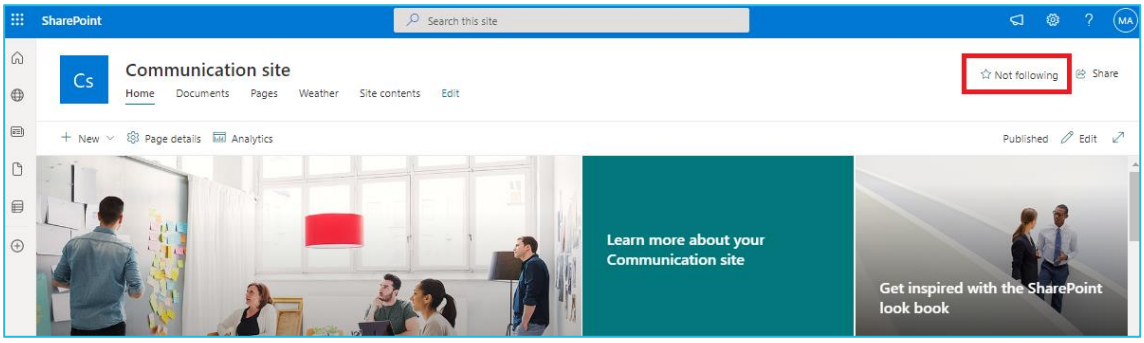
Task 4 – Download student files

1. In your web browser, navigate to https://github.com/QA365-co-uk/QAMPAPPS
2. Select the code button and then select Download ZIP.
3. Once the file has downloaded, open it, expand the QAMPAPPS-main folder, and then copy the Student folder to the root of your C:\ drive.



Task 5 – Confirm service functionality

1. Open the Outlook web client (https://outlook.office.com) and confirm you can access the account's email.
2. In the browser, open OneDrive for Business and confirm that your library has been created.
3. Open SharePoint and confirm you can access the communication site that has been created. You may want to mark this is a 'favorite' site by following it (Click on Not following).
4. Note: Your SharePoint URL will have the format https://QADelegate01000xxx.sharepoint.com (where xxx is your assigned student number).



5. Confirm you can open Power Apps.

Lab 01a – Advanced Controls: Part 1

Summary

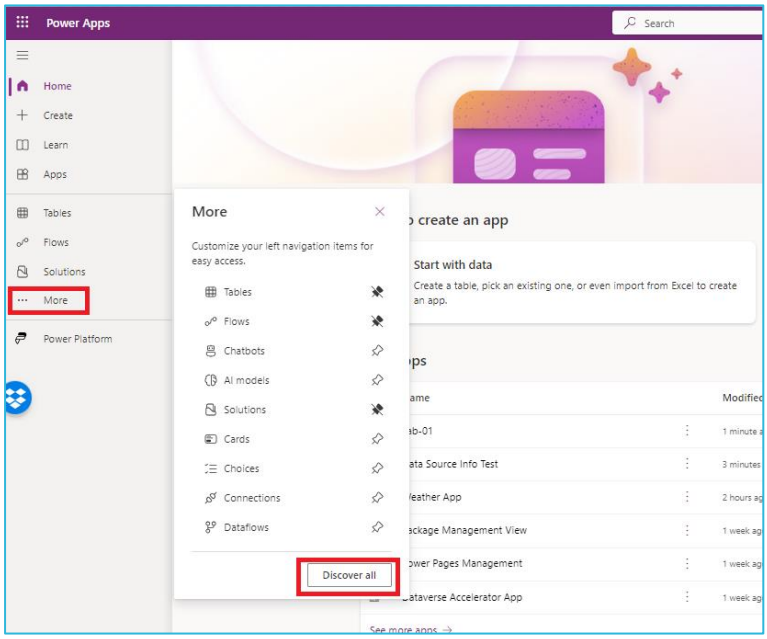
In this lab, you will create a canvas app that uses several of the controls mentioned in the associated module, including components, a timer control, a customized gallery, and a chart. For simplicity, this lab will use data stored in an Excel file that you will upload to One Drive for Business.

Task 1 – Lab preparation

1. In your course Chrome instance, sign into Office 365 and then open OneDrive.
2. If necessary, select the link that says Your OneDrive is ready.
3. Select Upload and then choose Files.
4. Choose the file C:\Student\Lab01\TaskList.xlsx to upload it.
5. The file should now be visible in your OneDrive library.

Task 2 – Create a component library

1. Switch to a new tab in your course Chrome instance.
2. Navigate to https://make.powerapps.com.
3. If "Component libraries" do not appear in the Power Apps navigation pane, then select More and then select Discover all.

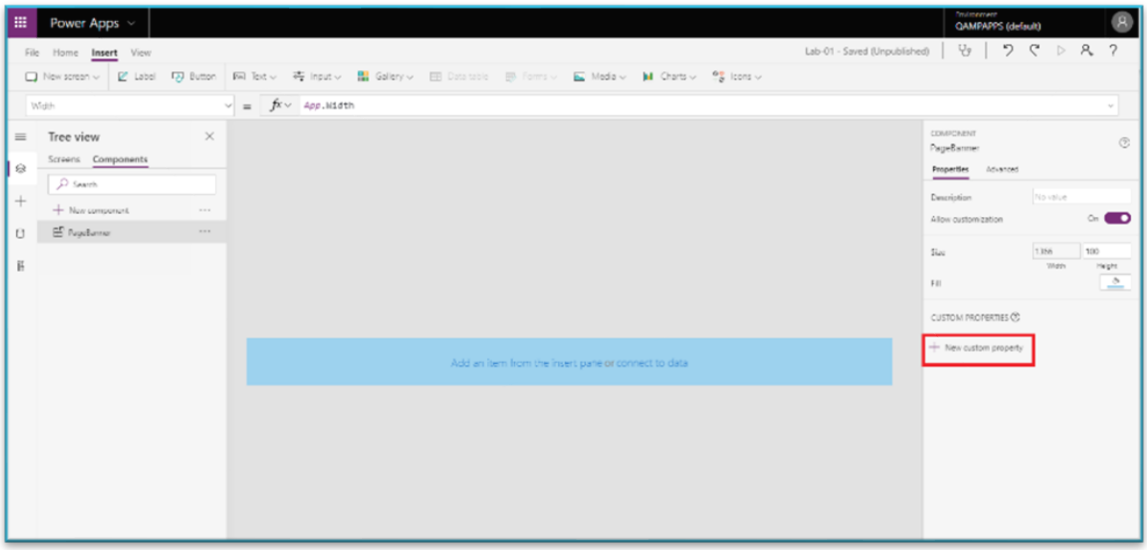


4. In the App enhancement section find and then select the pin icon beside Component libraries.

5. Select Component libraries in the navigation pane.
6. Select Create component library.
7. In the Component library dialog, enter the name Course Shared Components, and then select Create.
8. In the Tree view, rename the component (Component1) to PageBanner.
9. Set the following properties on the PageBanner component:

Property	Value
Fill	RGBA(153,207,236,1)
Height	100
Width	App.Width

10. With the PageBanner component selected, select New custom property in the property pane.



11. In the New custom property pane, enter the following values and then select Create.

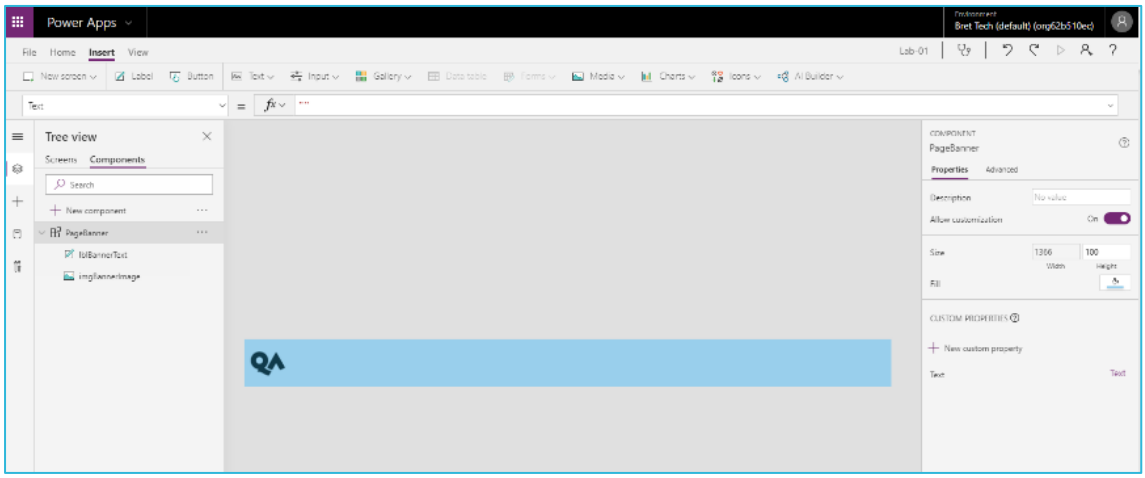
Property	Value
Display name	Text
Name	Text
Description	Page Banner Text
Property Definition	Input
Data Type	Text

12. The new property will now be listed under custom properties.
13. Set the value of the Text property of the component to be "" (empty double quotes).
14. Insert a text label control to PageBanner, and set the following property values:

Property	Value
Name	lblBannerText
Align	Align.Center
X	0
Y	0
Height	Parent.Height
Size	44
Text	PageBanner.Text
Width	Parent.Width

15. Add an Image control (Media section) to PageBanner and set the following property values

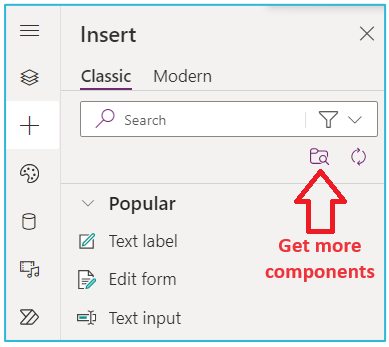
Property	Value
Name	imgBannerImage
X	0
Y	0
Height	100
Image	QALogo
Width	100



16. Save your component library.
17. Publish the component library.
18. Select the Back button to leave the app (library), and return to the Power Apps home page. You should see your component library appear.

Task 3 – Create a new canvas app and import a component

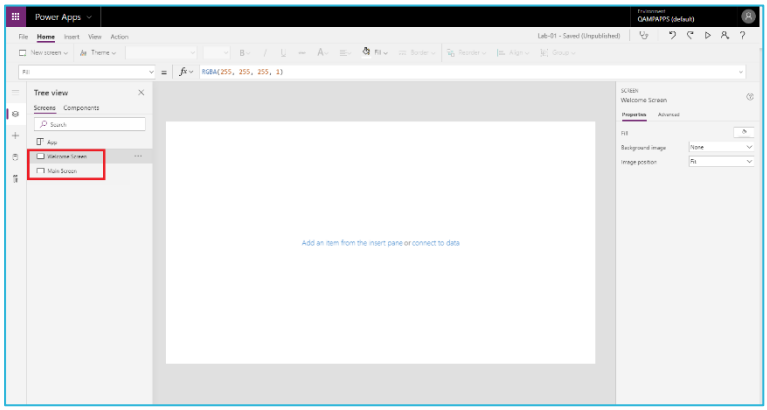
1. Create a new Canvas app from blank.
2. Give the app the name Lab-01 and choose the tablet format.
3. Select Create.
4. If necessary, deal with the welcome messages.
5. Using the settings menu, pick an appropriate icon and colour.
6. Save the app.
7. Using the Insert pane select Get more components (Folder with a search icon – just under the main search box in the Insert pane).



8. Under Course Shared Components, select PageBanner and then select Import.
9. Return to the Tree view.

Task 4 – Using a timer control

1. Make sure that you are editing app Lab-01 and then navigate to Screens in the tree control. You should see the default screen, named Screen1.
2. Rename Screen1 to Welcome Screen.
3. Add a new (blank) screen and rename it Main Screen.



4. Set the Fill property of both screens to RGBA(153,207,236,1). This is the same value we used for the banner component.

5. To each screen, insert a copy of the PageBanner component that you imported in the previous task (it is in the Library components section) and then set the properties of each as shown below.

Property	Welcome Screen Banner	Main Screen Banner
Name	comWelcomeScreenBanner	comMainScreenBanner
Text	"Welcome"	"Main Screen"

6. Add a Timer control to the Welcome Screen, move it towards the bottom right of the screen and set the following properties:

Property	Value
Name	timCountdown
AutoStart	true
Duration	5000
OnTimerEnd	Navigate('Main Screen')
Visible	false

7. Add a Text label control to the Welcome Screen and configure the following properties:

Property	Value
Name	lblRedirectMessage
X	75
Y	260
Align	Align.Center
Height	200
Size	24
Text	"You will be re-directed to the main page, in " & RoundUp(5-timCountdown.Value/1000,0) & " seconds"
Width	Parent.Width - 150

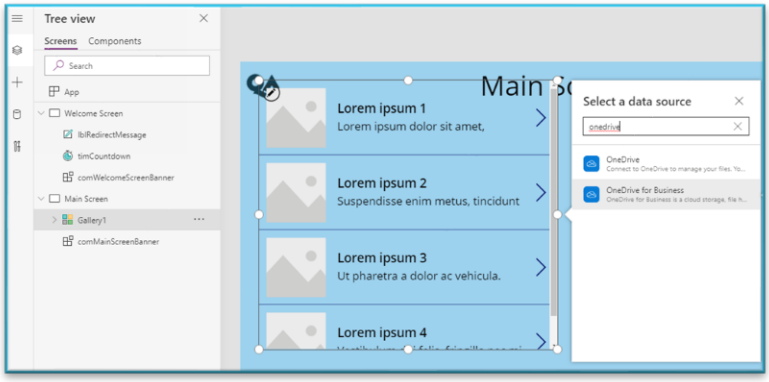
8. Save your app, then from the Welcome Screen preview the app to confirm that the timer and redirection is working.

Lab 01b – Advanced Controls: Part 2

Task 1 – Customizing a Gallery

In this app, we want a table-like view of the data that we are using. However, the table control is quite limiting in what we can do and customize with it, so instead we will be using a gallery and configuring it to look like a table.

1. On the Main Screen, insert a Vertical gallery. When the Select a data source dialog appears, type OneDrive into the search box and then select OneDrive for Business.



2. If necessary, in the details pane, select connect.
3. Select the file TaskList.xlsx (the file that we uploaded in Task 1) and then select the Tasks table.
4. Select Connect.
5. Rename Gallery1 to galTasks.
6. With galTasks selected, use the properties pane, to change the Layout to Title.
7. Set the following properties on galTasks:

Property	Value
X	40
Y	160
Height	500
TemplateFill	If(ThisItem.IsSelected,ColorFade('Main Screen'.Fill,0.5),'Main Screen'.Fill)
TemplateSize	64
Width	765

8. In the tree control, expand galTasks and then rename the current Text label control to lblGalTasks-Task. (The text label will have the name of TitleX where X is a number.)
9. Set the following properties on lblGalTasks-Task:

Property	Value
X	32
Y	15
Height	34
Size	20
Width	315

10. Copy lblGalTasks-Task and then paste it twice into the template of galTasks (i.e., make sure the copies appear as child objects of the gallery, not of the screen).
11. Rename the copied text labels as lblgalTasks-DueDate and as lblgalTasks-PercentComplete.
12. Set the following properties on the two copied text labels:

Property	lblGalTasks-DueDate	lblGalTasks-PercentComplete
X	374	534
Y	15	15
Align	Align.Left	Align.Right
Height	34	34
Size	20	20
Width	156	72

13. With the gallery galTasks selected, select the Edit link beside Fields in the gallery property pane.
14. In the data pane, select the following values for the fields.

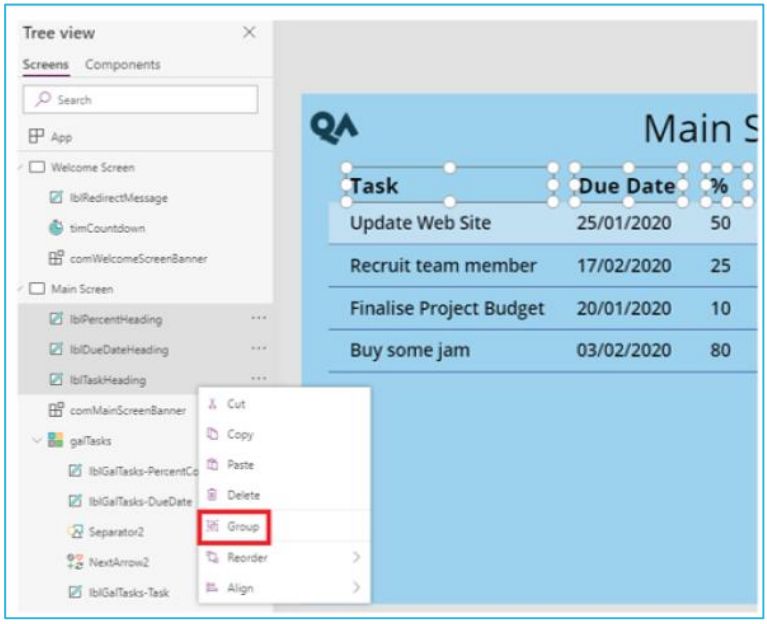
Field Name	Value
lblGalTasks-DueDate	DueDate
lblGalTasks-PercentComplete	ProgressPercent
lblGalTasks-Task	Task

15. Close the data pane.
16. Select lblGalTasks-DueDate in the Tree view, and then edit the Text property to read: Text(ThisItem.DueDate,DateTimeFormat.ShortDate).

17. Add three text labels to the Main Screen and set the following properties:

Property	First Text Label	Second Text Label	Third Text Label
Name	lblTaskHeading	lblDueDateHeading	lblPercentHeading
X	68	412	610
Y	111	111	111
Align	Align.Left	Align.Left	Align.Left
FontWeight	FontWeight.Bold	FontWeight.Bold	FontWeight.Bold
Height	52	52	52
Size	24	24	24
Text	"Task"	"Due Date"	"%"
Width	311	163	63

18. In the Tree view, use the ctrl key to select the three labels you have just added and edited, and then right-click and choose group.



19. Rename the new group as grpTaskGalleryHeaders.

Task 2 – Add a View Form

By customizing a gallery, we have complete control over what information we want to see, however, often there will be too much information to easily see in such a view. With this in mind, we are going to add a form so we can see all the information we want from the currently selected record.

1. Insert a Display form to the Main Screen.
2. Using the Display property pane, select Tasks as the Data source.
3. Select 0 selected and then select Add field.

4. Select all the fields and then select Add.
5. Reorder the fields so they are in the following order:
   - Task
   - Notes
   - Priority
   - ProgressPercent
   - DueDate
   - DateAdded
   - DateUpdated
6. Close the Fields pane.
7. Configure the form with the following properties:

Property	Value
Name	frmViewDetails
Width	460
X	850
Y	100
Height	560
Item	galTasks.Selected
Columns	1
Layout	Horizontal

8. In the Tree view, expand the form, and then select the DueDate_DataCardX.
9. Using the Advanced property pane, unlock the card.
10. Edit the Default property of the card to read:
    Text(ThisItem.DueDate,DateTimeFormat.ShortDate)
11. Repeat the above process for the other two date datacards.
12. Save the app.

Task 3 – Enabling New and Edit Functionality

1. On the Main screen add a button and set the following properties:

Property	Value
Name	btnNewTask
OnSelect	Set(varMode,"New")
Text	"New Task"
Width	160
X	50
Y	680

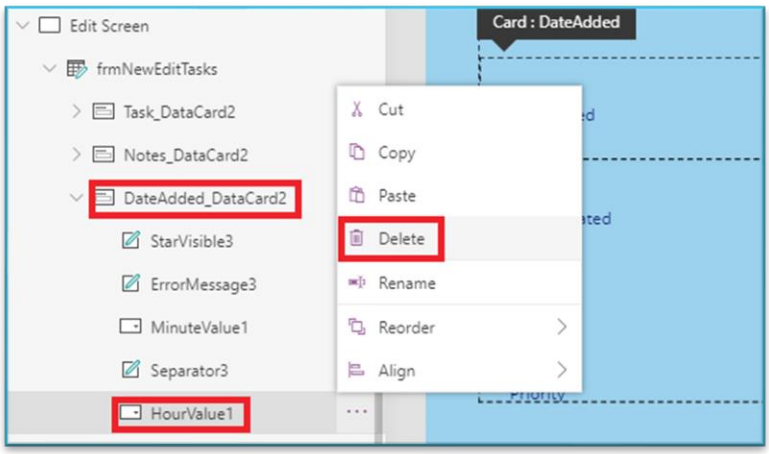
2. Using the alt key, select the button to run the OnSelect code to populate the variable.
3. Add a new (blank) screen named Edit Screen and set the fill property to be the same as the other screens RGBA(153,207,236,1).
4. Add a PageBanner component and set the text to varMode & " Task Screen".
5. Rename the banner comEditScreenBanner.
6. Insert an Edit Form and rename it frmNewEditTasks.
7. Select Tasks as the Data source.
8. Select 0 selected, and then add the fields in the following order: Task, Notes, DateAdded, DateUpdated, DueDate, Priority, ProgressPercent.
9. Close the Fields pane.
10. Configure the form with the following properties:

Property	Value
Columns	1
Layout	Horizontal
Height	490
Item	galTasks.Selected
Width	800
X	54
Y	114

The date-time cards have all been added with both date and time value. We only need dates so will delete the three control for hours and minutes. This will introduce some errors that we will then have to fix.

11. Expand frmNewEditTasks in the Tree view to show the data cards.
12. Select the DateAdded data card, and then select Advanced in the property pane.
13. Select the padlock icon to unlock the data card.

14. Expand the DateAdded data card and then delete the HourValue dropdown.



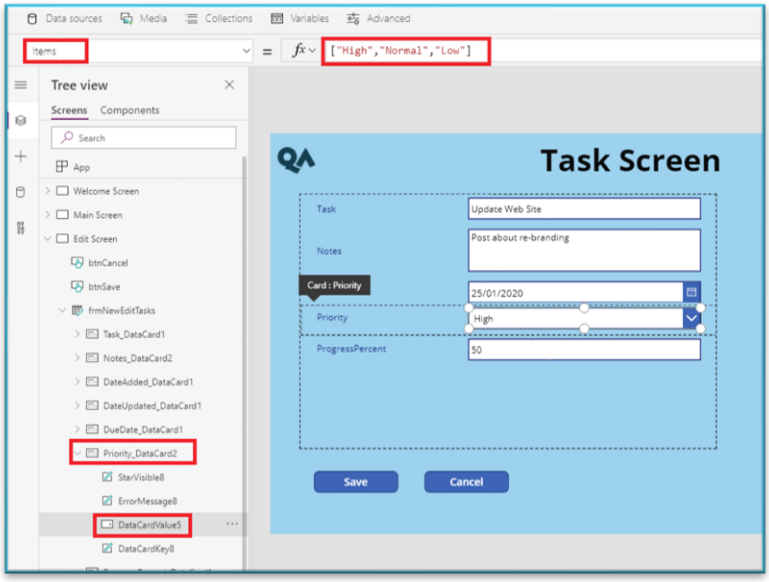
15. Delete the MinuteValue control.
16. Delete the Separator control.
17. You should now see some red crosses on the screen by the data card indicating an error. Select the left cross and then choose Edit in the formula bar.
18. The first edit should be for the Update property of the card. Delete the first + and everything after it so the property reads DateValueX.SelectedDate.
19. Repeat step 17 replacing the shown value with 0 until the errors are all dealt with.
20. Repeat steps 12-19 for the other two data cards that have dates in them.
21. These deletions will also have affected the layout of the controls in the data cards. To fix this for each of the three affected data cards, change the Y value of the DataCardKey to 10.

Note: The DateAdded and DateUpdated fields need to be in the card so that we can set their values when adding new and editing tasks; however, they are not fields that we want to interact with in the form. We will programmatically set values and then hide the controls.

22. Expand the DateAdded_DataCard and then select the DateValueX.
23. Set the DefaultDate property to:
    If(varMode="New",Now(),galTasks.Selected.DateAdded)
24. Select the DateAdded card and then set its Visible property to false.
25. Expand the DateUpdated_DataCard and then select the DateValueX.
26. Set the DefaultDate property to If(varMode="New","",Now()).
27. Select the DateUpdated_DataCard and set its visible property to false.

Note: We also want to update two other datacards to provide better functionality.

28. Select the frmNewEditTasks form and then in the property pane, select the link beside the Fields section (it will say something like 7 selected).
29. If necessary, expand the Priority field so the Control type is visible. Change the control type to Allowed Values.
30. Close the fields pane, then unlock the Priority_DataCard.
31. Select the DataCardValueX for the Priority_DataCard and update its Items property to be: ["High","Medium","Low"].



32. Unlock the ProgressPercent_DataCard and then select its DataCardValue.
33. Set the Default property to If(varMode="New",0,galTasks.Selected.ProgressPercent).
34. Add two buttons to the Edit Screen and set the following properties.

Property	Button 1 Value	Button 2 Value
Name	btnSave	btnCancel
OnSelect	SubmitForm(frmNewEditTasks); Navigate('Main Screen')	ResetForm(frmNewEditTasks); Navigate('Main Screen')
Text	Save	Cancel
X	84	296
Y	648	648

35. Return to the Main Screen and select btnNewTask.
36. Update its OnSelect property to the following code:

```
Set(varMode,"New");NewForm(frmNewEditTasks);Navigate('Edit Screen')
```

37. In the Tree view, expand galTasks and then select the NextArrow icon.
38. Set its properties as below.

Property	Value
Name	icoEditTask
Height	47
Icon	Icon.Edit
OnSelect	Set(varMode,"Edit"); EditForm(frmNewEditTasks); Navigate('Edit Screen')
Width	65
X	690
Y	10

39. Save the app and then test the New and Edit functionality.

Task 4 – Adding a Chart and other finishing touches

In this task, we will add a few finishing touches to our app. The reality is that the steps here have been added primarily as an example of what can be done and to demonstrate specific techniques. In a live app, you may choose to approach these in a different way.

We will be adding a chart to the space currently occupied by the view form. The first step will be to add a control to toggle the viewing of the chart/gallery.

1. Add a toggle control (Input menu) to the Main Screen and then configure the following properties:

Property	Value
Name	tglChartVisible
FalseText	“Show Chart”
TrueText	“Hide Chart”
Width	184
X	1130
Y	690

2. Edit the visible property of frmViewDetails to !tglChartVisible.Value
3. Using the alt button, toggle the control and confirm that the form becomes hidden. Leave it hidden for now.

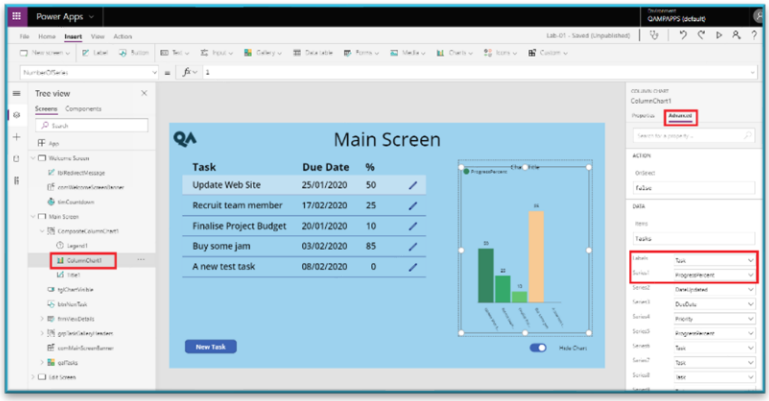
4. Add a column chart to the main screen and set the following properties for the CompositeColumnChartX component:

Property	Value
X	915
Y	131
Height	519

5. Set the following properties to the ColumnChart subcomponent:

Property	Value
Items	Tasks
ItemsGap	10
NumberOfSeries	1
SeriesAxisMax	100

6. With the ColumnChart subcomponent still selected, select advanced in the properties pane.
7. In the data section, set the value of Labels to be Task and Series1 to be ProgressPercent.



8. Delete the LegendX subcomponent of the chart.
9. Change the text of the TitleX subcomponent to "Task Progress" and its VerticalAlign to VerticalAlign.Top.
10. Set the visible property of the CompositeColumnChartX to tglChartVisible.Value
11. You should now be able to toggle between the chart and the form.
12. Add a happy icon (Emoji – Smile) to the main screen and set the following properties:

Property	Value
Name	icnHappySad
Color	See below
Height	128
Icon	See below
Visible	!tglChartVisible.Value
Width	144
X	1010
Y	495

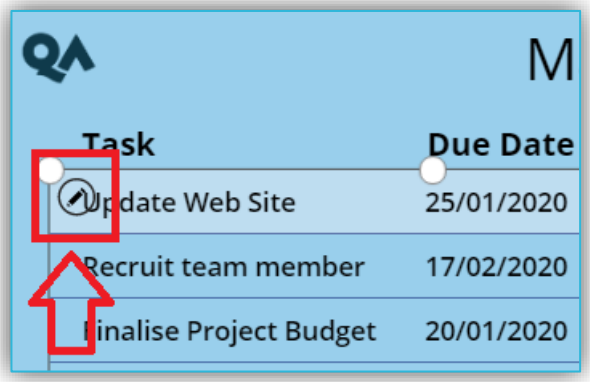
Color code:

```
If(Value(galTasks.Selected.ProgressPercent)<25,Color.Red,Value(galTasks.Selected.ProgressPercent)<50,Color.Orange,Value(galTasks.Selected.ProgressPercent)<75,Color.Blue,Color.Green)
```

Icon code:

```
If(Value(galTasks.Selected.ProgressPercent)<25,Icon.EmojiSad,Value(galTasks.Selected.ProgressPercent)<50,Icon.EmojiFrown,Value(galTasks.Selected.ProgressPercent)<75,Icon.EmojiNeutral,Icon.EmojiSmile)
```

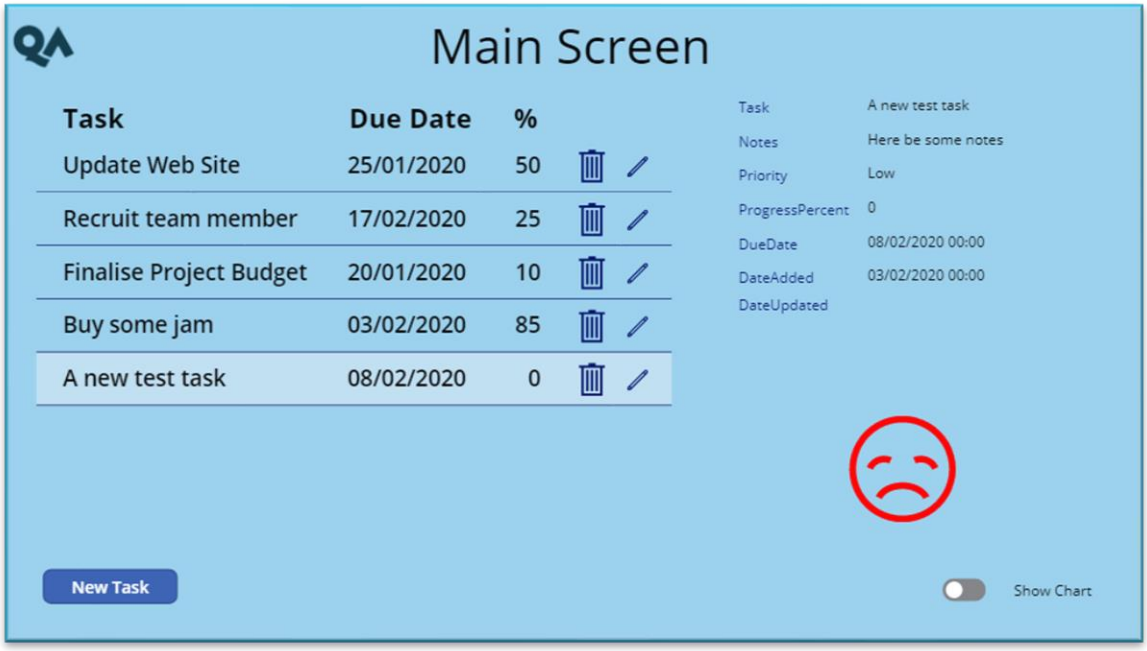
13. The icon should now change based on the item selected and should disappear with the form.
14. Select the galTasks gallery and then select the pen icon to edit the gallery template.



15. Insert the trash icon into the gallery. Make sure that you see multiple copies of the icon. If you only see one then you have added it to the screen, not to the gallery. If this is the case, delete the icon and try again.
16. Set the following properties to the trash icon.

Property	Value
Name	icoDeleteTasks
Height	45
OnSelect	Remove(Tasks,ThisItem)
Width	52
X	642
Y	10

17. Save the app and test the full functionality.



Lab 02 – Responsive app

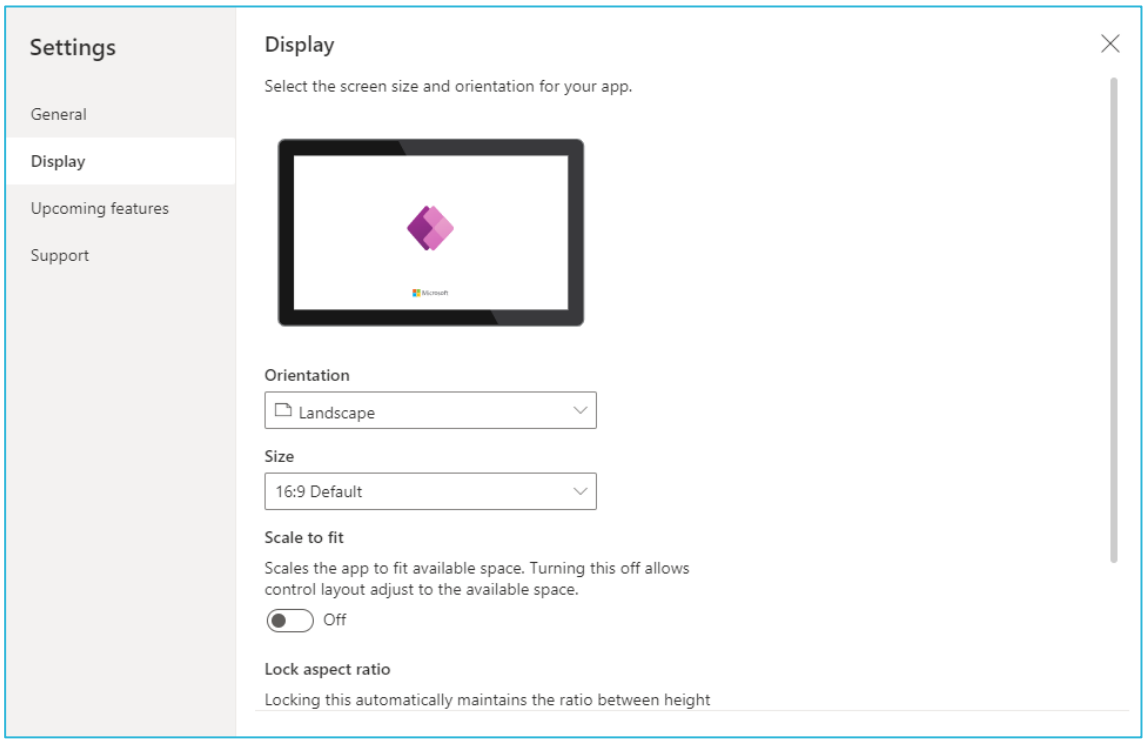
Summary

In this lab, you will create an app that is responsive (reacts to changes in the size of the app). You will take advantage of more traditional techniques as well as the more recent responsive container controls. In this lab, many of the properties required to make apps responsive can only be found in the properties pane.

This app won't actually do anything, but you will be able to see the responsive control working when you run it.

Task 1 – Create app

1. Create a new blank canvas app choosing the tablet layout.
2. Save the app as Lab 02.
3. Rename the screen Input Screen and set the fill to RGBA(153,207,236,1).
4. Select settings on the menu and then in the Display section turn Off Scale to fit.



5. Using the Insert pane, insert a Vertical container (Layout section).

6. Set the following properties on the vertical container you just added:

Property	Value
Name	conScreen
X	0
Y	0
Height	Parent.Height
Width	Parent.Width
Vertical Overflow	Scroll

7. Make sure conScreen is selected and insert a Container, rename the container conHeader.
8. Set the following properties on conHeader:

Property	Value
Width	Parent.width
Align in container	Set by container
Minimum height	100

9. With conHeader selected, insert a text label, and set the following properties:

Property	Value
Name	lblBanner
Align	Center
Height	Parent.Height
Size	See code below
Text	"A responsive customer input app with a long title"
Width	Parent.Width
X	0
Y	0

10. For the text label size, use the following code:

```
Switch(
   'Input Screen'.Size,
   1,24,
   2,28,
   3,32,
   4,36
)
```

Note: The banner will fill the entire screen, but that will change once we add other objects to the container.

11. Select conScreen in the Tree view and then insert a Horizontal container control. Ensure that this container appears under conHeader in the Tree view; if this is not the case, use the menu control (…) and choose reorder to move it.
12. Set the following properties on the newly added container:

Property	Value
Name	conInputs
Align (vertical)	Center
Gap	10
Wrap	On
Width	Parent.Width
Align in container	Set by container
Minimum Height	450

13. Select conScreen in the Tree view and then insert a Horizontal container control. Ensure that this container appears under conInputs in the Tree view (but is not a child container); if this is not the case, use the menu control (…) and choose reorder to move it.
14. Set the following properties on the newly added container:

Property	Value
Name	conSubmitButton
Justify (horizontal)	Center
Width	Parent.Width
Align in container	Set by container
Flexible height	Off
Height	50

15. With conSubmitButton selected, insert a button control, and set the following properties:

Property	Value
Name	btnSubmit
Text	Submit

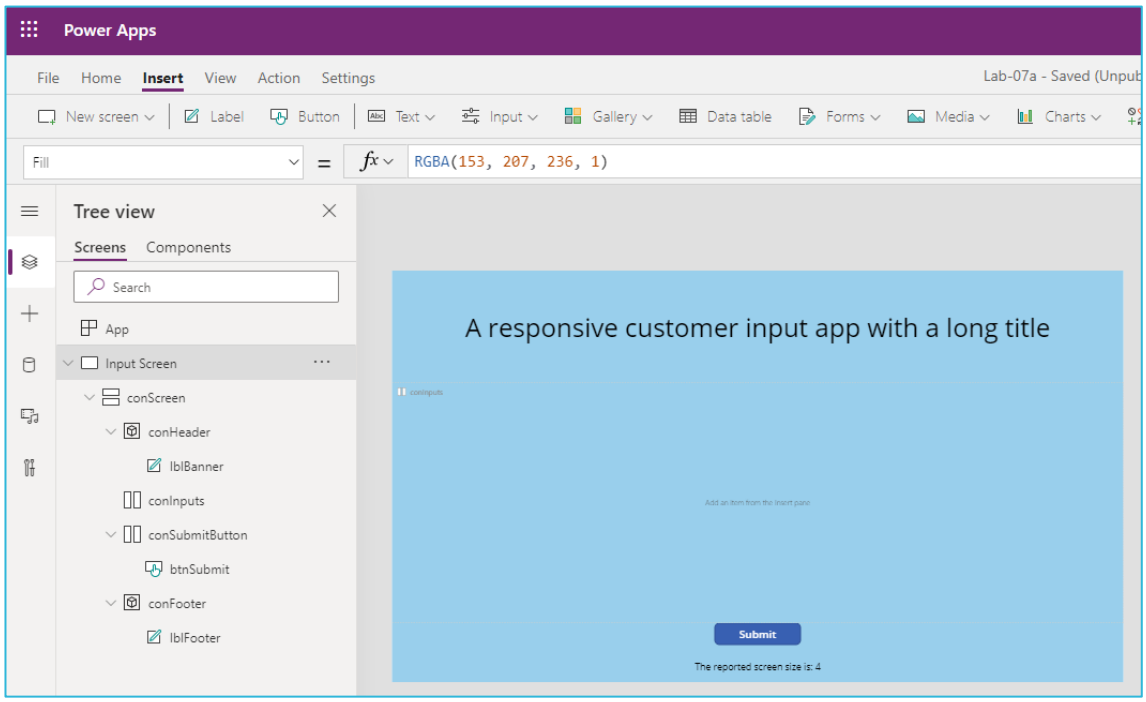
16. Select conScreen in the Tree view and insert a container control.
17. Ensure the newly added container is at the bottom of conScreen.

18. Set the following properties on the newly added container:

Property	Value
Name	conFooter
Width	Parent.Width
Align in container	Set by container
Flexible height	Off
Height	60

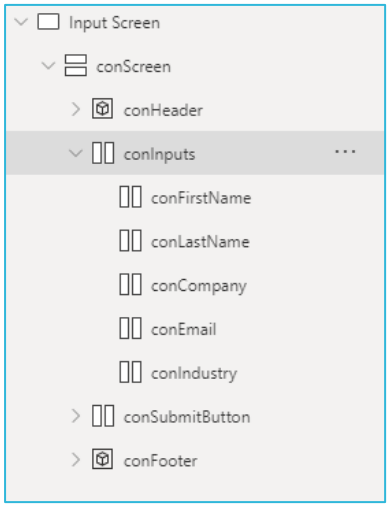
19. Ensure conFooter in selected in the Tree view and insert a label control.
20. Set the following properties on the newly added label:

Property	Value
Name	lblFooter
Text	"The reported screen size is: " & 'Input Screen'.Size
Align	Align.Center
Height	Parent.Height
Width	Parent.Width
X	0
Y	0



Next, we will be adding five horizontal containers to conInputs (one container per property that we want inputted). When adding multiple containers like this, it is important to make sure they all get added at the same level and not nested inside each other. To do this, you need to make sure the parent is re-selected after each is added.

21. Select conInputs in the Tree view, insert a horizontal container and name it conFirstName.
22. Select conInputs in the Tree view, insert a horizontal container and name it conLastName.
23. Select conInputs in the Tree view, insert a horizontal container and name it conCompany.
24. Select conInputs in the Tree view, insert a horizontal container and name it conEmail.
25. Select conInputs in the Tree view, insert a horizontal container and name it conIndustry.

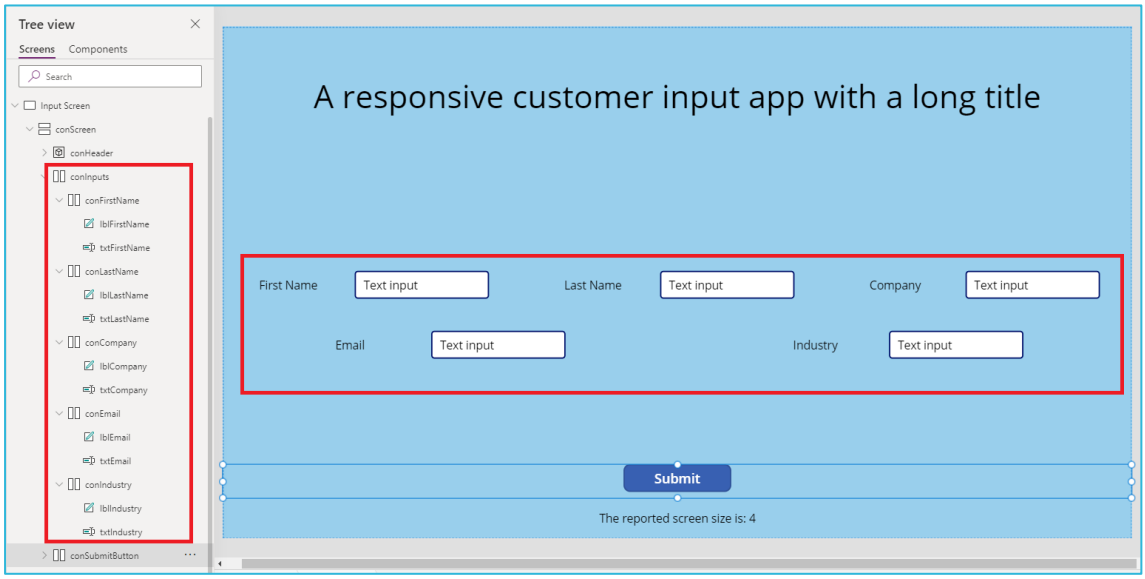


26. On each of the newly added containers, set the following properties.

Property	Value
Justify (horizontal)	Center
Align (vertical)	Center
Align in container	Set by container
Height	80
Minimum width	350

27. Select conFirstName and then insert a text label control and a text input control.
28. Rename the text label to lblFirstName.
29. Rename the text input control to txtFirstName.
30. Set the width of txtFirstName to 200.
31. Set the text property of lblFirstName to "First Name".

32. Repeat steps 27-31 for the remaining four newly added containers, using appropriate names and labels for the controls you add.



33. Save the app, publish it, and then check that it is responsive.

Lab 03 – Advanced data operations

Summary

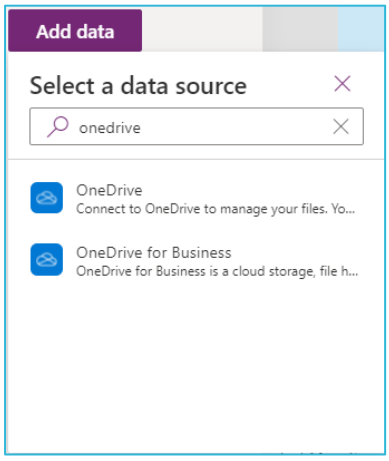
In this lab, we will be concentrating on the functions that allow us to shape data. We will be loading data into a collection from an external date source and then using several different functions to manipulate the data within the app.

Then, we will add a Patch function to allow updates to be made back to the original data source. To ensure that the available time is spent looking at the data functions, a starting point for the app to be produced is provided with most controls placed and configured as far as possible. You will be building on this starting point to complete the app.

This task encourages you to work out the syntax of the code you will need in the app. Should you need it, the code is provided in the Code Answers section of this lab.

Task 1 – Upload data and load app starting point

1. Upload the file C:\Student\Lab03\Equipment.xlsx to your OneDrive library.
2. In your course browser instance, navigate to https://make.powerapps.com.
3. In the Tree view, select Apps.
4. On the menu bar at the top select Import App, and then select From file (.msapp).
5. Browse to C:\Student\Lab03\ Lab 03 - Start.msapp and select open.
6. Using the Save icon, select Save as section enter the name Lab 03 and then select Save.
7. Select the Data pane, and then select Add data.
8. In the Select a data source search box, type OneDrive and then select OneDrive for Business.

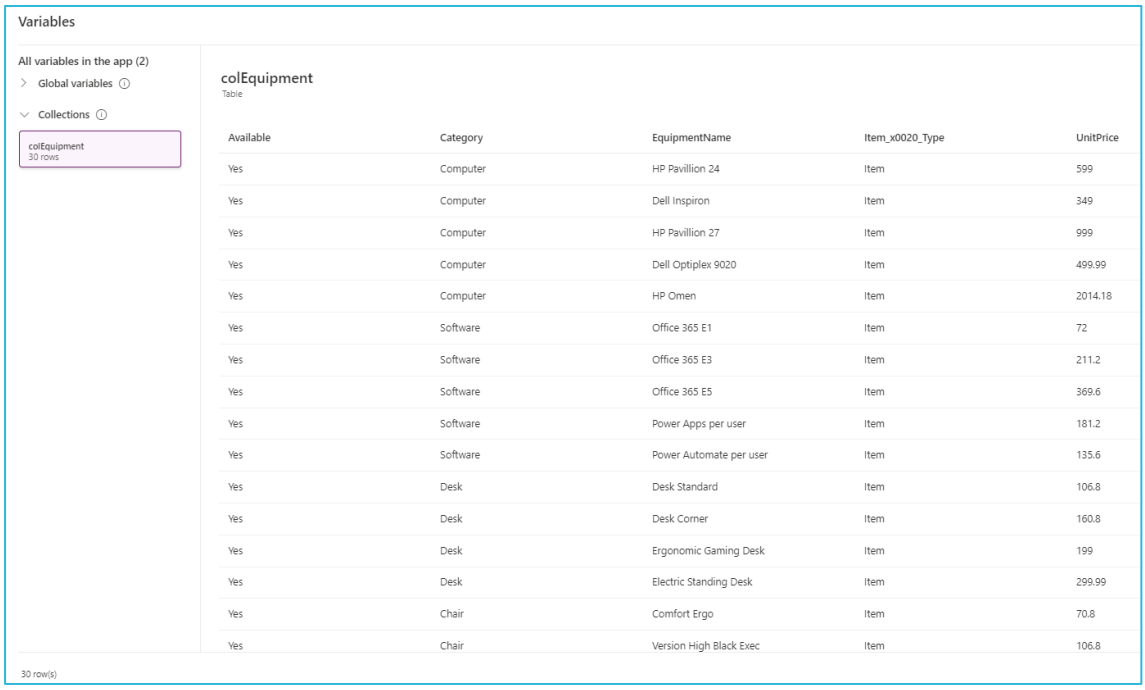


9. If prompted, select Connect to connect to the OneDrive for Business data source.
10. Select the Equipment.xlsx file from your OneDrive library.

11. When prompted, select the EquipmentList table from the file and then select Connect.
12. Save your work.

Task 2 – Create the Collection and add Table

1. Ensure that you can see the Tree view, and then select the App object in the Tree view.
2. Configure the OnStart property to create a collection called colEquipment using the ClearCollect function. Use EquipmentList as the source of this collection, but add a column called "Available" that has the value "Yes".
3. Select the menu (…) beside the App object and choose Run OnStart.
4. In the left navigation pane select Variables and then expand Collections. Using the menu control of colEquipment, select View Table. You should see the values of the collection.

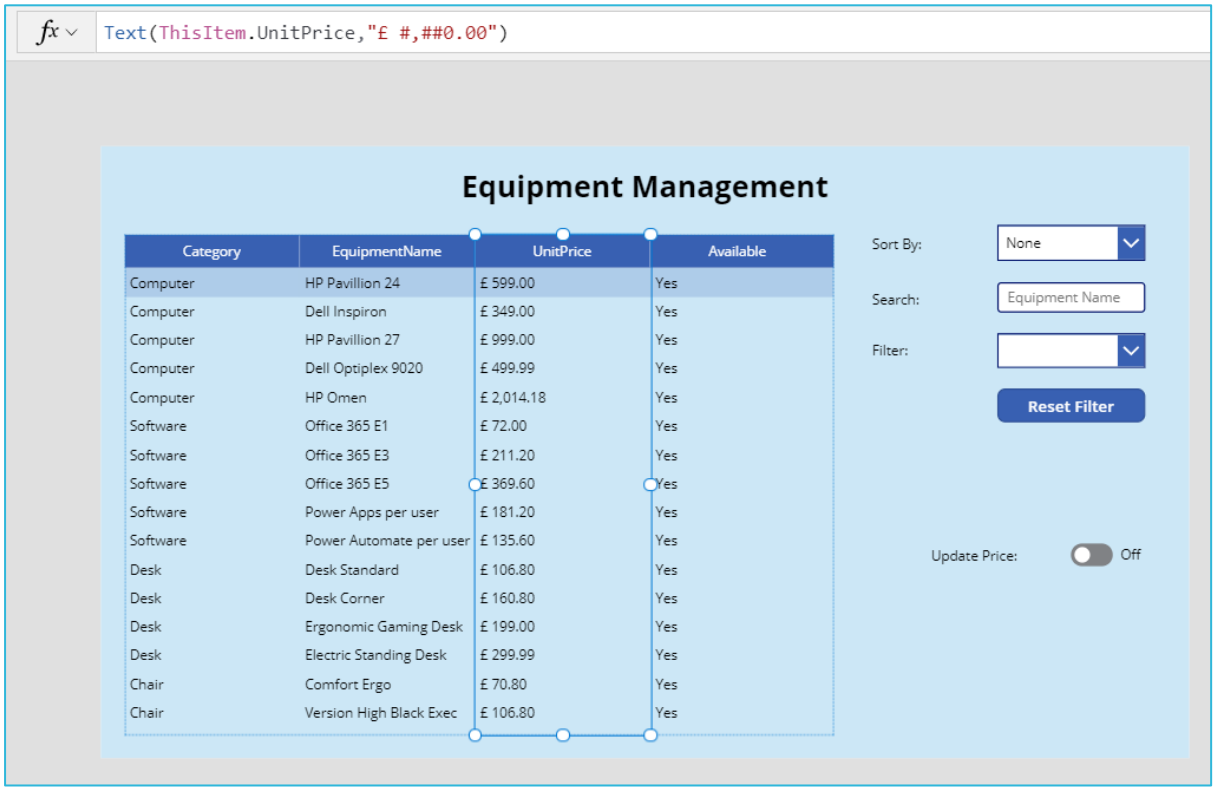


Available	Category	EquipmentName	Item_x0020_Type	UnitPrice
Yes	Computer	HP Pavillion 24	Item	599
Yes	Computer	Dell Inspiron	Item	349
Yes	Computer	HP Pavillion 27	Item	999
Yes	Computer	Dell Optiplex 9020	Item	499.99
Yes	Computer	HP Omen	Item	2014.18
Yes	Software	Office 365 E1	Item	72
Yes	Software	Office 365 E3	Item	211.2
Yes	Software	Office 365 E5	Item	369.6
Yes	Software	Power Apps per user	Item	181.2
Yes	Software	Power Automate per user	Item	135.6
Yes	Desk	Desk Standard	Item	106.8
Yes	Desk	Desk Corner	Item	160.8
Yes	Desk	Ergonomic Gaming Desk	Item	199
Yes	Desk	Electric Standing Desk	Item	299.99
Yes	Chair	Comfort Ergo	Item	70.8
Yes	Chair	Version High Black Exec	Item	106.8

5. Select Cancel to return to the app.
6. Insert a Data Table and use colEquipment as the data source.
7. Rename the table tblEquipment. Position it under the screen label and to the left of the other controls.
8. With tblEquipment selected, select Edit fields in the Properties pane.
9. Select Add field.

10. Select the following fields and then select Add:
    - Category
    - EquipmentName
    - UnitPrice
    - Available
11. In the Tree view, expand tblEquipment and then select the UnitPrice_ColumnX column.
12. In the properties pane, select the advanced tab.
13. Select Unlock to change properties.
14. Edit the Text property to read:

```
Text(ThisItem.UnitPrice,"£ #,##0.00")
```

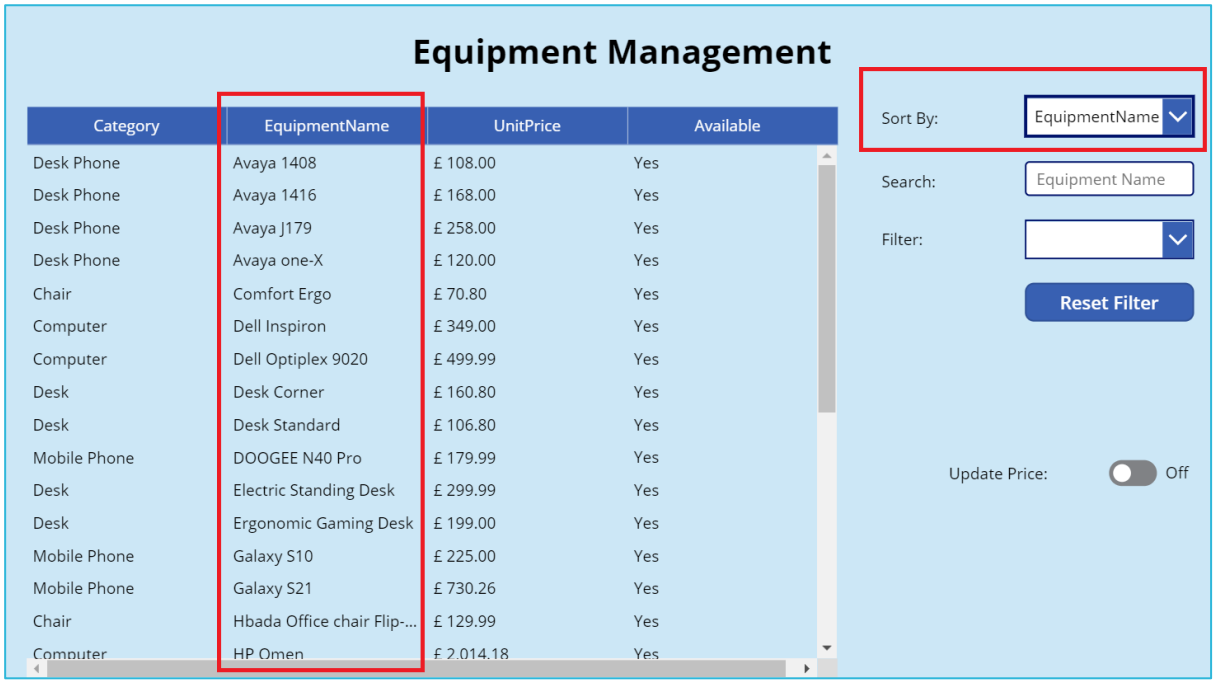


15. Save your work.

Task 3 – Sorting the data in the table

1. Currently, the table displays the equipment in the same order as it is in the Excel spreadsheet. We want to be able to sort this information by Category or by EquipmentName. To allow this, a drop-down control has been added, and it has been populated with the values of "None", "Category" and "EquipmentName".

2. Activating this drop-down control populates a variable, varSort, with the selected value.
3. Using the SortByColumns function we can sort the table using the value from the drop-down control, however, "None" is not the name of a column in the collection so we will need to test for that value in the drop-down control. If "None" is chosen, then we can just use the value colEquiment in the Items property of the table.
4. However, if a value other than "None" is chosen, we want to sort by the Value of the Selected Text.
5. Update the Items property of tblEquipment to sort by the value selected in the drpSortBy drop-down control if that value is not "None". You will need to use the If and SortByColumns functions.
6. Test your app to confirm the sorting works.



7. Save your work.

Task 4 – Adding Search functionality

1. Next, we want to add search functionality to our app and table. The app has a text input control that can be used for this (txtSearch). We want to search against the EquipmentName column.
2. As we previously added an If function to our Items property, the reference to the collection (colEquipment) now appears twice in the code. This means that our search

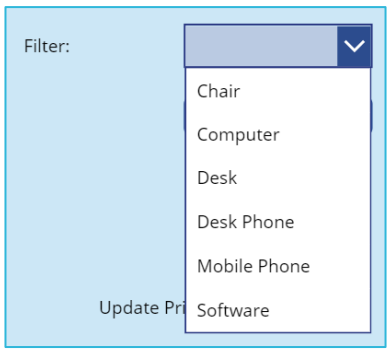
function will also need to appear twice in the code (replacing the current reference to colEquipment).

3. Update the Items property of tblEquipment to search the EquipmentName column for any text typed into txtSearch. You will want to use the Search function for this.
4. Test your work.

5. Save your work.

Task 5 – Adding Filtering functionality

1. We want to be able to filter by the category of equipment. A drop-down control has been added to assist with this. However, its Items property is set to the default value (we had no access to the categories until the data source was added in Task 1).
2. Update the Items property of drpFilter to display the categories from colEquipment. You can use the Distinct function to return each category only once. Sort this list alphabetically.



3. Next, we will need to update the Items property of tblEquipment to reflect the filtering. We can filter simply by using the SelectedText.value property of drpFilter. However,

we also need to not filter if no filter is selected. To cater for this situation, we can use an If function in our code.

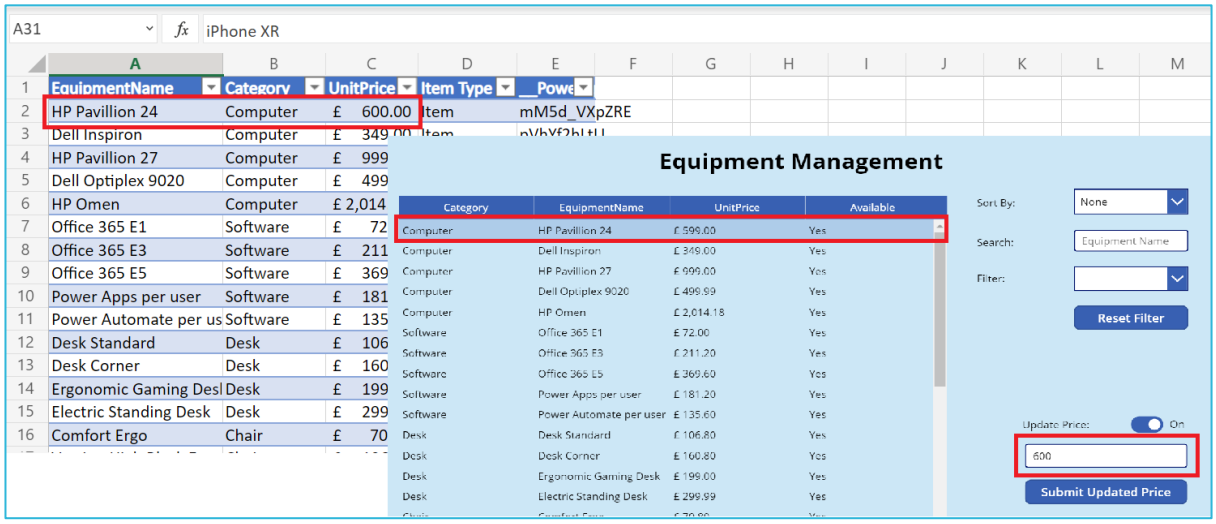
4. In our If function, we can check to see if the value is equal to the result of the Blank() funtion.
5. Update the Items property of tblEquipment replacing each current reference to colEquipment. For each existing occurrence of colEquipment use the Filter function to filter the category by the value displayed in the filter drop-down control. To ensure that all records are returned if no filter is specified, use an Or in the expression and check if the dropdown control is equal to Blank().
6. Test your work.

7. Save your work.

Task 6 – Using the Patch function

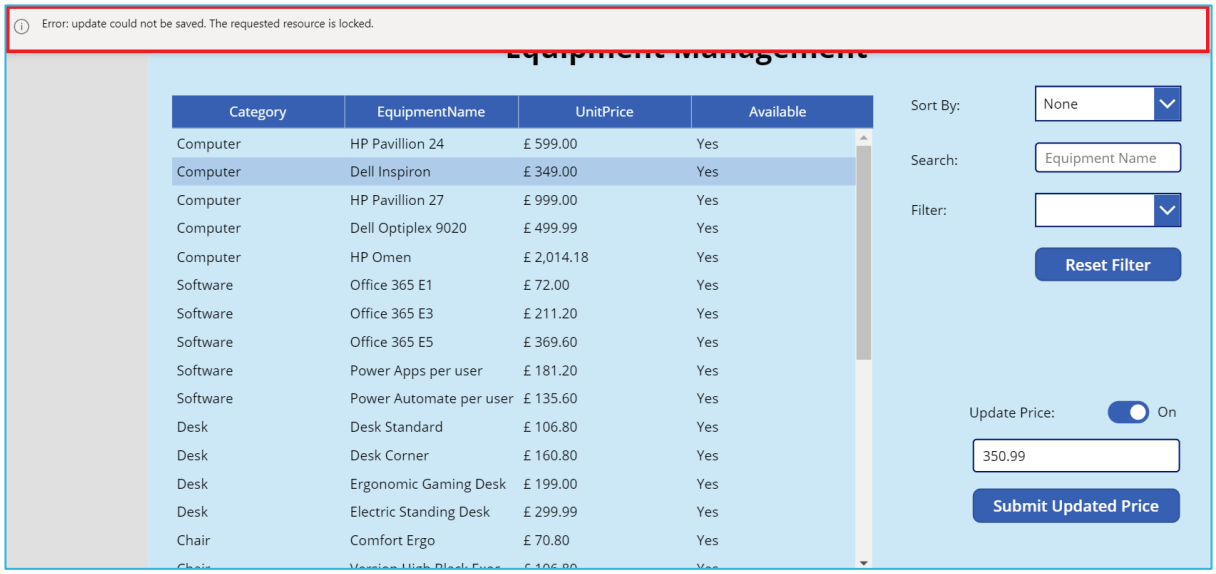
1. Now, we will add functionality to update the price. This update needs to be performed against the original data source (EquipmentList), not the collection we have been using so far. Once the update has been done, the collection (coEquipment) will need to be updated to reflect the change we have just made.
2. Ensure that the Excel spreadsheet we are using as the data source is not open.
3. If you cannot see the text input box and button to update process, then use the Update Price toggle control to make these visible.
4. Update the OnSelect property of btnUpdatedPrice. You want to use a Patch function to update EquipmentList. The property you need to update is called UnitPrice and you will need to use the Value function to convert the text value of txtUpdatedPrice.
5. Often in solutions like this we can refer to the record being patched using the Selected property of the gallery or table that is displaying the records. However, in our case, we cannot do this as the table displays a collection that has been modified from the original data source.

6. To locate the record to be patched, we will need to use the Filter function to find records in EquipmentList where the EquipmentName value is equal to the EquipmentName value of the selected record in tblEquipment. The filter function returns a table of data, we only want a record so we will also need to use the First function to return a single record.
7. Check your code works and that it updates the spreadsheet. Close the spreadsheet once you are done.



8. Save your work.
9. Finally, we need to check the Patch update was done successfully. If this is the case, we need to update the collection and reset the update controls. If the Patch was unsuccessful, then we need to display an error message.
10. Update the OnSelect property of btnUpdatedPrice to check for the success of the Patch function, put this code after the Patch function.
11. Use the Errors function to return any error generated from the Patch function and then use the IsEmpty function to check if there are any errors.
12. If there are no errors, you will need to perform the following steps:
    - Refresh the EquipmentList data source.
    - Rebuild colEquipment using ClearCollect.
    - Reset the toggle control tglUpdatePrice. (This control already has a command to reset the value of txtUpdatedPrice when changed).
13. If there are errors, use the Notify function to display a suitable error message. If you wish, First(Errors(EquipmentList)).Message can be used to display the specific error. NotificationType.Error should be used as the Notification type.
14. Confirm that you can edit a price and that this is the reflected in the app.

15. Open the spreadsheet and attempt another update, confirm that the update fails and that an error message is displayed.



16. Save your work.

Code answers

Task 2 – Step 2

```
ClearCollect(
   colEquipment,
   AddColumns(
      EquipmentList,
      Available,
      "Yes"
   )
)
```

Task 3 – Step 4

```
If(
   drpSortBy.SelectedText.Value = "None",
   colEquipment,
   SortByColumns(
      colEquipment,
      varSort
   )
)
```

Task 4 – Step 3

```
If(
   drpSortBy.SelectedText.Value = "None",
   Search(
      colEquipment,
```

```
        txtSearch.Text,
        EquipmentName
    ),
    SortByColumns(
        Search(
            colEquipment,
            txtSearch.Text,
            EquipmentName
        ),
        varSort
    )
)
```

Task 5 – Step 2

```
Sort(
    Distinct(
        colEquipment,
        Category
    ),
    Value
)
```

Task 5 – Step 6

```
If(
    drpSortBy.SelectedText.Value = "None",
    Search(
        Filter(
            colEquipment,
            drpFilter.SelectedText.Value = Blank() Or Category = drpFilter.SelectedText.Value
        ),
        txtSearch.Text,
        EquipmentName
    ),
    SortByColumns(
        Search(
            Filter(
                colEquipment,
                drpFilter.SelectedText.Value = Blank() Or Category = drpFilter.SelectedText.Value
            ),
        txtSearch.Text,
        EquipmentName
        ),
        varSort
    )
)
```

Task 6 – Step 6

```
Patch(
   EquipmentList,
   First(
      Filter(
         EquipmentList,
         EquipmentName = tblEquipment.Selected.EquipmentName
      )
   ),
   {UnitPrice: Value(txtUpdatedPrice.Text)}
)
```

Task 6 – Step 13

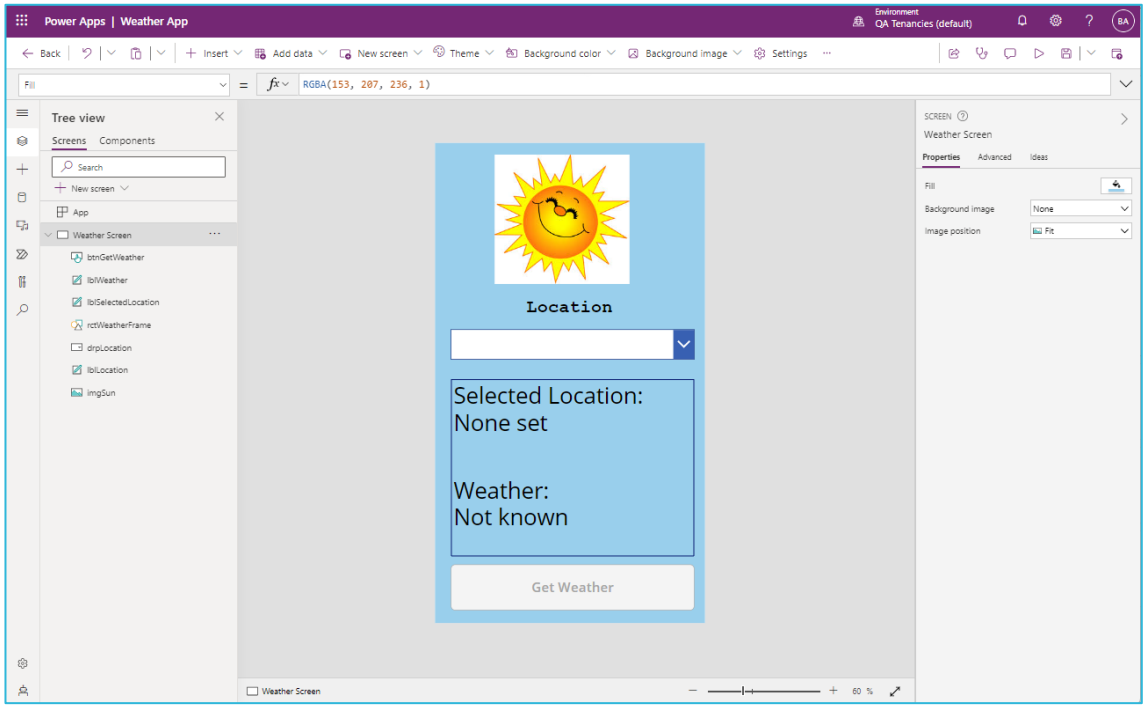
```
Patch(
   EquipmentList,
   First(
      Filter(
         EquipmentList,
         EquipmentName = tblEquipment.Selected.EquipmentName
      )
   ),
   {UnitPrice: Value(txtUpdatedPrice.Text)}
);
If(
   IsEmpty(Errors(EquipmentList)),
   Refresh(EquipmentList);
   ClearCollect(
      colEquipment,
      AddColumns(
         EquipmentList,
         Available,
         "Yes"
      )
   );
   Reset(tglUpdatePrice),
   Notify("Error: update could not be saved. " & First(Errors(EquipmentList)).Message,
   NotificationType.Error)
)
```

Lab 05 – Working with Power Automate

Task 1 – Import a Power App

In this task, you will import a partially configured Power Apps ready to be integrated with Power Automate. This app uses two variables: varLocation and varWeather. varLocation stores the location we wish to get the weather for and is passed to the Power Automate flow as a parameter. varWeather stores the weather for that location and is returned from the Power Automate flow.

1. Open Power Apps and select Apps from the Tree view.
2. From the top menu, select Import app and then select From file (.msapp).
3. Select Browse and then select the Weather App.msapp file in the Lab05 folder of the student files. Select Open.
4. The app should now load into the Power Apps studio.

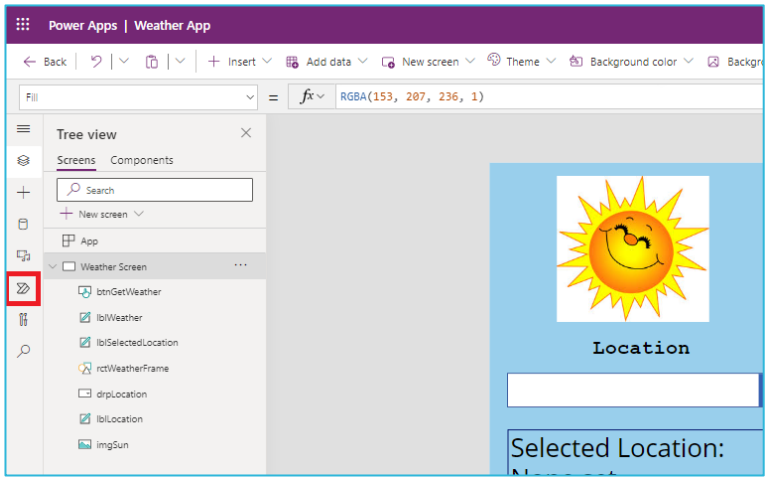


5. Save the app.

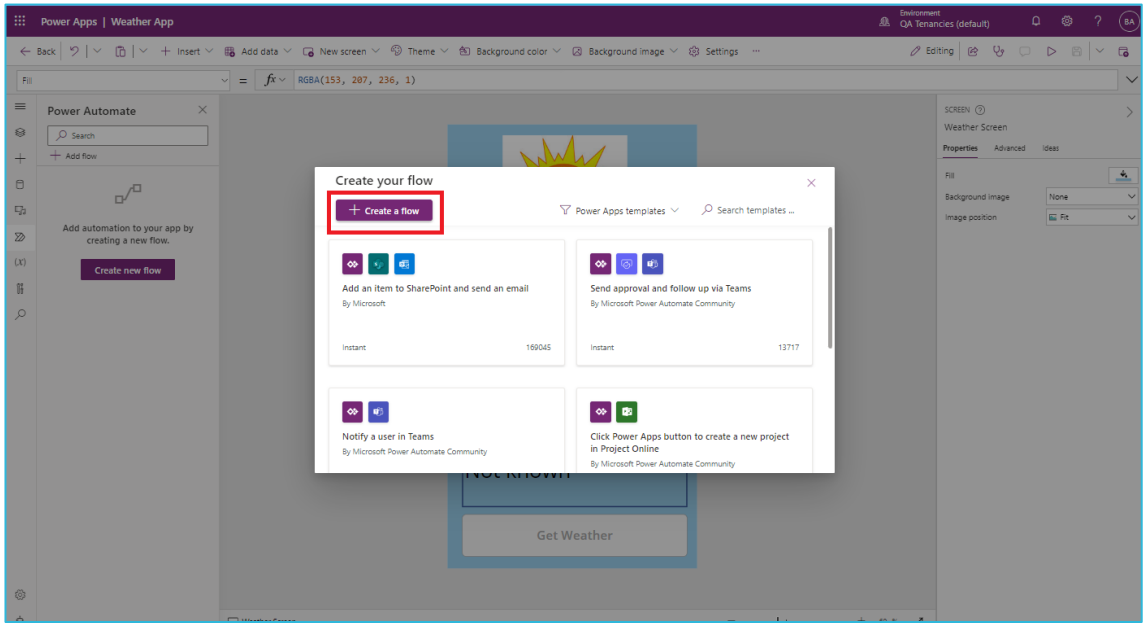
Task 2 – Integrate Power Automate

In this task, we will associate a Power Automate flow (yet to be created) with the Get Weather button. Power Apps controls such as buttons have an OnSelect property which is where you can specify what will happen when the control is pressed. You will associate a Power Automate flow with the OnSelect property.

1. Using the Navigation pane (to the left of the Tree view) select the Power Automate icon.

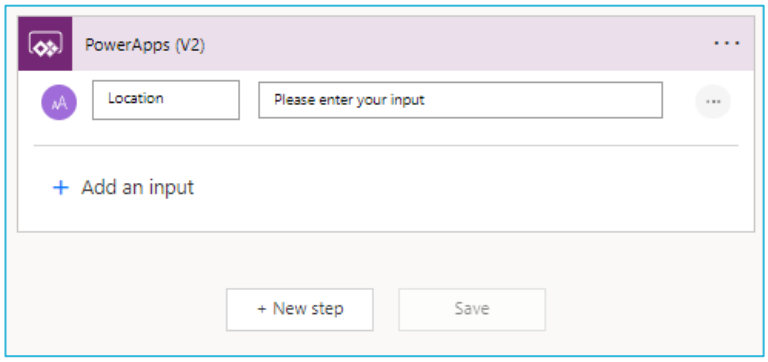


2. The Power Automate pane will open. Select Create new flow.
3. Select Create a flow.

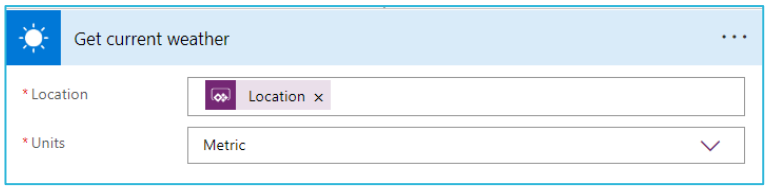


4. In the top-left corner, select Untitled and rename the flow to Power Apps Weather Flow.
5. Click on the purple bar of the trigger named Power Apps (V2) to expand the trigger.

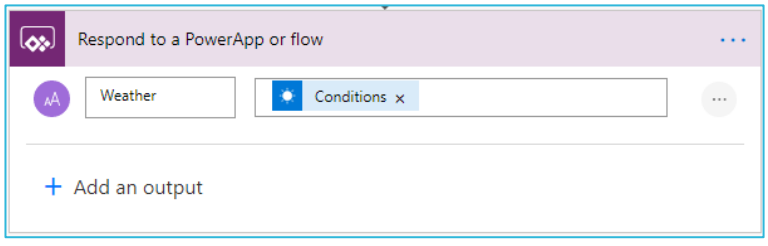
6. Click Add an input. Select Text and give it the name of Location.



7. Click New step and choose the Get current weather action from the MSN weather connector.
8. Click in the Location field of the Get current weather action and select Location from the dynamic content picker.
9. Select Metric as the Units.



10. Click New step and then select a Respond to a PowerApp or flow action from the PowerApps connector.
11. In the Respond to a PowerApp or flow action click Add an output. Choose Text.
12. Give the output the name Weather and then add the Conditions dynamic content from the Get Current weather action.



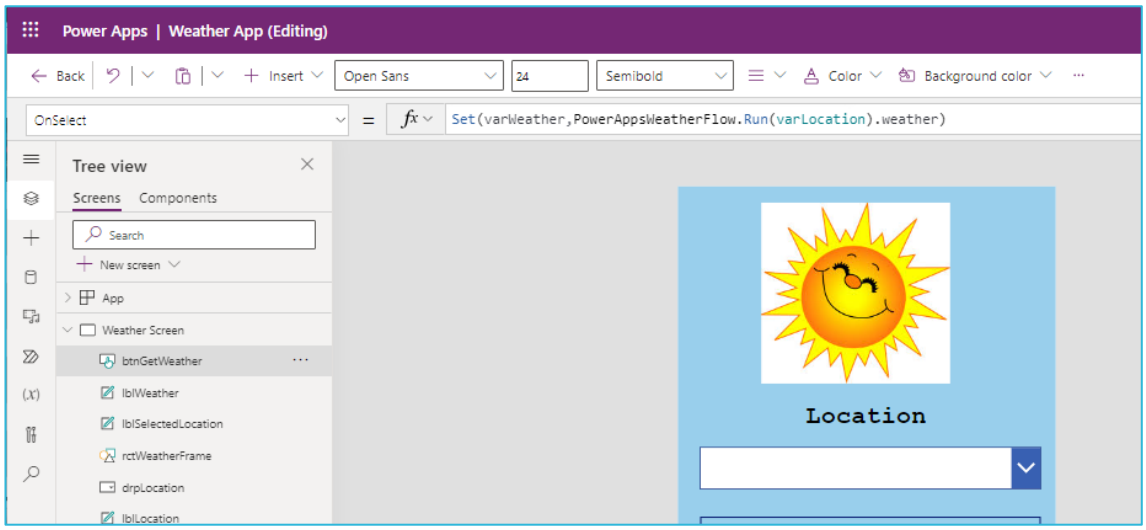
13. Save the flow.

Task 3 – Configure the app to use the flow

1. You will be returned to the app in Power Apps studio. You should now see the flow listed in the Power Automate pane.

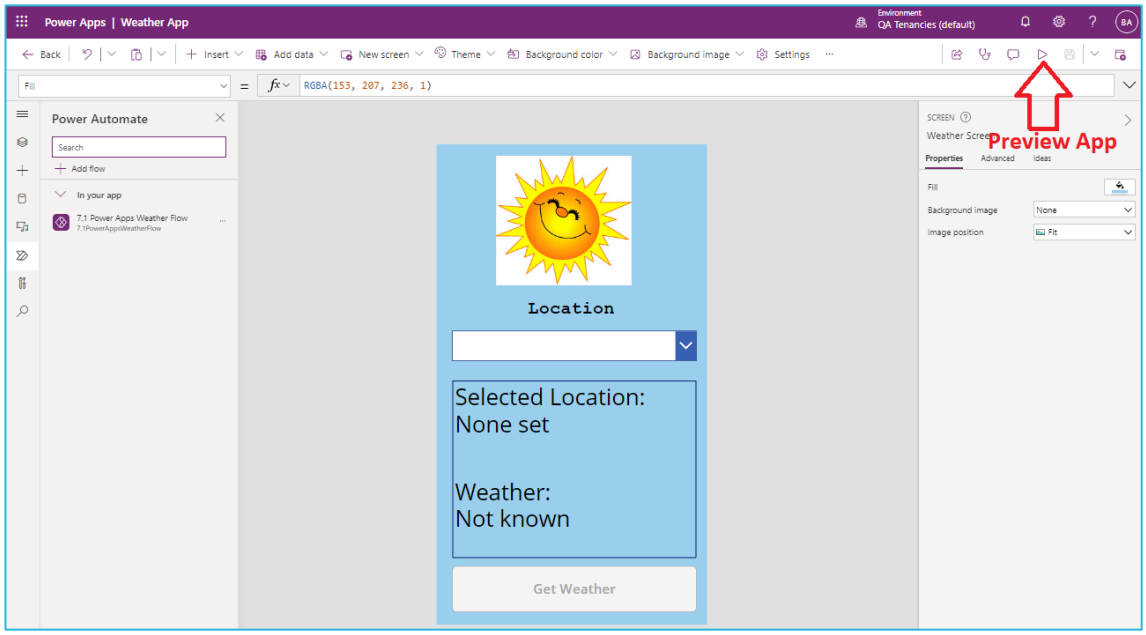
2. Select the Get Weather button and then set its OnSelect property to:

```
Set(varWeather,PowerAppsWeatherFlow.Run(varLocation).weather)
```



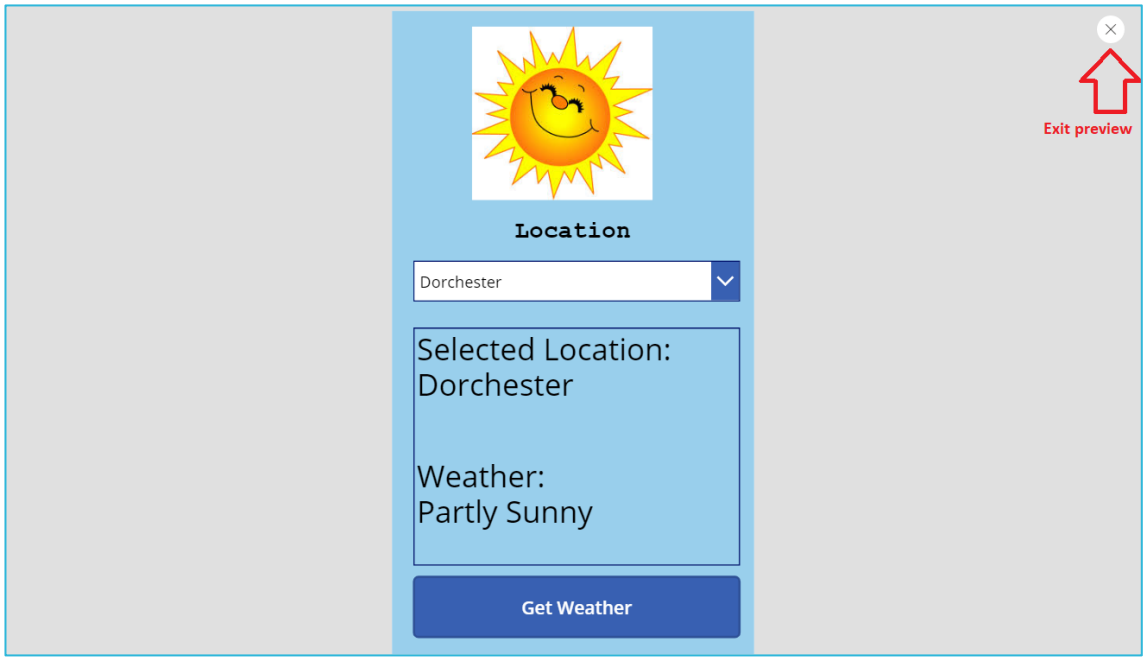
3. Save the App.

4. Use the preview app button to test the app.



5. Pick a location and then click Get weather.

6. After a short delay, the weather for that location should be displayed. Press the X in the top right to exit the app preview.



7. Switch to Power Automate and check the flow history to confirm the flow ran successfully.

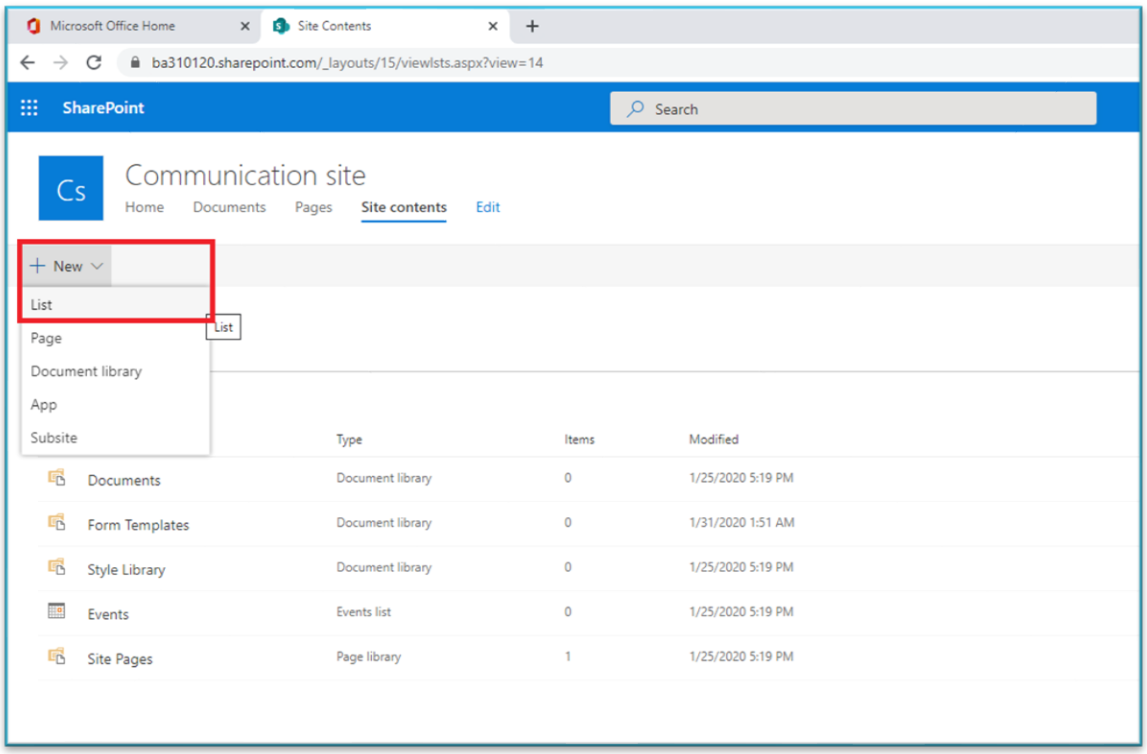
Lab 06a – Working with SharePoint lists

Summary

In this lab, we will be using SharePoint lists as a data source and looking at the techniques of patching complex fields and linked tables using automatically generated IDs.

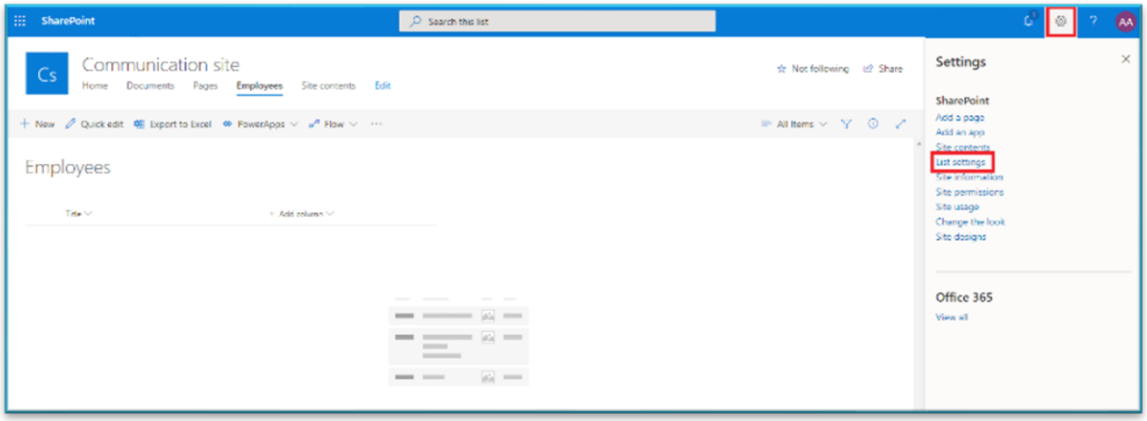
Task 1 – Setting up SharePoint lists

1. Using the Chrome instance configured for this course, navigate to the Communication site in SharePoint.
2. Select the link for Site contents.
3. Select New and then select List from the drop-down.

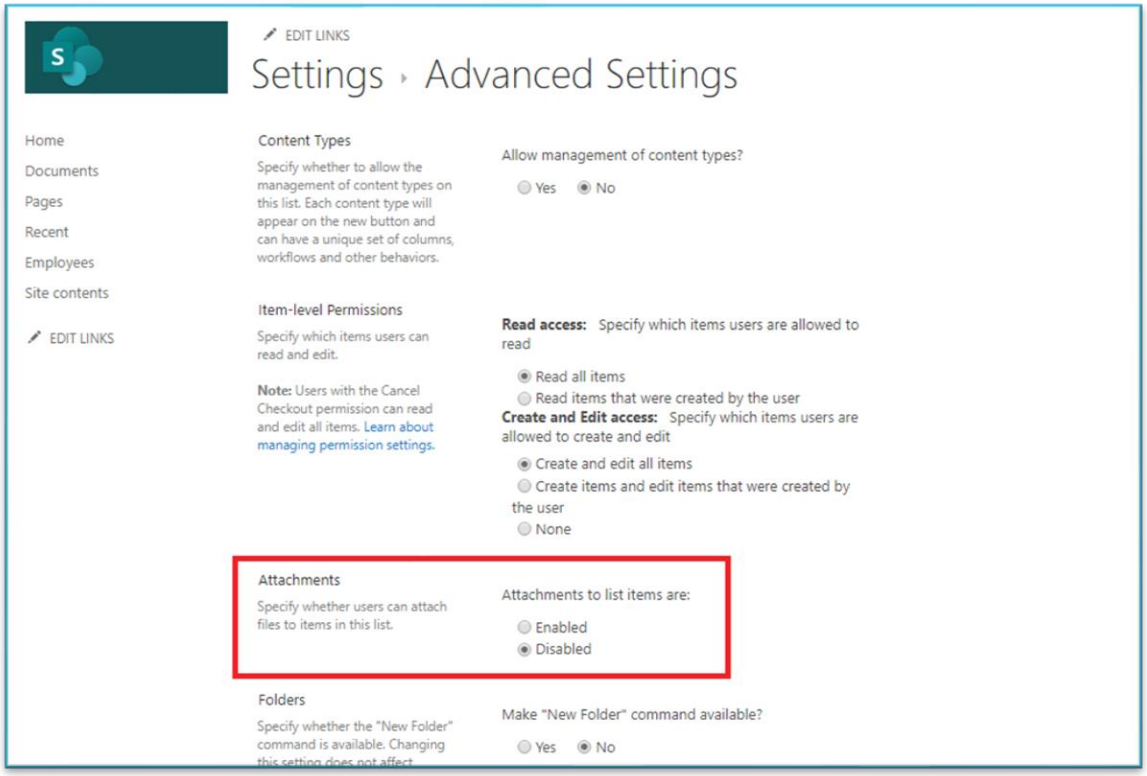


4. Select Blank list.
5. Give the list the name Employees, and then select Create.
6. If you are presented with a welcome wizard, close it.

7. In the top-right, select the settings cog and then choose List Settings.



8. Select Advanced settings, and then set Attachments to Disabled. Scroll down and select OK.

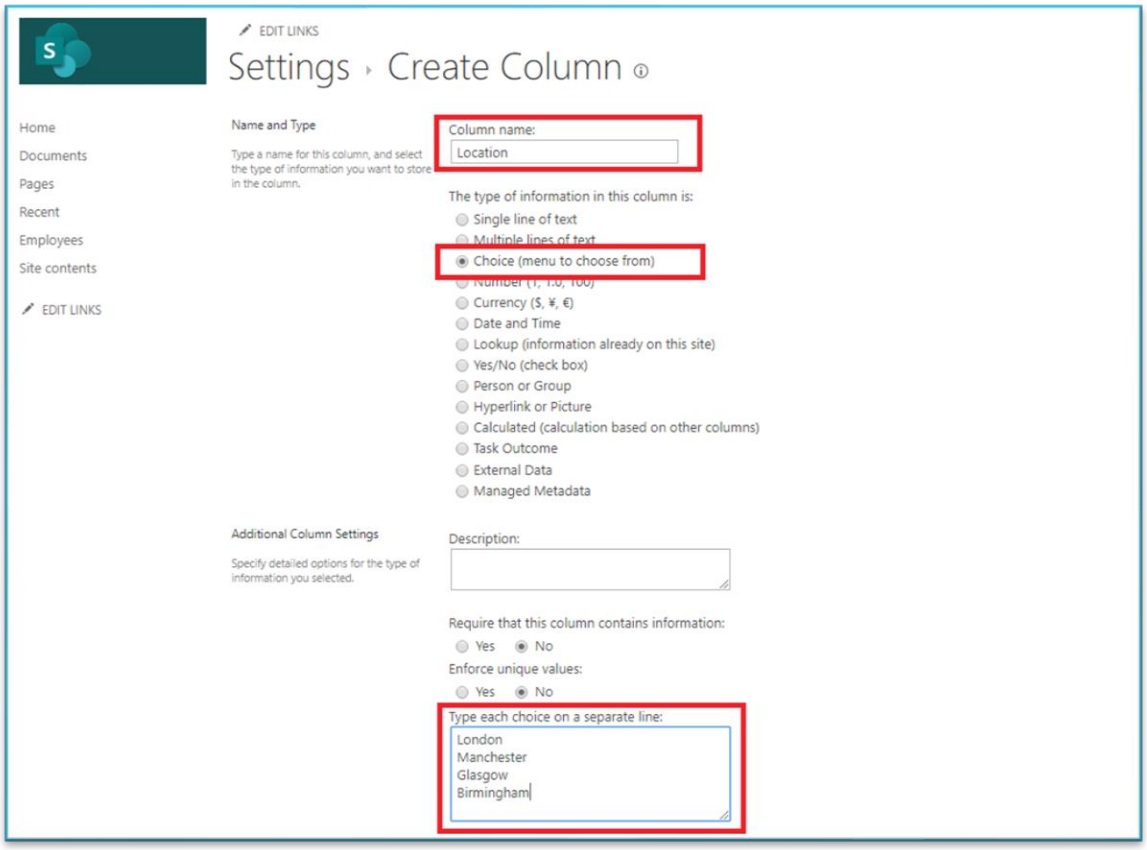


9. In the warning box, select OK.
10. Back on the list settings page in the Columns section, select the Title column to edit it.
11. Change the Column Name to EmployeeID and then select OK.
12. Just below the line in the columns section, select Add from existing site columns.
13. Use the Select site columns from drop-down to choose Core Contact and Calendar Columns.

14. Select and add the following site columns and then select OK:
    - Full Name
    - Department
    - Job Title
15. Select Create column.

Note: We could add location from site columns too, but to show the Power Apps syntax for complex fields, we will manually create the Location property as a choice type.

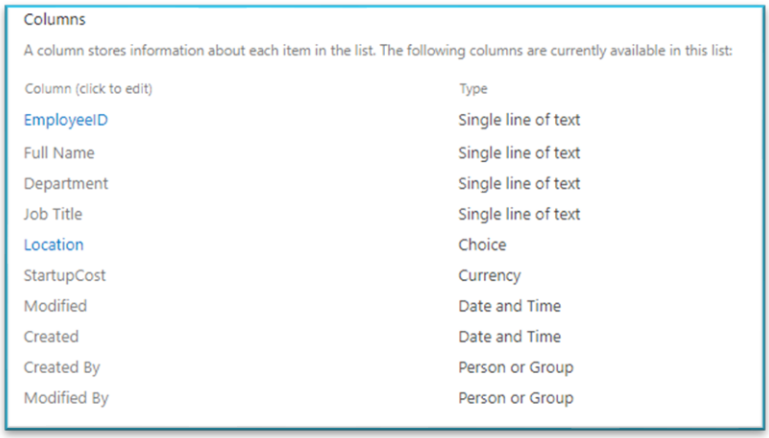
16. Create a column with the column name of Location. Set the type of information to be Choice and the allowed choices to be London, Manchester, Glasgow, and Birmingham.



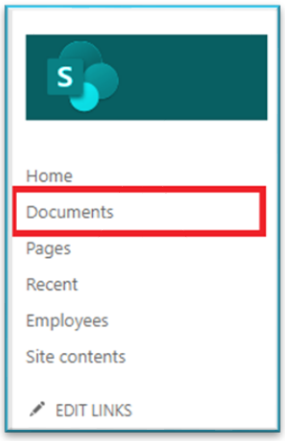
17. Select OK.
18. Select Create column, and create a new column with the following properties:

Property	Value
Column name	StartUpCost
Type	Currency
Currency format	£123,456.00 (United Kingdom)

19. You should now have the following columns:

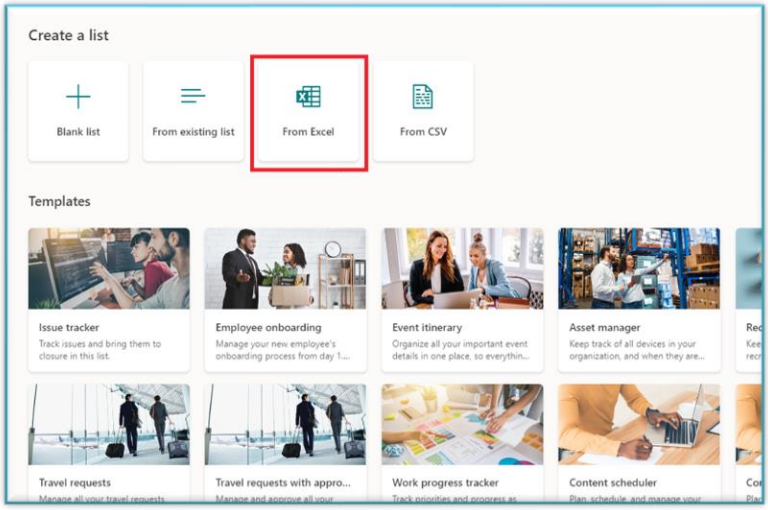


20. Using the side navigation bar, select Documents to navigate to the default document library.



21. Select Upload, and then select Files.
22. Browse to and then select C:\Student\Lab06\Equipment.xlsx.
23. Navigate to Site Contents.
24. Select New, and then choose List.

25. Select From Excel.



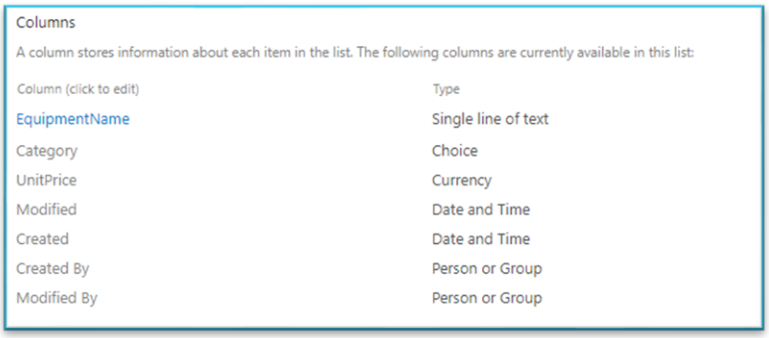
26. Select the file Equipment.xlsx and then select Next.
27. In the Customize dialog, using the drop down make the following changes:

Property	New Type of Data
Category	Choice
UnitPrice	Currency
Item Type	Do not import

28. Select Next.
29. Accept the name Equipment and then select Create.
30. After a few seconds, the list will be created complete with data, however, we need to make a few changes to the column definitions. Select the Settings cog and select List settings.
31. Repeat steps 6-8 to disable attachments.
32. If necessary, rename the Title column to EquipmentName. (Microsoft are in the process of rolling out changes to the process or creating a list from an Excel file so the first column may already be named EquipmentName).
33. Select on the Category column to edit it.
34. Scroll down and enter the following values. (Type each choice on a separate line.)
    - All
    - Computer
    - Software
    - Desk
    - Chair
    - Desk Phone

- Mobile Phone

35. Select Save to accept the changes to this column.
36. In List settings, select the UnitPrice column to edit it.
37. Change the Currency format to £123,456.00 (United Kingdom).
38. Select Save to accept the changes to this column.



39. Select Site contents and then create a New list, choose Blank list and call it NewStarterEquipment.
40. Repeat steps 6-8 to disable attachments.
41. Rename the Title column to NewStarterID.
42. Create a new column with the following settings:

Property	Value
Column name	Item
Type	Lookup (information already on this site)
Get information from	Equipment
In this column	EquipmentName

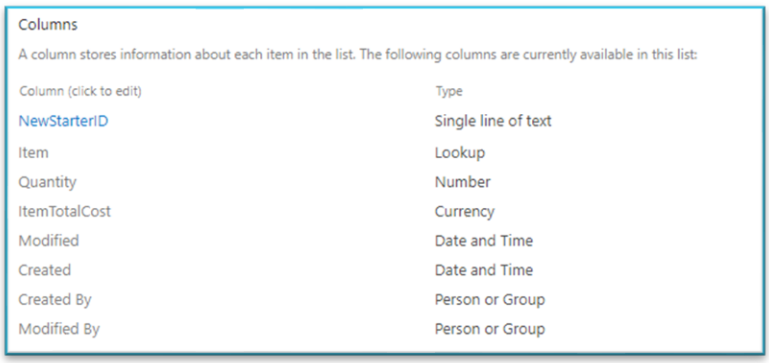
43. Create a new column with the following settings:

Property	Value
Column name	Quantity
Type	Number

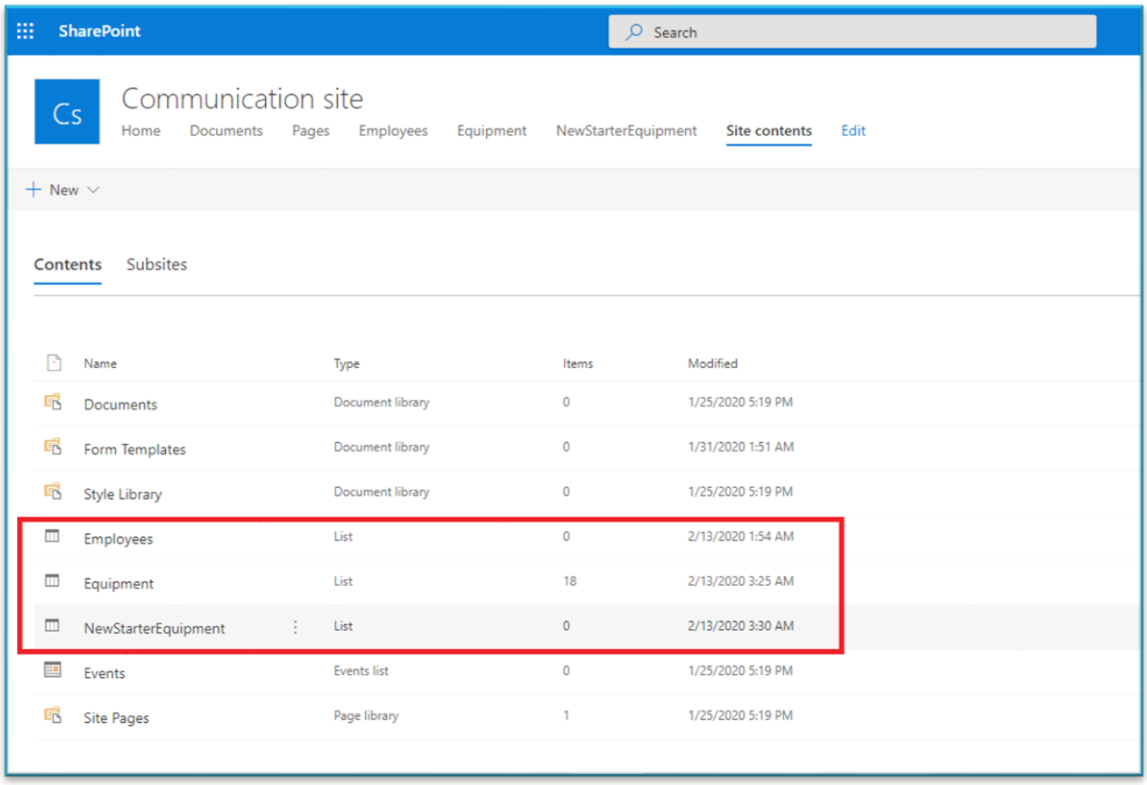
44. Create a new column with the following settings:

Property	Value
Column name	ItemTotalCost
Type	Currency
Currency format	£123,456.00 (United Kingdom)

45. The list columns should now look like the below:



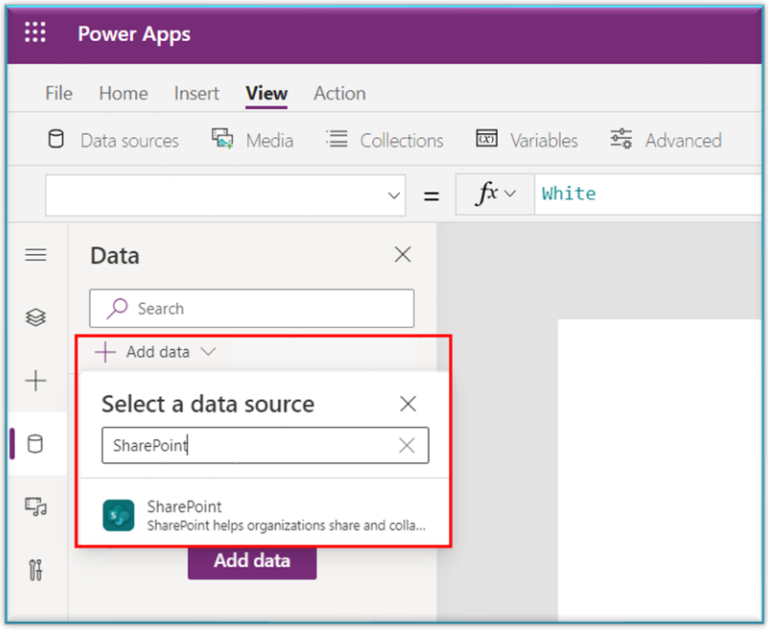
46. Return to Site contents and confirm you can see the three new lists.



Task 2 – Import app and configure existing controls

1. Navigate to https://make.powerapps.com and select Apps in the navigation pane.
2. Select Import app, and then select From file (.msapp).
3. Select Browse.
4. Find and select the file Lab06a.msapp in the Lab06 folder of the student files and then select Open.
5. Use Save As to Save the app with the name Lab-06a.
6. Select the Data pane.

7. Select Add data.
8. In the Select a Data source search box, type SharePoint.



9. Select the SharePoint returned result and then in the details pane, select Connect directly (cloud services) and then select Connect.
10. Select your Communication site and then choose the three lists you created, Employees, Equipment and NewStarterEquipment.
11. Select Connect.
12. Select the Tree view and then select the App object and then configure the OnStart property with the following code.

```
ClearCollect(colEquipment,AddColumns(Equipment,Quantity,1))
```

13. Select the more dots beside the app and then choose Run OnStart.
14. Select the Drop-down control drpLocation, and set the following properties:

Property	Value
Items	Choices(Employees.Location)

15. Select the Drop-down control drpEquipType, and set the following properties:

Property	Value
Items	Choices(Equipment.Category)

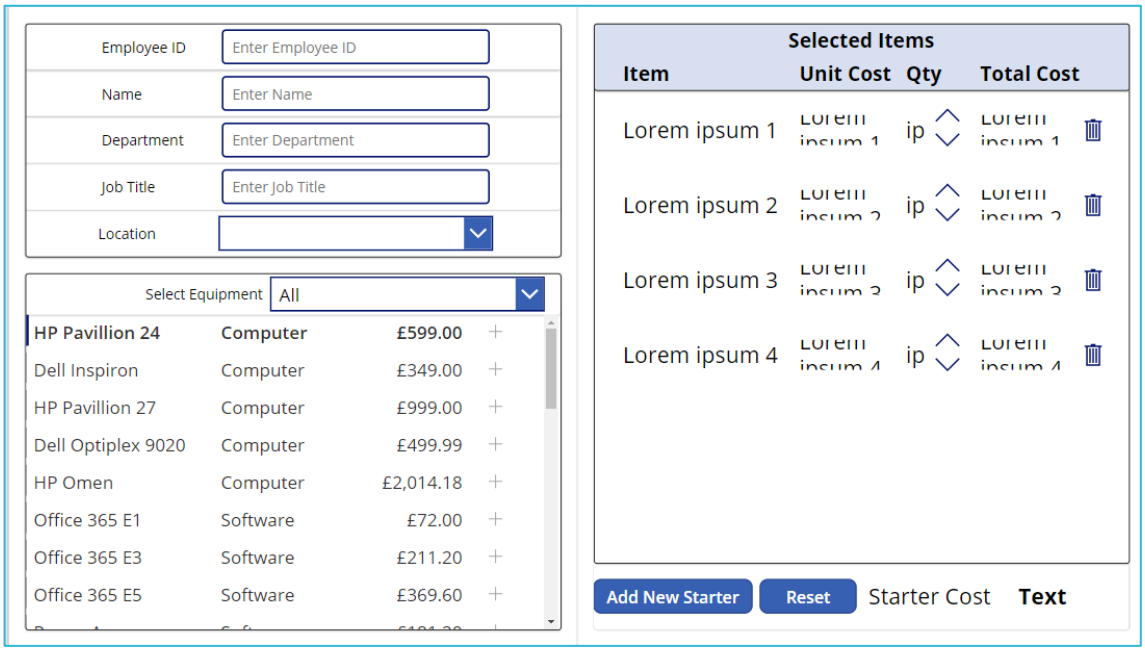
16. Select the gallery galEquipment and configure the Items property of the gallery to the following code:

```
If(
   drpEquipType.Selected.Value = "All",
   colEquipment,
   Filter(
      colEquipment,
      Category.Value = drpEquipType.Selected.Value
   )
)
```

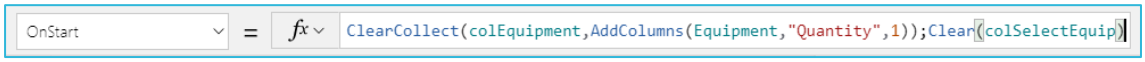
17. Expand the gallery in the Tree view, and then the Text values of the following text labels:

Label Name	Text value
lblEquipmentName	ThisItem.EquipmentName
lblEquipmentType	ThisItem.Category.Value
lblUnitPrice	Text(ThisItem.UnitPrice,"£#,##0.00")

18. Set the OnSelect of icnAdd to Collect(colSelectEquip,ThisItem).
19. Preview the app and use the + icon in galEquipment to select a couple of items. (This is not necessary, but it means you can see the labels in the other gallery better once you start configuring them.)



20. Edit the App OnStart property and add the following code: ;Clear(colSelectEquip)



21. In the Tree view, expand conRight and then expand conSelectedEquipment.

22. Set the Data Source (Items) property of galSelectedEquip to colSelectEquip.
23. Expand the gallery and set the text properties of the following text labels as shown in the table below.

Label	Text property
lblSelectedName	ThisItem.EquipmentName
lblSelectedPrice	Text(ThisItem.UnitPrice,"£#,##0.00")
lblSelectedQty	ThisItem.Quantity
lblSelectedTotalPrice	Text(ThisItem.UnitPrice*ThisItem.Quantity,"£#,##0.00")

24. Set the OnSelect property of icnRemoveSelected to:

```
Remove(colSelectEquip,ThisItem)
```

25. Set the OnSelect properties of the icnUp and icnDown icons as shown in the table below:

Icon	OnSelect
icnUp	Patch(colSelectEquip,ThisItem,{Quantity:ThisItem.Quantity+1})
icnDown	Patch(colSelectEquip,ThisItem,{Quantity:ThisItem.Quantity-1})

26. Set the DisplayMode property of icnDown to the following code:

```
If(ThisItem.Quantity=1,DisplayMode.Disabled,DisplayMode.Edit)
```

27. Set the Text property of lblStarterCost to:

```
Text(Sum(colSelectEquip,UnitPrice*Quantity),"£#,##0.00")
```

28. Set the OnSelect property of btnReset to the following code:

```
Clear(colSelectEquip);
Reset(txtEmployeeID);
Reset(txtName);
Reset(txtDepartment);
Reset(txtJobTitle);
Reset(drpLocation);
Reset(drpEquipType)
```

29. Set the OnSelect property of btnAddStarter to the following code:

```
ClearCollect(
   colAddedEmpRecord,
   Patch(
      Employees,
      Defaults(Employees),
      {Title: txtEmployeeID.Text},
      {FullName: txtName.Text},
```

```
    {Department: txtDepartment.Text},
    {JobTitle: txtJobTitle.Text},
    {
      StartUpCost: Sum(
        colSelectEquip,
        UnitPrice * Quantity
      )
    },
    {
      Location: {
        '@odata.type': "#Microsoft.Azure.Connectors.SharePoint.SPListExpandedReference",
        Id: 1,
        Value: drpLocation.Selected.Value
      }
    }
  )
);
ForAll(
  colSelectEquip,
  Patch(
    NewStarterEquipment,
    Defaults(NewStarterEquipment),
    {NewStarterID: First(colAddedEmpRecord).ID},
    {
      Item: {
        '@odata.type': "#Microsoft.Azure.Connectors.SharePoint.SPListExpandedReference",
        Id: ID,
        Value: EquipmentName
      }
    },
    {Quantity: Quantity},
    {ItemTotalCost: UnitPrice * Quantity}
  )
);
Clear(colSelectEquip);
Reset(txtEmployeeID);
Reset(txtName);
Reset(txtDepartment);
Reset(txtJobTitle);
Reset(drpLocation);
Reset(drpEquipType)
```

30. Save the app and test it. Check the appropriate values were added to the Employees and NewStarterEquipment lists.

Lab 06b – File upload to a SharePoint library

Summary

In this lab, we will look at the steps required to upload a file to a SharePoint library. We will also use the returned information so we can link to the file from a SharePoint list.

Task 1 – Setting up list and library

1. Navigate to your SharePoint communication site and then go to Site contents.
2. Select New, and then select List.
3. Choose Create from blank List.
4. Give the list the name ExpensesList and then select Create.
5. Once you are returned to the list, select the settings cog and then choose List settings.

Note: Usually at this stage we would go into advanced settings and disable attachments. However, for this lab we need access to a list with attachments enabled to grab the attachment control. Once we have that control, we could disable attachments if required.

6. Using the Create column option, create the following columns:

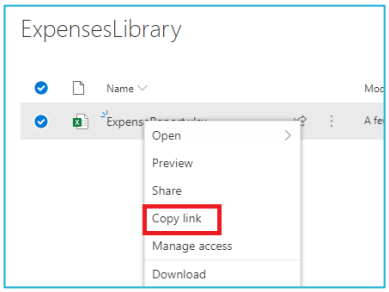
Column Name	Type	Extra Info if required
ExpenseDate	Date and Time	
Category	Choice	Values: Public Transport, Accommodation, Food, Professional Fees, Misc
Customer	Single line of text	
Amount	Currency	Currency Format: United Kingdom
SupportingDocument	Hyperlink or Picture	

7. Return to Site Contents and create a New Document Library.
8. Give the library the name ExpensesLibrary and then select Create.

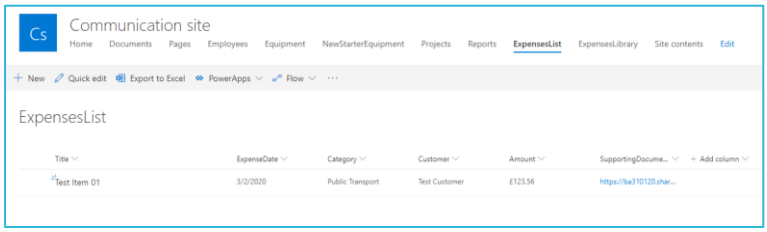
Note: The next steps are necessary only to give us some data to make our life easier while working in Power Apps.

9. While in the ExpensesLibrary, upload a file (any document will do, but you should find some sample documents in C:\Student\Lab06\Office Documents).

10. Once the file is uploaded, right-click the file name and then choose Copy link.

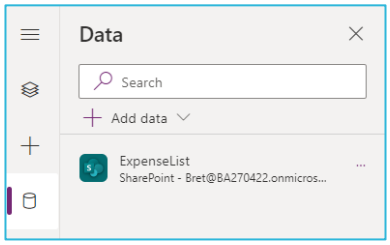


11. Navigate to the ExpensesList list and create a new item. Use the copied URL for the supporting document link.



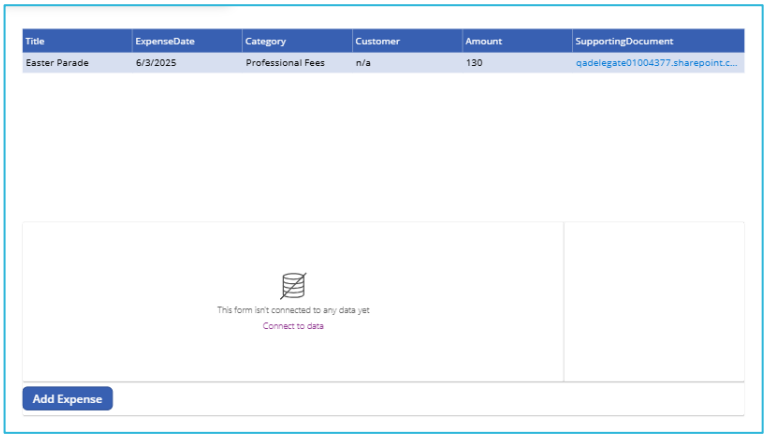
Task 2 – Set up basic controls to view and add items

1. Navigate to Power Apps select the Apps node and then import an app, and then select From file (.msapp).
2. Navigate to and Select Lab-06b.msapp in the Lab06 folder of the student files and then select Open.
3. Save the app as Lab06-b.
4. Add a Data source to the app. Use the SharePoint connector and select your Communication site. Pick the ExpensesList.

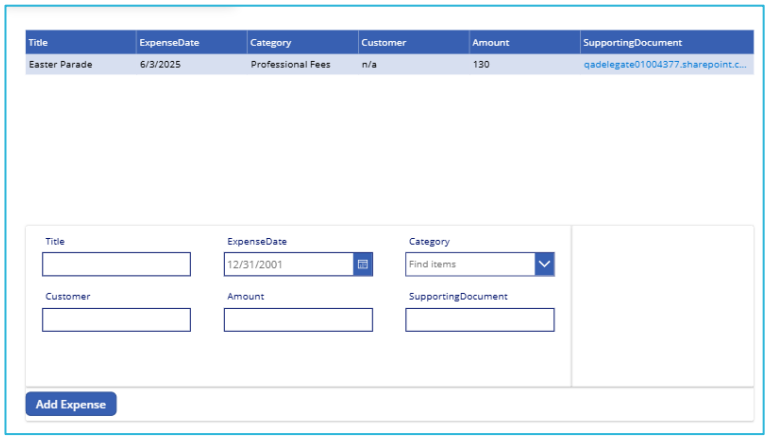


5. In the Tree view, select tblExpenses and set its Data source to be ExpenseList.

6. If the Attachments column is displayed, then remove that column by using the X selected option for the table. Remove any other metadata columns that may have been added.



7. In the Tree view, select frmAddExpense. This can be found under: Expenses Screen > conExpensesScreen > conDataEntry > conForm.
8. Set the Data Source of frmAddExpense to ExpensesList.
9. Using the link beside fields, ensure that the following fields are selected: Title, ExpenseDate, Category, Customer, Amount and SupportingDocument.

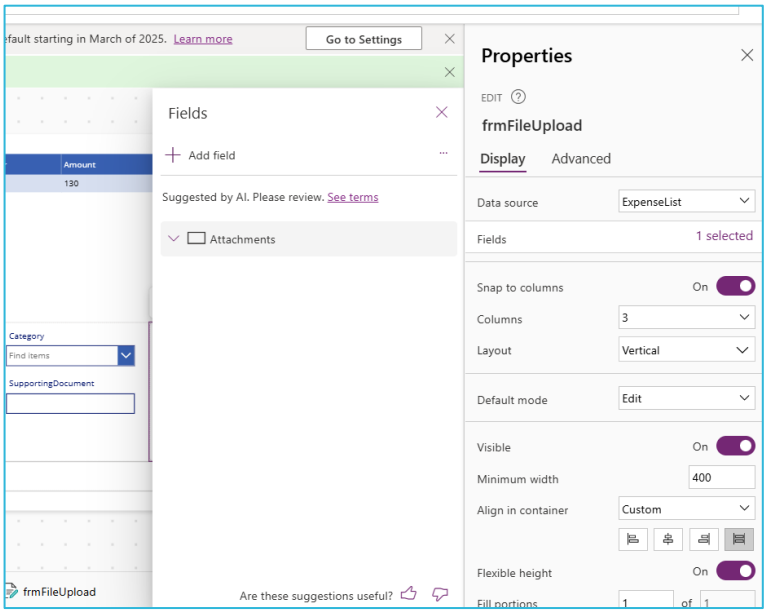


10. Edit App.OnStart to add the following command.

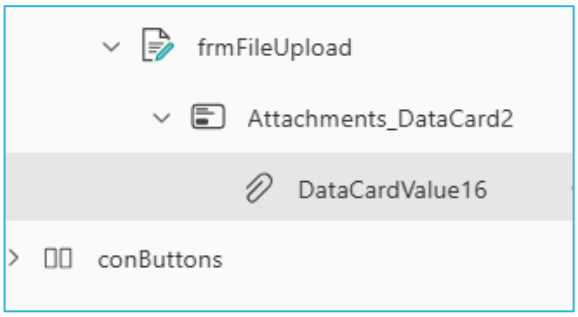
```
Set(varUploadedImage,Blank())
```

11. Run the App.OnStart code to populate the variables.
12. In the tree view, select imgFile and change its image property to varUpLoadedImage.
13. In the tree view, navigate to and then select conUploadControl.
14. Into this container, Insert an Edit Form. Choose ExpenseList as the data source. Rename this form frmFileUpload.
15. With frmFileUpload selected, click Edit fields in the Properties pane.

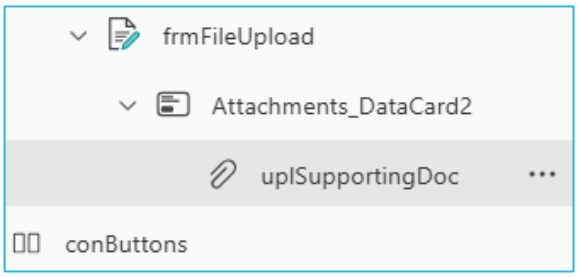
16. In the Fields pane, add the Attachments field and remove all the other fields.



17. Close the fields pane.
18. In the tree view, expand frmFileUpload, select the Attachments data card and then unlock it.
19. In the tree view, expand the Attachments data card and then delete all the controls except the DataCardValue.



20. Rename the DataCardValue to uplSupportingDoc.



21. In the tree view, right-click frmFileUpload and then select Reorder | Move to top.
22. Set the following properties on frmFileUpload

Property	Value
Columns	1
Minimum width	200

Minimum Height	100
Align in container	AlignInContainer.Stretch
OnReset	Set(varUploadedImage, Blank())

23. Set the following properties on the AttachmentsDataCard.

Property	Value
X	0
Y	0
Height	Parent.Height
Width	Parent.Width

24. Set the following properties on uplSupportingDoc

Property	Value
Max attachments	1
X	0
Y	0
Height	Parent.Height
Width	Parent.Width
OnAddFile	Set(varUploadedImage,Last(Self.Attachments).Value)
OnRemoveFile	Set(varUploadedImage, Blank())

25. In the Tree view, expand conButtons (conButtons is inside conExpensesScreen)
26. Select imgFile.
27. Set the Image property to be

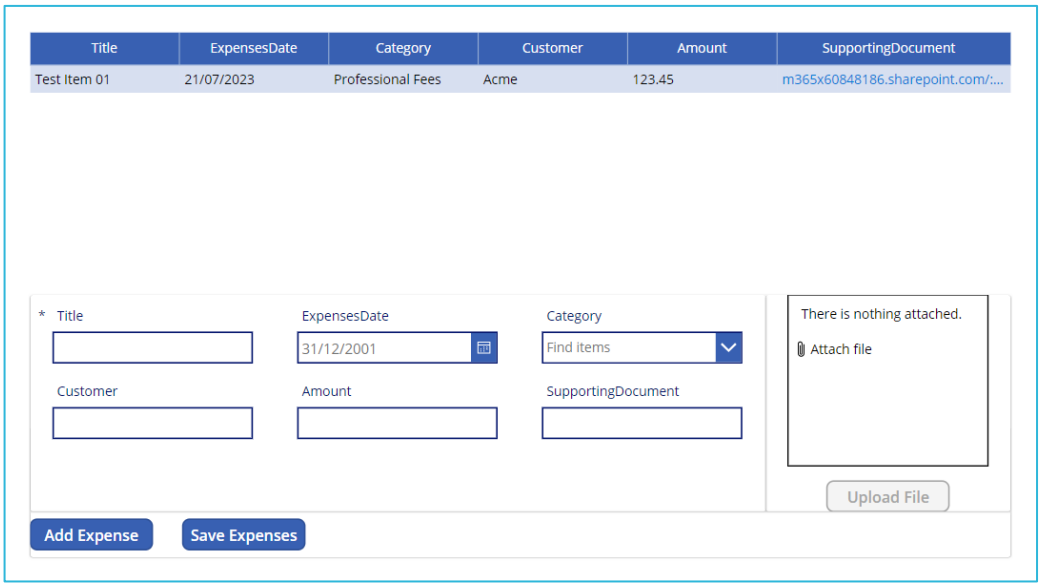
```
varUploadedImage
```

28. Select btnAddExpense and edit the OnSelect property to read.

```
Set(varAddExpense,true);NewForm(frmAddExpense);NewForm(frmFileUpload)
```

29. Press the alt key and then select the "Add Expense" button to run the OnSelect code. (The attachment control, Upload file button and Save Expenses button should become visible).
30. Set the DisplayMode property of btnUploadFile to the following code:

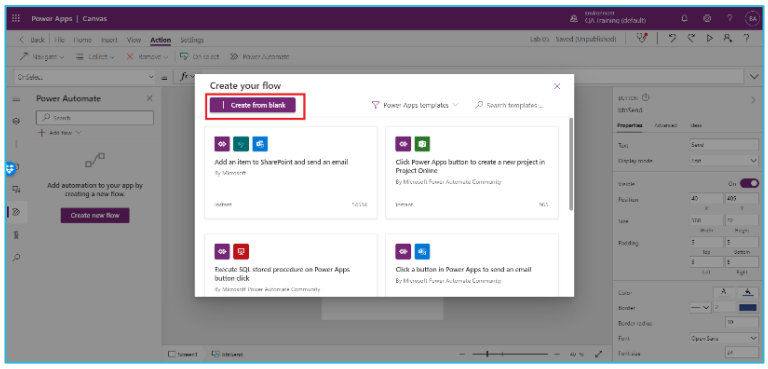
```
If(imgFile.Image=Blank(),DisplayMode.Disabled,DisplayMode.Edit)
```



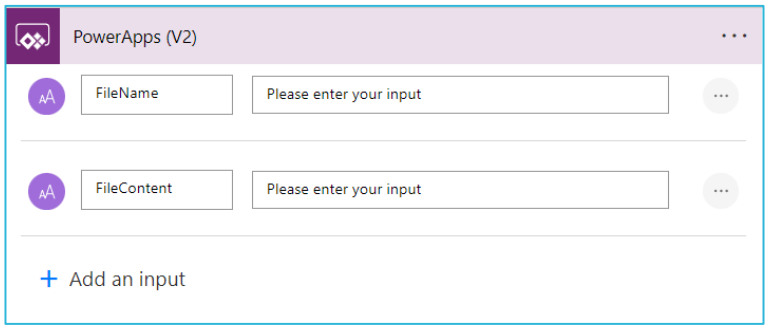
31. Save the app.

Task 3 – Create the Power Automate flow

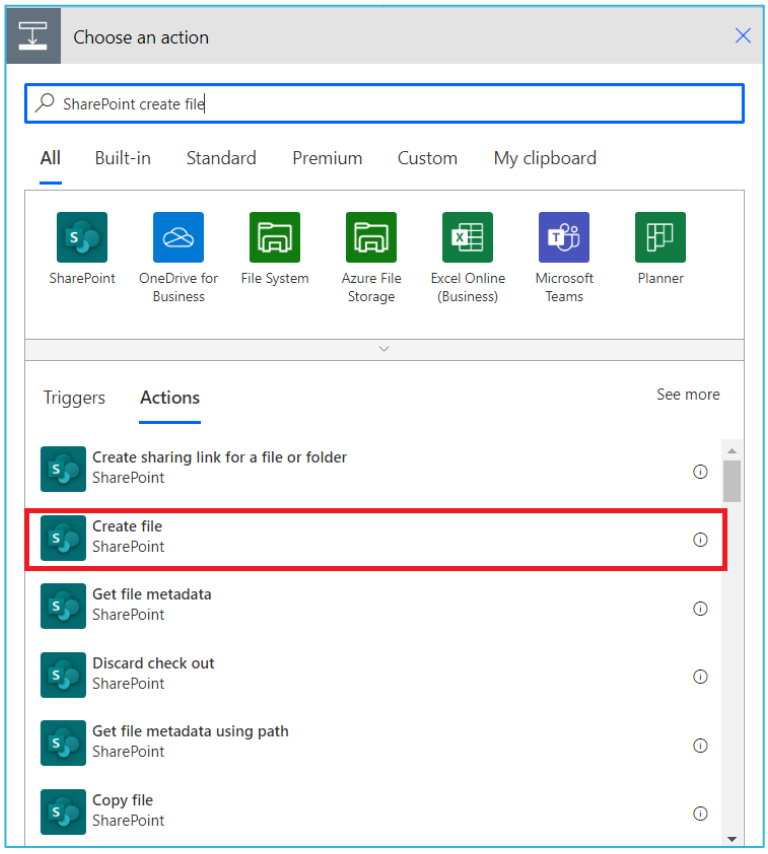
1. Using the Power Automate pane select Create new flow.
2. Select Create from blank.



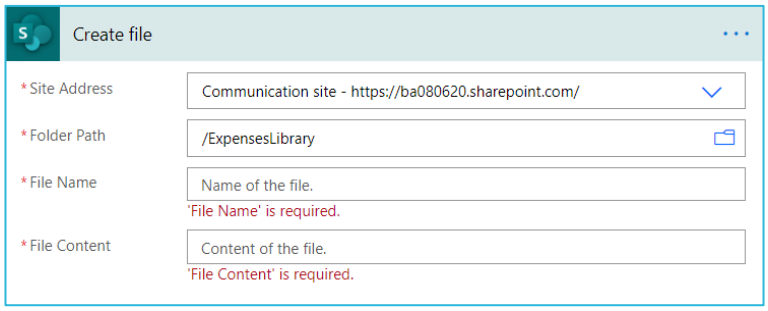
3. Select the current name of the flow (Untitled) and then rename the flow Upload Expense File.
4. Expand the PowerApps (V2) trigger, select Add an input.
5. Select Text.
6. Rename the Input to FileName.
7. In the PowerApps (V2) trigger, select Add an input.
8. Select Text.
9. Rename the Input 1 to FileContent.



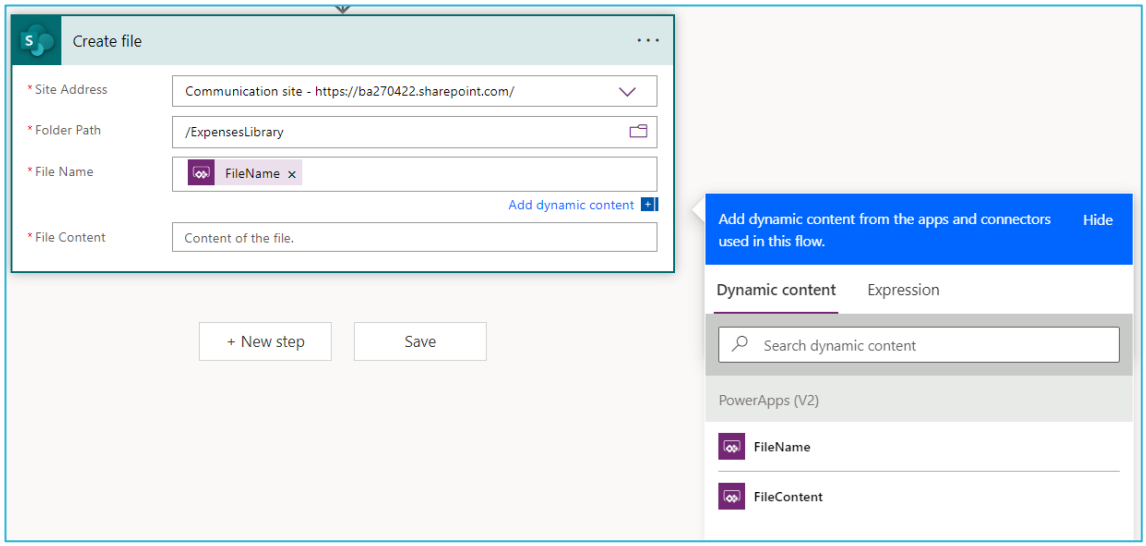
10. Select New step.
11. In the search box type SharePoint create file and then select the Create file action that appears. Ensure that this is a SharePoint action.



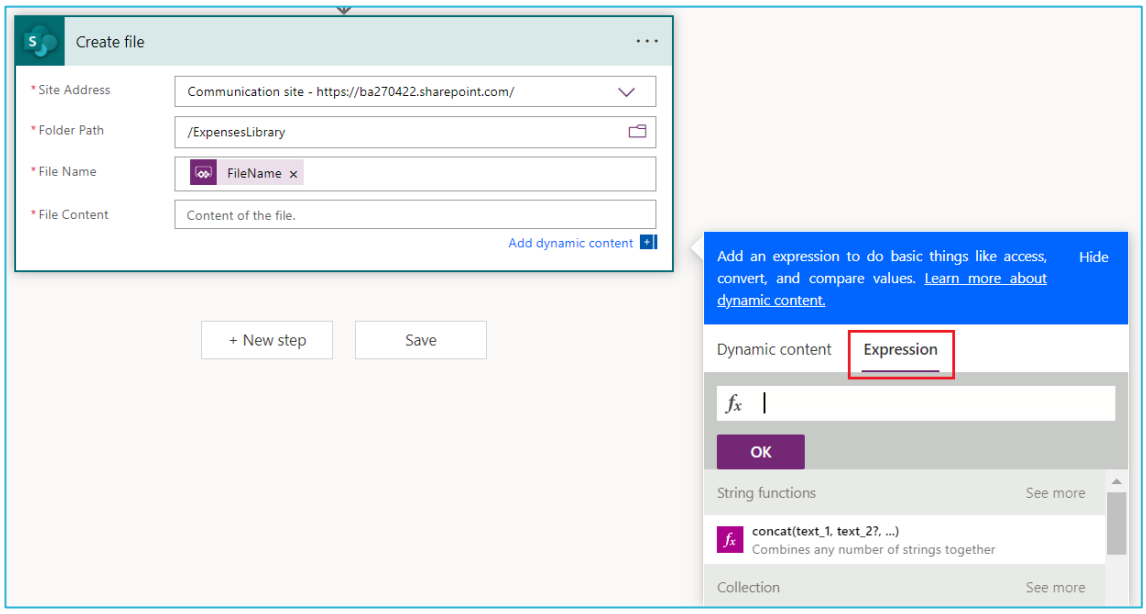
12. The Create file action will be added to the flow. Use the Site Address dropdown control to specify your Communication site.
13. In the Folder Path field, use the folder browse control to select the ExpensesLibrary document library.



14. Select in the File Name field.
15. If the dynamic content box does not appear, then select the Add dynamic content link below this field.
16. Select the FileName dynamic content.



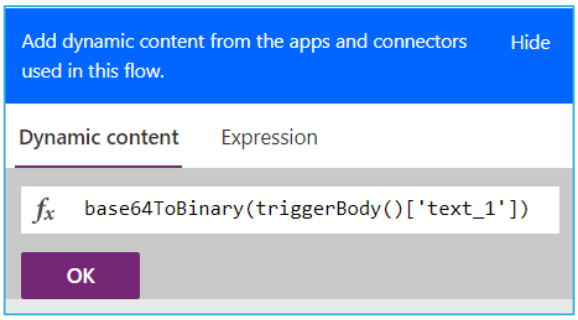
17. In the Create file action, select into the File Content field.
18. In the dynamic content box, select the Expression tab.



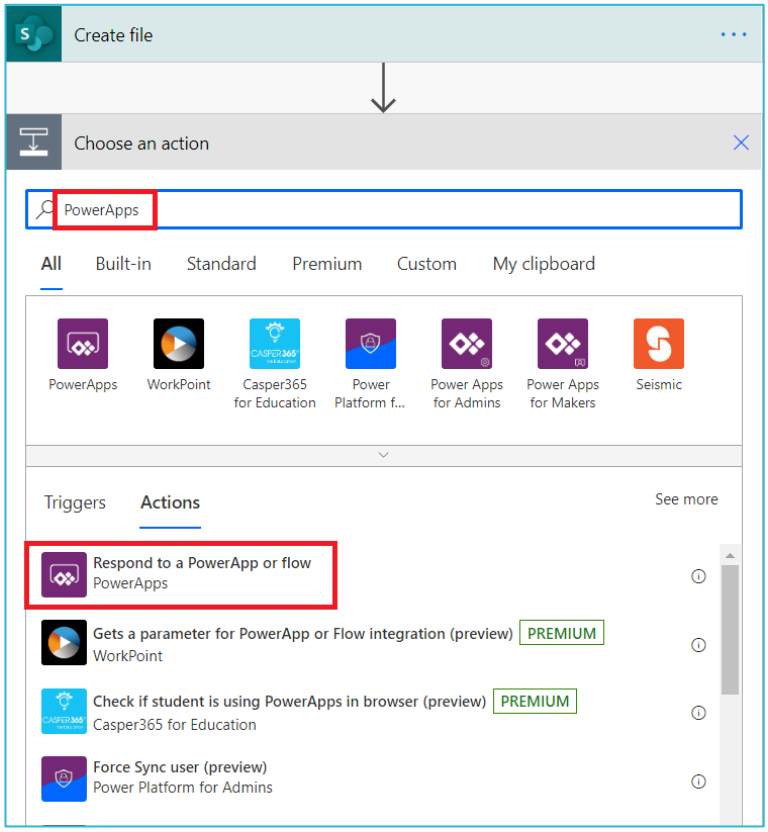
19. Scroll down to the Conversion functions section and select See more.

20. Select base64ToBinary(value).
21. base64ToBinary() will be copied into the function (fx) box. Ensure that the cursor is inside the parenthesis of that command.
22. Select the Dynamic content tab.
23. Select FileContent.
24. The function box should now read:

```
base64ToBinary(triggerBody()[text_1]).
```

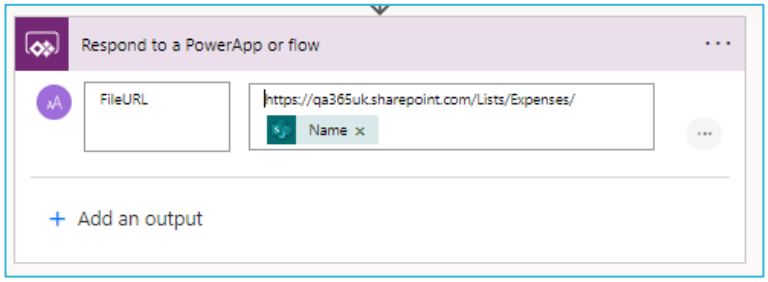


25. Select OK.
26. Below the Create file box select New Step.
27. In the search box, type Power Apps.
28. Choose the Respond to a Power App or flow action.



29. In this step, select Add an output.

30. Select Text.
31. Enter the title of FileURL.
32. Select into the value box and then type the URL to the ExpensesLibrary. Finish the URL with forward slash (/), then with the cursor just after the URL you have typed, select the Name dynamic content value.



33. Select Save to save the flow.
34. You should now be returned to PowerApps.

Task 4 – Complete the code for file upload

1. Return to your app in PowerApps.
2. Select the btnUploadFile button.
3. Add the following two commands into the OnSelect property of btnUploadFile:

```
Set(varFileContent,JSON(imgFile.Image,JSONFormat.IncludeBinaryData));
Set(varBase64Only,Mid(varFileContent,Find(",",varFileContent)+1,Len(varFileContent)-
Find(",",varFileContent)-1));
```

4. Ensure the previous code ends with a semicolon and then add the code below:

```
Set(varFileURL,UploadExpenseFile.Run(Last(uplSupportingDoc.Attachments).Name,varBase64O
nly).fileurl);
```

5. Add the following code after the last semicolon.

```
ResetForm(frmFileUpload)
```

```
Set(
    varFileContent,
    JSON(
        imgFile.Image,
        JSONFormat.IncludeBinaryData
    )
);
Set(
    varBase64Only,
    Mid(
        varFileContent,
        Find(
            ",",
            varFileContent
        ) + 1,
        Len(varFileContent) - Find(
            ",",
            varFileContent
        ) - 1
    )
);
Set(
    varFileURL,
    UploadExpenseFile.Run(
        Last(uplSupportingDoc.Attachments).Name,
        varBase64Only
    ).fileurl
);
ResetForm(frmFileUpload)
```

6. In the Tree view, expand the frmAddExpense and then unlock the SupportingDocument data card.
7. Set the following properties on the DataCardValue of the SupportingDocument data card:

Property	Value
Default	varFileURL
DisplayMode	DisplayMode.View

8. Save the app and try it. Ensure that both the SharePoint list and the document library are correctly populated. If you are previewing in the PowerApps studio, make sure you run App.OnStart.

Lab 07 – Data validation

Summary

In this lab, we will create a simple form that validates the user input before allowing submission of the data to the data source.

Task 1 – Preparation and first screen

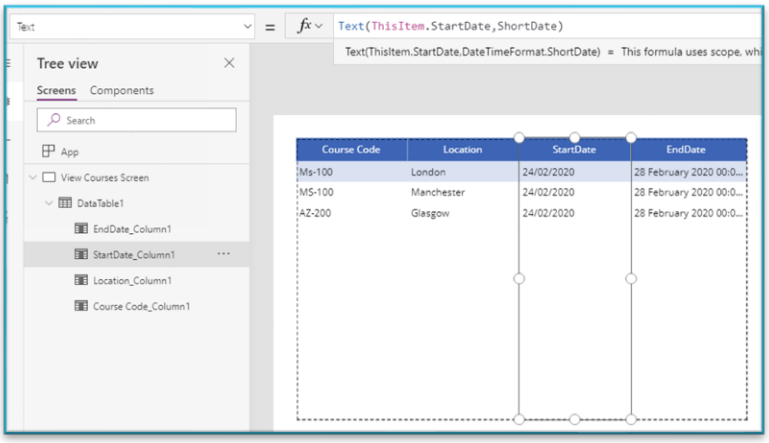
1. Upload the file C:\Student\Lab07\ValidatedCourses.xlsx to your OneDrive for Business library.
2. In PowerApps, create a new blank canvas app called Lab 07 using the tablet format. Save the app.
3. Rename the screen to View Courses Screen.
4. Add as a Data source the Courses table from ValidatedCourses.xlsx to your app using the One Drive for Business connector.
5. Add a Data table and configure the following properties:

Property	Value
Data source	Courses
Fields	Course Code, Location, StartDate, EndDate
Name	tblCourses
Height	500
Width	800
X	40
Y	40

6. Expand tblCourses and select StartDate_ColumnX.

7. Change the Text property to the following code:

```
Text(ThisItem.StartDate,DateTimeFormat.ShortDate)
```



8. Expand tblCourses and select EndDate_ColumnX.
9. Change the Text property to the following code:

```
Text(ThisItem.EndDate,DateTimeFormat.ShortDate)
```

10. Add a button and configure the following properties:

Property	Value
Name	btnAddCourse
Height	80
Text	“Add Course”
Width	300
X	900
Y	40

Task 2 – Data validation screen

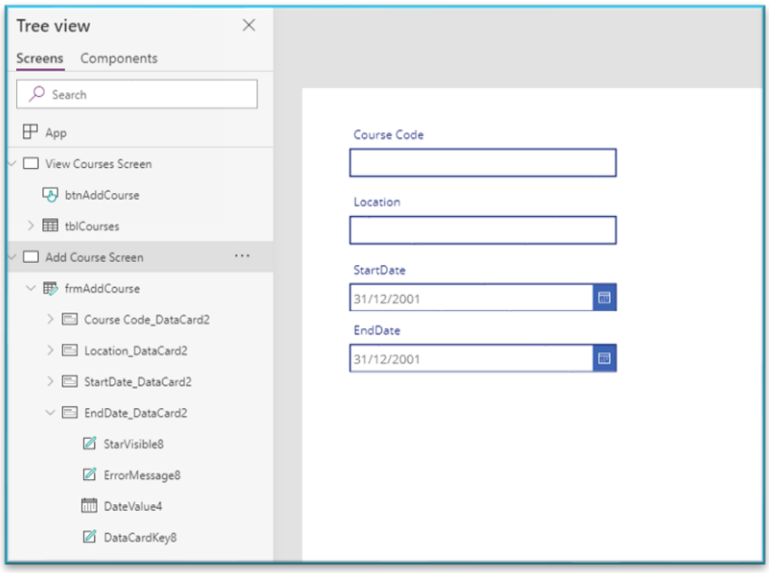
1. Add a new blank screen and rename it Add Course Screen.
2. Add an Edit form and configure the following properties:

Property	Value
Data source	Courses
Fields	Course Code, Location, StartDate, EndDate
Columns	1
Layout	Vertical
Name	frmAddCourse
Height	420
Width	450
X	40
Y	40

3. In the Tree view, expand frmAddCourse.
4. Unlock the CourseCode_DataCardX.
5. Set the card's width property to be Parent.Width.
6. Repeat the last two steps for all the remaining data cards in this form.
7. Expand StartDate_DataCardX
8. Delete the following child controls from the StartDate data card:
   - MinuteValueX
   - SeparatorX
   - HourValueX
9. Set the following properties to remove the errors that now appear:

Control	Property	Value
DataCard	Update	DateValueX.SelectedDate
ErrorMessageX	Y	0
DateValueX	Width	Parent.Width - 60

10. Repeat the previous three steps for the EndDate_DataCardX.



11. Set the OnSelect property of btnAddCourse (View Courses Screen) to the following code:

```
ResetForm(frmAddCourse);
Navigate('Add Course Screen', ScreenTransition.Fade);
NewForm(frmAddCourse)
```

12. Return to the Add Courses Screen.
13. Insert a text label control and configure the following properties:

Property	Value
Name	lblErrors
Color	Color.Red
Height	220
Size	20
Text	""
VerticalAlign	VerticalAlign.Top
Width	1000
X	70
Y	470

14. Add a button and configure the following properties:

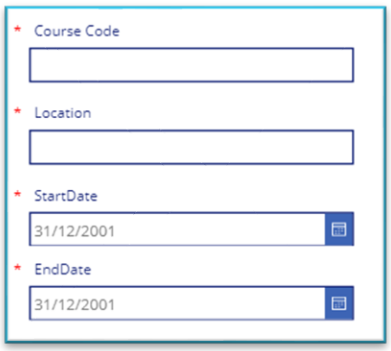
Property	Value
Name	btnSaveNewCourse
DisplayMode	If(frmAddCourse.Valid && IsBlank(lblErrors.Text),DisplayMode.Edit,DisplayMode.Disabled)
Height	90
OnSelect	SubmitForm(frmAddCourse);ResetForm(frmAddCourse);Navigate('View Courses Screen')
Text	"Save New Course"
Width	330
X	640
Y	65

Note: Although it is good practice to rename the controls in your apps, most developers only rename the top-level controls (for example, they don't rename the data cards and nested controls on forms), and this is the practice that, so far, we have been adopting. However, in the next section we will be referring to several child objects so we shall rename them to make referring to them easier.

15. In the Tree view, expand frmAddCourse.
16. Using the table below, rename the data cards and in each data card the DataCardvalueX control.

Data Card	New Data Card Name	New DataCardValue Name
Course Code_DataCardX	dcdCourseCode	txtCourseCode
Location_DataCardX	dcdLocation	txtLocation
StartDate_DataCardX	dcdStartDate	dteStartDate
EndDate_DataCardX	dcdEndDate	dteEndDate

17. Edit the properties of dcdCourseCode and set the required property to true.
18. Expand dcdCourseCode and set the color property of the child StarVisibleX to Color.Red.
19. Repeat the previous two steps for the remaining data cards.



20. Provide hint text by editing the HintText property of the following controls:

Control Name	HintText Value
txtCourseCode	"AA-000"
txtLocation	"London, Manchester or Glasgow"

21. Set the Update property of dcdCourse code to the following code:

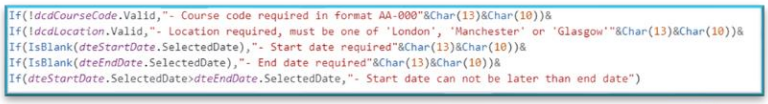
```
If(IsMatch(txtCourseCode.Text,Match.Letter&Match.Letter&Match.Hyphen&Match.Digit&Match.Digit&Match.Digit),txtCourseCode.Text)
```

22. Set the Update property of dcdLocation to the following code:

```
If(!IsBlank(LookUp(["London","Manchester","Glasgow"],Value=txtLocation.Text).Value),txtLocation.Text)
```

23. Set the Text property of lblErrors to the following code:

```
If(!dcdCourseCode.Valid,"- Course code required in format AA-000"&Char(13)&Char(10))&
If(!dcdLocation.Valid,"- Location required, must be one of 'London', 'Manchester' or 'Glasgow'"&Char(13)&Char(10))&
If(IsBlank(dteStartDate.SelectedDate),"- Start date required"&Char(13)&Char(10))&
If(IsBlank(dteEndDate.SelectedDate),"- End date required"&Char(13)&Char(10))&
If(dteStartDate.SelectedDate>dteEndDate.SelectedDate,"- Start date can not be later than end date")
```



24. Save the app and test it. (Make sure you select the first screen when testing.)

End of Labs

Congratulations – you have now completed all the core labs for the Mastering Power Apps course. You have worked through advanced controls, data shaping, flow integration, and validation techniques - well done!

Please keep an eye on your email for the end of course assessment, we would be very grateful if you completed this.

Appendix

This lab is optional and intended for learners working with embedded SharePoint list forms. It is no longer required for the core course flow.

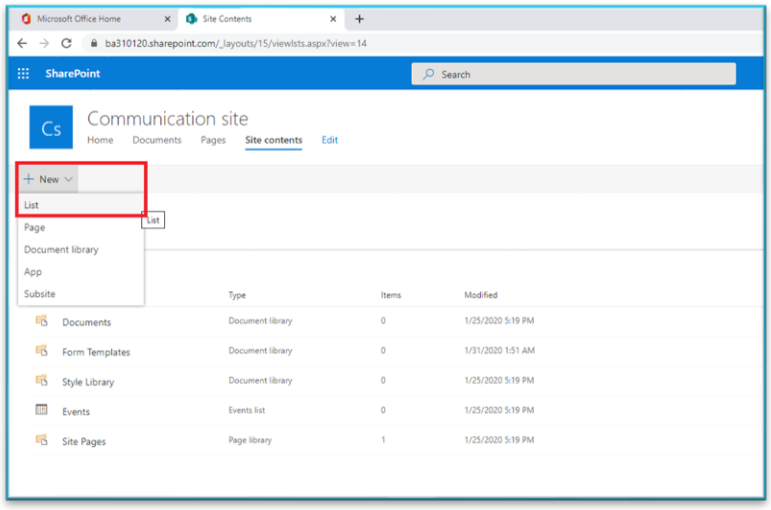
Lab – Replacing SharePoint forms

Summary

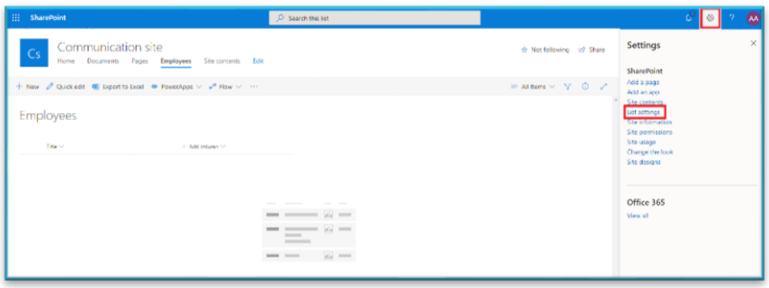
In this lab, we will be replacing the built-in SharePoint forms with ones we design in Power Apps.

Task 1 – Setting up SharePoint list

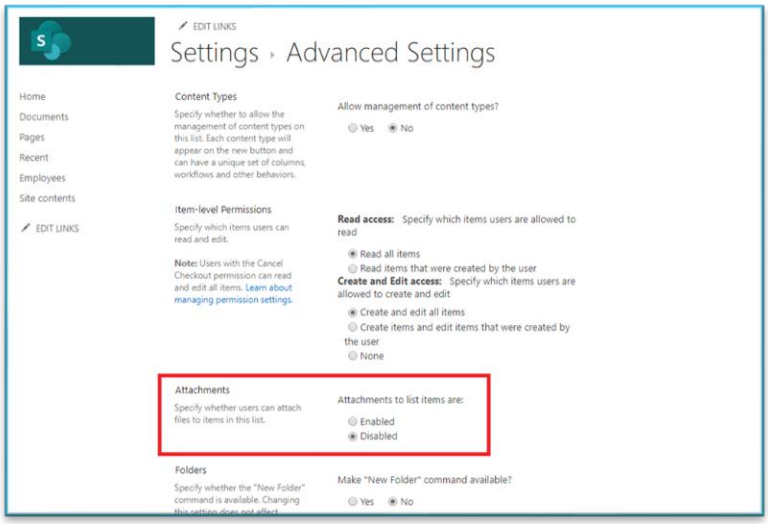
1. Using the Chrome instance configured for this course, navigate to your Communication site in SharePoint.
2. Select the link for Site contents.
3. Select New and then select List from the drop-down.



4. Create a Blank list.
5. Give the list the name Projects, and then select Create.
6. If you are presented with a welcome wizard, close it.
7. In the top right, select the settings cog and then choose List Settings.



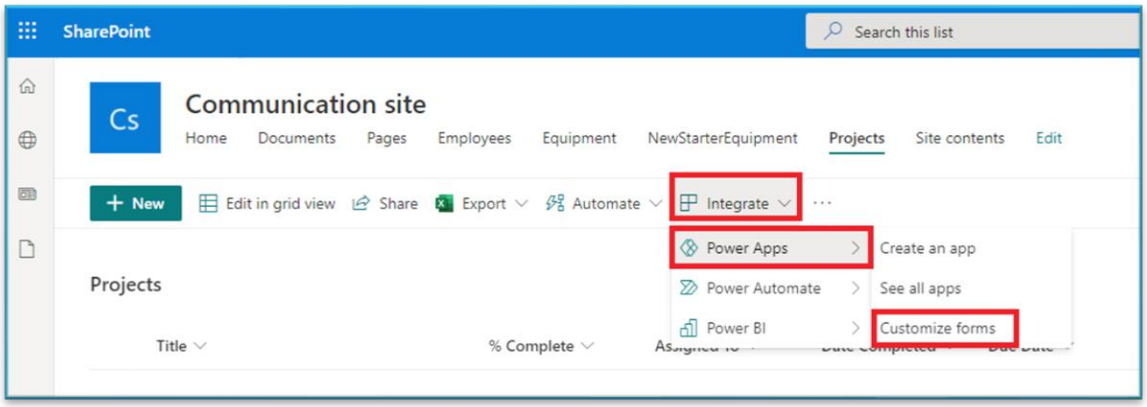
8. Select Advanced settings, and then set Attachments to Disabled. Scroll down and select OK.



9. In the warning box, select OK.
10. Just below the line in the columns section, select Add from existing site columns.
11. Use the Select site columns from drop-down to choose Core Task and Issue Columns.
12. Select and add the following site columns and then select OK:
    - % Complete
    - Assigned To
    - Date Completed
    - Due Date
    - Priority

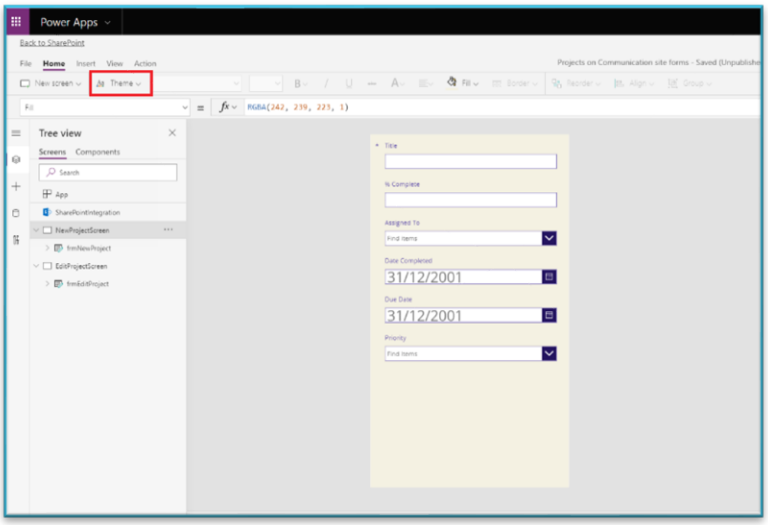
Task 2 – Create a SharePoint form with two screens

1. Navigate to the Projects SharePoint list.
2. Select Integrate | Power Apps from the tasks bar and then select Customize forms.



3. Using the Tree view, duplicate FormScreen1.

4. Name the two screens: NewProjectScreen and EditProjectScreen.
5. Rename the form on NewProjectScreen to frmNewProject.
6. Rename the form on EditProjectScreen to frmEditProject.
7. To clearly distinguish between the SharePoint native form and the Power Apps form, use the Theme drop down to apply a theme to the Power Apps (for example Lavender)



8. Change the Y value of frmNewProject to 70.
9. Insert a Text label control to NewProjectScreen and set the following properties:

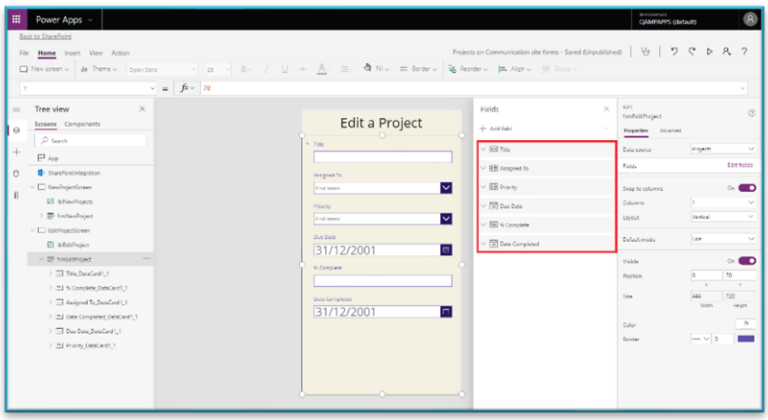
Property	Value
Name	lblNewProject
Align	Align.Center
Height	70
Size	28
Text	"Create a New Project"
X	0
Y	0
Width	444

10. Change the Y value of frmEditProject to 70.

11. Insert a Text label control to EditProjectScreen and set the following properties:

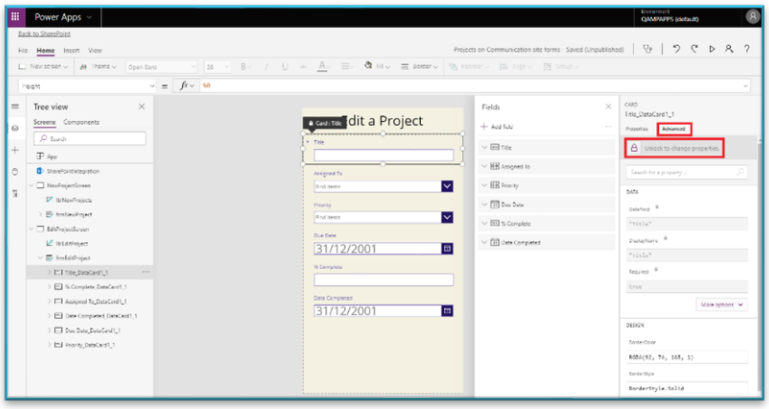
Property	Value
Name	lblEditProject
Align	Align.Center
Height	70
Size	28
Text	"Edit a Project"
X	0
Y	0
Width	444

12. In the Tree view, select frmNewProject.
13. In the properties pane, select 0 selected.
14. Remove the % Complete and Date Completed fields.
15. Close the fields pane.
16. In the Tree view, select frmEditProject.
17. In the properties pane, select 0 selected.
18. Change the fields order so they are (from top to bottom):
    - Title
    - Assigned To
    - Priority
    - Due Date
    - % Complete
    - Date Completed

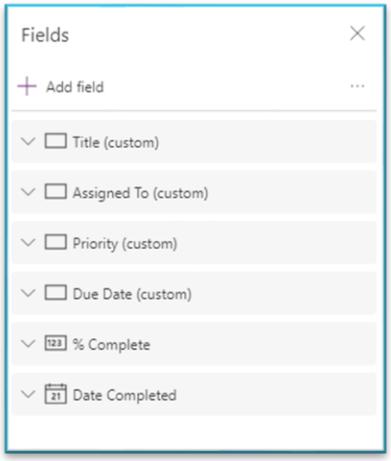


19. In the Tree view, expand frmEditProject, and then select the datacard for the Title (Title_DataCard_X).

20. In the properties pane, select Advanced and then Unlock the data card.



21. Set the DisplayMode of the card to be DisplayMode.View.
22. Repeat the previous two steps to set the DisplayModes of Assigned To, Priority, and Due Date to DisplayMode.View.
23. If you re-check the form's fields dialog, you should now see that the top four fields now show as custom ones.



24. Set the following properties on the screens:

Screen	Property	Value
NewProjectScreen	OnVisible	Set(varForm,"New")
EditProjectScreen	OnVisible	Set(varForm,"Edit")

25. In the Tree view, select the SharePointIntegration node.

26. Edit the properties as shown in the tables below:

Property Name	OnCancel
Original Value	ResetForm(frmNewProject)
New Value	ResetForm(frmNewProject);ResetForm(frmEditProject)

Property Name	OnEdit
Original Value	EditForm(frmNewProject)
New Value	EditForm(frmEditProject);Navigate(EditProjectScreen)

Property Name	OnNew
Original Value	NewForm(frmNewProject)
New Value	NewForm(frmNewProject);Navigate(NewProjectScreen)

Property Name	OnSave
Original Value	SubmitForm(frmNewProject)
New Value	If(varForm="New",SubmitForm(frmNewProject),SubmitForm(frmEditProject))

Property	OnView
Original Value	ViewForm(frmNewProject)
New Value	ViewForm(frmEditProject);Navigate(EditProjectScreen)

27. Using the file menu, save the app and then Publish to SharePoint.
28. Return to the Projects list in SharePoint and create a new item.
29. View the item you just created and confirm you get the Edit form, but it is read only.
30. Edit the item and confirm that you can only edit the % Complete and Date Completed fields.

Learn. To Change.

QA.com