Microsoft Power Platform

Hands-on Labs

Activate – Power Platform: Building a Power App

Published: May 2024
Workshop Version: 2.7.6

Table of Contents

Lab Overview

Abstract

This is a beginner level lab with optional intermediate modules for you to get hands-on experience with the Microsoft Power Platform technologies: Power Apps and Power Automate.

The lab includes step-by-step instructions to build a **Device Orders** solution in a few hours, using existing modern best practices.

Technologies covered:

- **Power Apps**: A software-as-a-service application platform that enables power users in line of business roles to easily build and deploy custom business apps. You will learn how to build *canvas apps*.
- **Power Automate**: A business service for a line of business specialists and IT Pros to build automated workflows and integrations. You will learn how to build *cloud flows*.

Make sure to follow all the pre-requisite steps listed in this document before starting the labs.

Once you have completed the lab, please share your feedback with the Engineer delivering this course.

For a list of additional learning resources and introductory videos, see: http://aka.ms/powerapps-resources

Lab Structure and Learning Objectives

This lab is divided into one primary module and a few optional modules:

The primary module focuses on:

1. Power Apps canvas apps best practices
2. Working with connected data sources
3. Working with screens and responsive UI
4. Working with classic and modern controls
5. Using Power Fx formulas, and actions to customize the user experience
6. Saving, publishing, and sharing an app
7. Integrating Power Automate with Power Apps

Optional modules cover the following topics:

1. Packaging as a Power Platform solution
2. Integration with Microsoft Teams

Prerequisites

Task 1: Ask your instructor to share a copy of the lab files

1. Save a local copy of the lab contents. Save it to a local folder such as C:\PowerAppsLab.

Task 2: Sign-in to Power Apps

1. Go to https://make.powerapps.com and sign-in with your business or school account.
   This is the same as your Microsoft 365 login.

Task 3: Check environment and connector access

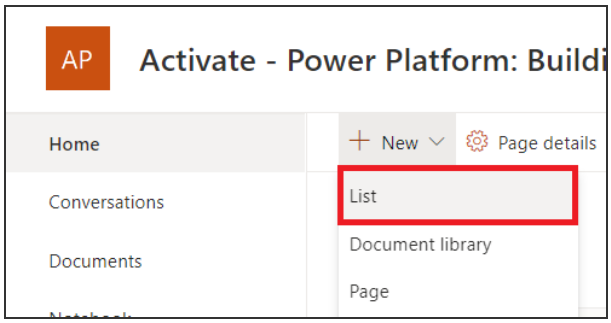
1. Ensure that you have access to the correct environment to create test Power Apps and Power Automate flows (as per your organizations governance and usage policy)
2. Ensure that the Approvals connector is available in the environment
3. If you plan to use Dataverse in this lab, make sure environment and licensing are appropriate.

If you have any questions, please, reach out to the instructor.

Task 4: Prepare data source to track device order data

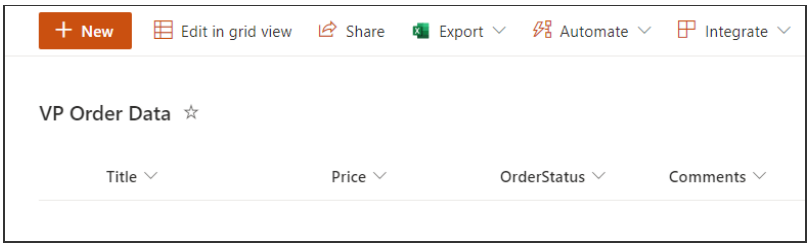
> **Note**: Power Apps can work with a variety of data sources.
> Primary options include SharePoint, Dataverse, and SQL Server, as these data sources are delegable and can handle larger datasets.

1. Create a list with name **<YOUR INITIALS> Order data** (e.g., *VP Order Data*).



2. Add the following columns to the list:
    - **Price**: Number
    - **OrderStatus**: Single line of text
    - **Comments**: Single line of text

    Title column exists out-of-box and will be used later **DO NOT rename it**.



Power Apps canvas studio layout

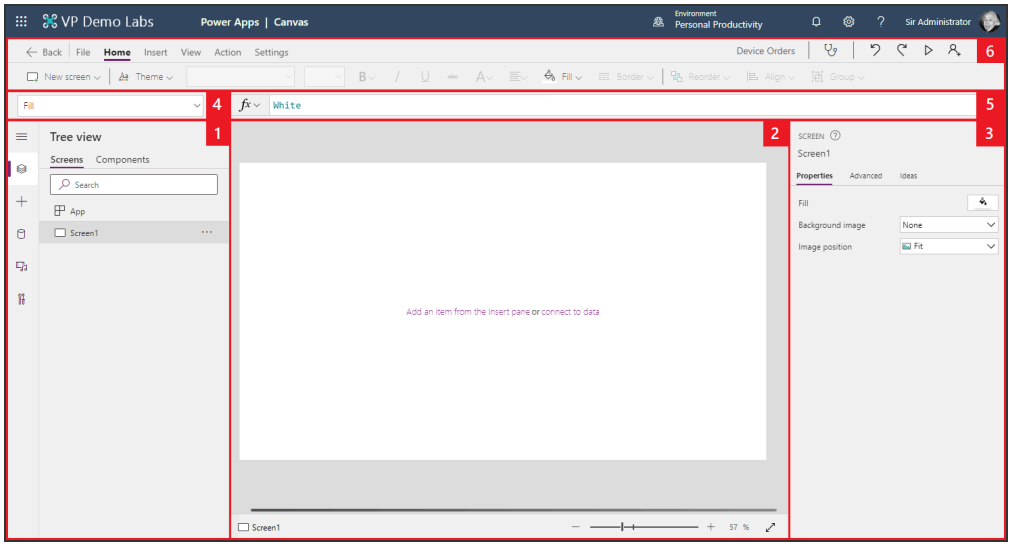
Power Apps canvas studio is a web-based application (https://make.powerapps.com) that you can use in any of the following browsers:

Browser	Operating system
Microsoft Edge (latest version)	Windows 10 or later
Google Chrome (latest version)	Windows 10 or later, macOS 10.13 or later

Power Apps canvas studio has three panes and a toolbar that make app creation feel like building a slide deck in PowerPoint. Formulas are entered within a function bar that is like Excel.

Interface layout:

1. **App authoring pane**, which allows you to navigate screens and controls, data sources etc.
2. **Canvas pane**, which contains the app screen that you are working on, screen selector and zoom options
3. **Properties pane**, where you configure properties for controls, bind to data and set additional advanced settings
4. **Properties list**, where you select the property for the selected control that you want to configure
5. **Formula bar**, where you add PowerFx formulas that define the behavior of a selected control
6. **Toolbar** (toolbar), where you perform common actions including customizing design elements.



Device Orders scenario

Imagine an organization where every three years, the employees go through a hardware refresh cycle. The organization would like to build a customized app that runs on the web and mobile devices that will help streamline the device order and approval process. Moreover, they do not have traditional development resources available, such as .NET, Javascript/Typescript developer, to create this application.

Solution Overview

Microsoft Power Platform technologies enable users familiar with technology (aka ‘citizen developers’) to build a customized device ordering solution. The application user interface and interaction logic are built in Power Apps, the approval process is automated using Power Automate, and the device order data is stored in a SharePoint list.

Key features of the solution:

1. Ability to browse through a selection of devices and filter by its category or model line
2. Select devices to compare
3. View detailed specifications for the selected devices on a second comparison screen
4. Select a device to order
5. Order / request a device
6. Send an automated approval request email when the order is placed
7. Allow the approver to approve or reject an order and add comments without leaving their inbox
8. Notify the user when an order is approved or rejected

Exercise 1: Creating the app and layout

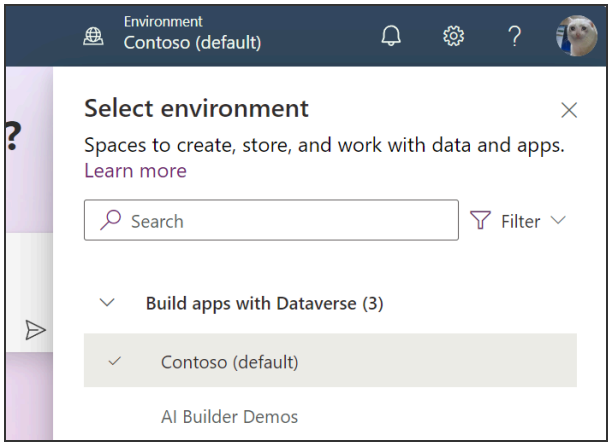
> **Important:**
> Do not start this exercise unless **Prerequisites** are complete.
> If you are experiencing problems with progressing through these labs – please, **ask your instructor for help**.

Task 1: Sign into Power Apps

In this task, you will sign into Power Apps

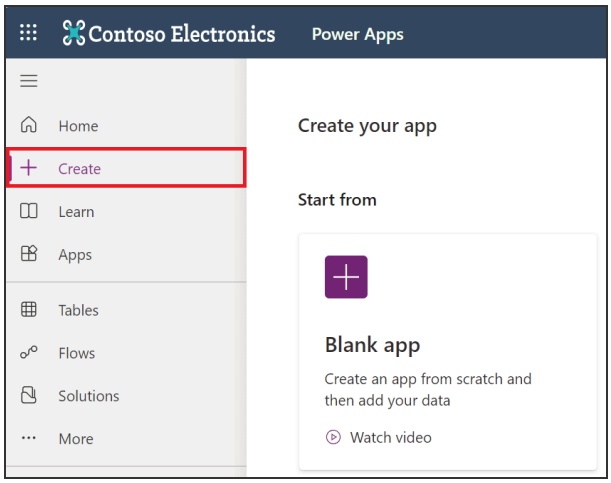
1. Navigate to https://make.powerapps.com and sign in with your business or school account if you have not already done so
2. Before creating an app, switch to the correct environment (where your organization permits development apps to be created).
   Click the Environments drop-down in the top right of the screen to switch to the correct environment.
   > **Note:** Developer environment may be a preferred option if you plan to do optional exercises in this lab.

   Ask instructor if you have doubts.

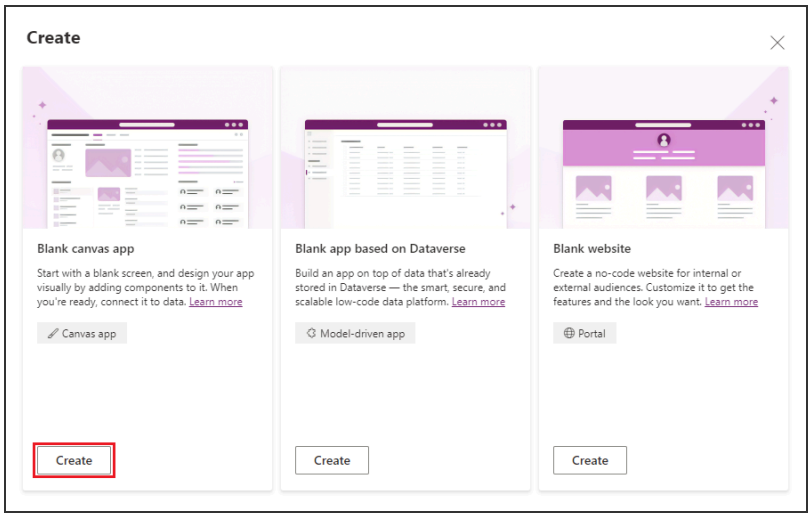


Task 2: Create a new app

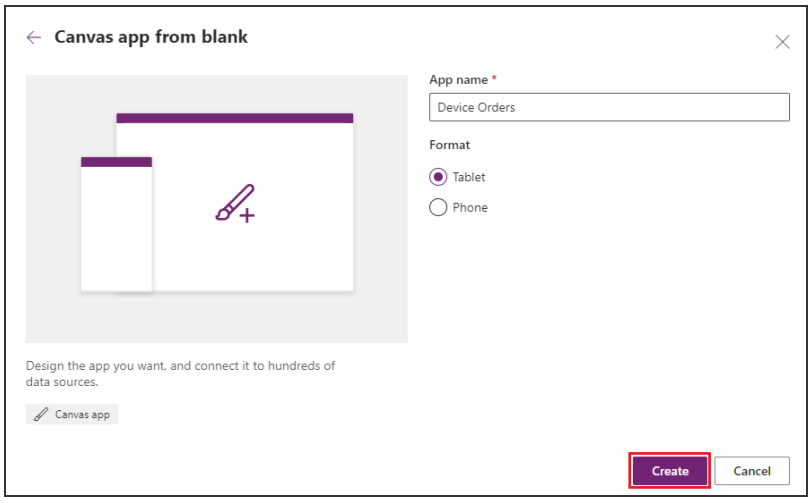
1. In the Home screen, find **+Create** in left navigation menu, then click **'Blank app'** tile



2. In the next dialog under 'Blank canvas app' option click 'Create'



3. Input 'Device Orders' in the App name field. Select 'Tablet' for Format. Click **'Create'**

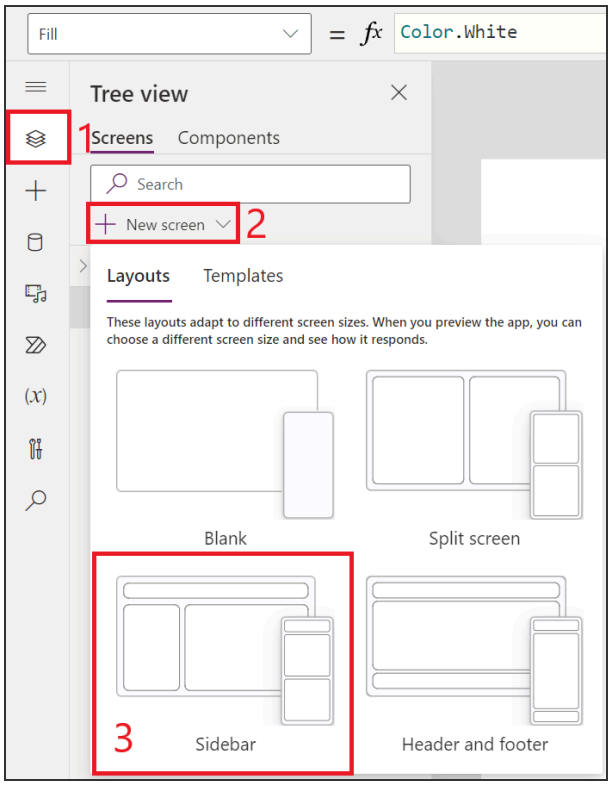


4. Click 'Skip' if you receive the Welcome to Power Apps Studio prompt
5. If prompted, select your region, then click **'Get Started'**

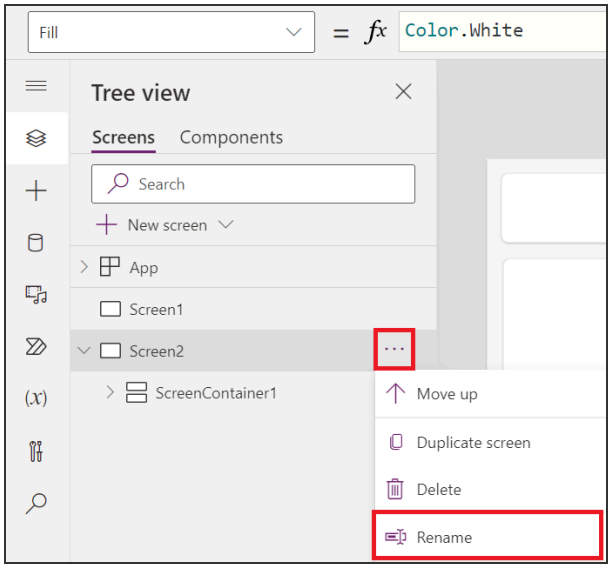
Task 3: Add a screen with layout

> **Canvas Apps Coding Standards and Guidelines**
> It is a good practice to rename screens and controls as you create them so that they are easier to locate as you work with formulas that reference different controls. In this lab, we will rely on the best practices and naming conventions defined in Canvas apps coding standards and guidelines.
> Full document link: https://aka.ms/powerappscanvasguidelines

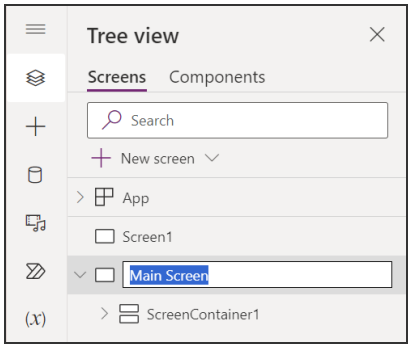
1. Make sure you have selected **Tree View (1)** in the **App authoring pane**
   Click **+New screen (2)** and select **Sidebar (3)** layout



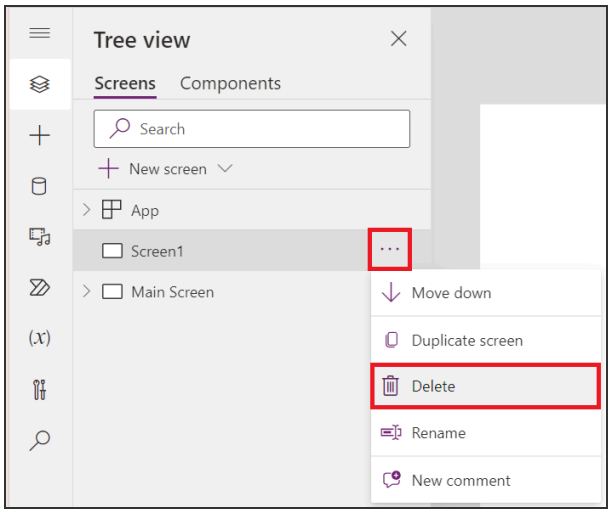
2. Click **ellipsis "…"** next to Screen2 (or right click) and select the **Rename** option



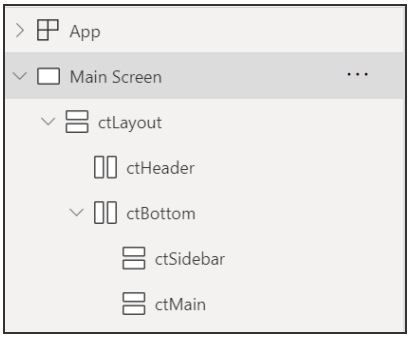
3. Change the name to **Main Screen**



4. Click **ellipsis "…"** next to Screen1 (or right click) and select the **Delete** option



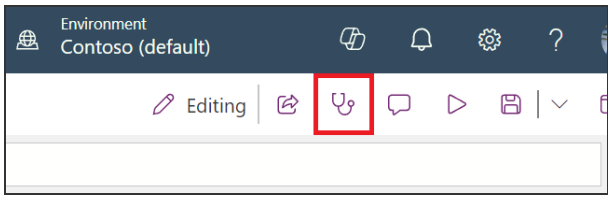
5. In **Tree View** pane, **Rename** containers according to naming conventions:
   - ScreenContainer1 -> ctLayout
   - HeaderContainer1 -> ctHeader
   - BottomContainer1 -> ctBottom
   - SidebarContainer1 -> ctSidebar
   - MainContainer1 -> ctMain



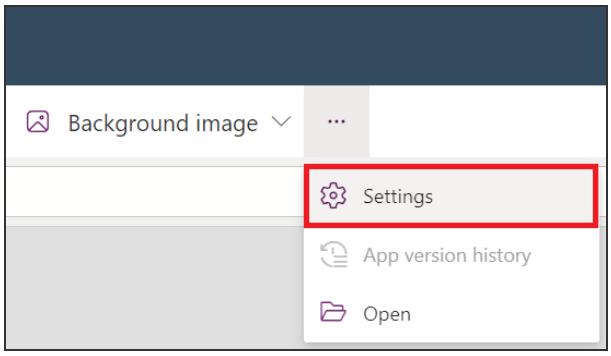
Task 4: Save the application

In this task, you will save an initial version of the app. Power Apps saves automatically each 2 minutes after the first save (Tip: This setting can be disabled).

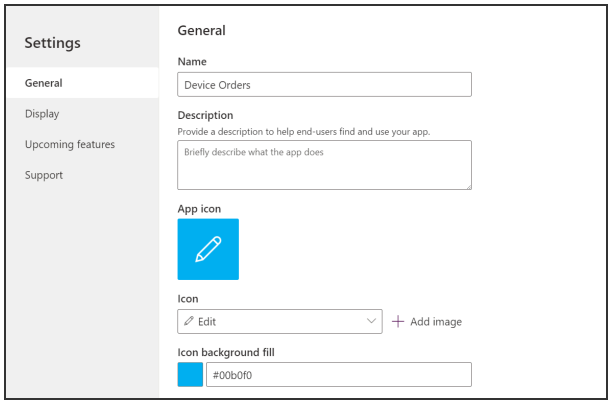
1. First, check if there are any errors in the app.
   Click on the **App checker** icon



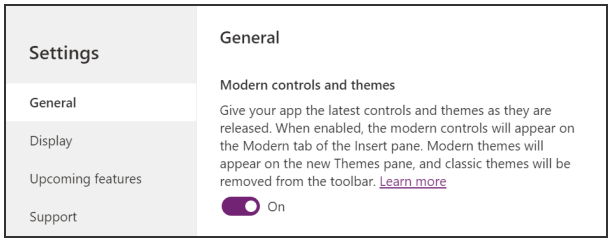
2. **App checker** pane will become visible. If there are any errors, they will be displayed here
3. Close the App checker pane
4. Click **Settings** button in the toolbar



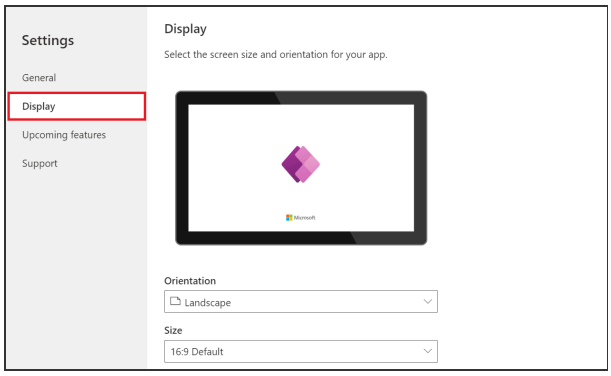
5. In the **Settings – General** dialog, you can:
   i. Change your App name
   ii. Customize the App icon (choose a background color and icon)



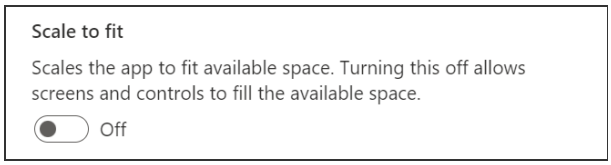
   iii. Enable **Modern control and themes**



6. Select **Display** tab to view available screen size and orientation and aspect ratio settings. For this app, we will leave it at the default setting of Landscape with 16:9 aspect ratio



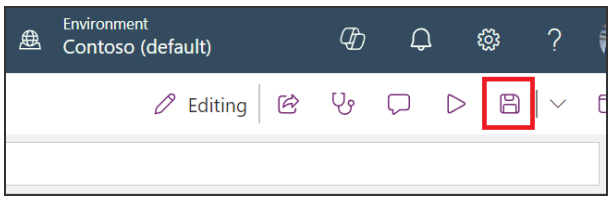
7. Set **Scale to fit** as **Off**.



   This setting will make app screen responsive instead of simple app resize, but it will require you to leverage techniques to enable dynamic layout.

   For more details read: Create responsive layouts in canvas apps

8. Click **'X'** in the top-right corner to close the dialog
9. Click **Save** button in the toolbar toolbar

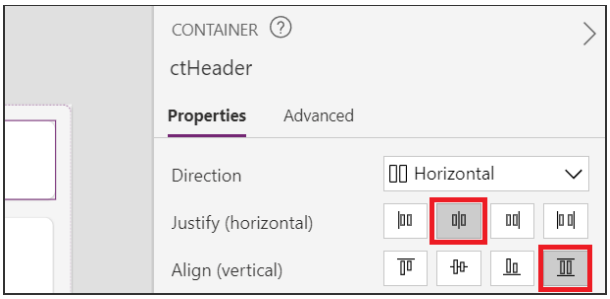


> **Note:** In Power Apps, when you save a version of your app, the first version is published by default and available to everyone that you share the app with. Subsequent saves are only visible to the app maker in the studio. You must explicitly publish it for all app users to get the update.

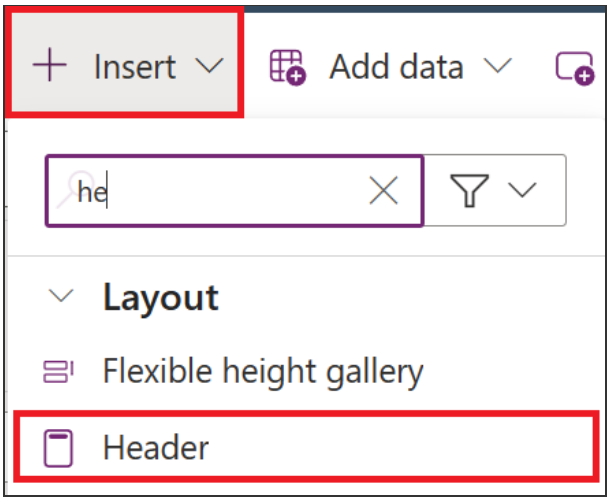
Task 5: Add a screen header

In this task, you will add a Header (modern control) to the Main Screen that displays a logo, a screen name and the logged in user's photo.

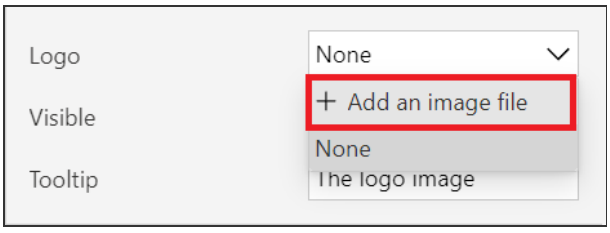
1. Select **ctHeader** container
   a. Justify (horizontal) -> **Center**
   b. Align (vertical) -> **Stretch**



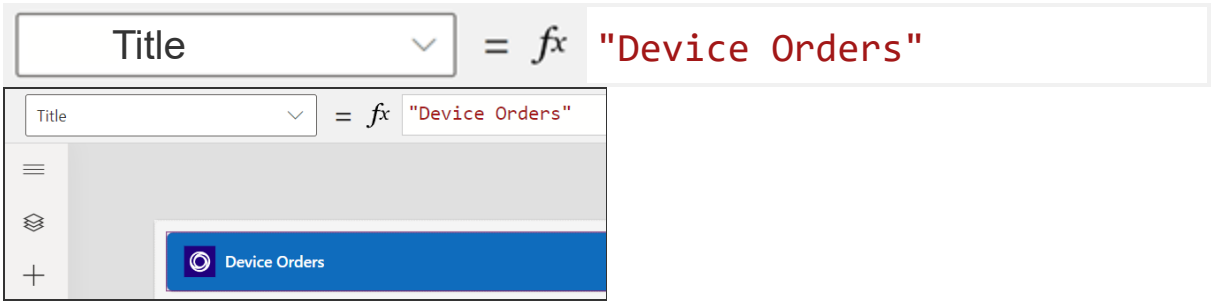
2. Select the **Insert** button in the toolbar and select **Header** control



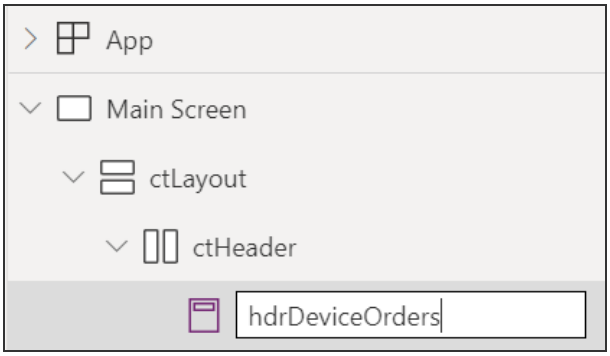
3. In Properties pane, find **Logo** property and **+Add an image file**. Upload *contoso-logo.png* from the lab files to use it as a logo.



4. Set **Title** property to **Device Orders**



5. Change the **Header** control name to **hdrDeviceOrders**



6. Save the app

Exercise 2: Connect the app to data

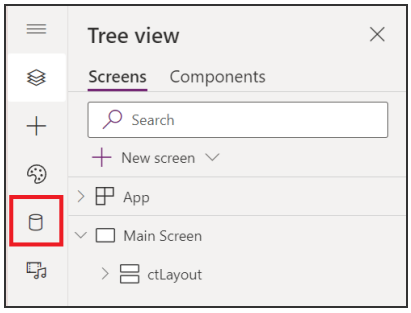
In this exercise you will learn how to connect the app to the various data sources required for the app to both display devices and track device orders.

> **Note:** What is a data connection? In Power Apps, a data connection adds data to your app. Your canvas app can connect to SharePoint, Dataverse, SQL Server, and over 1000 first- and third-party services.

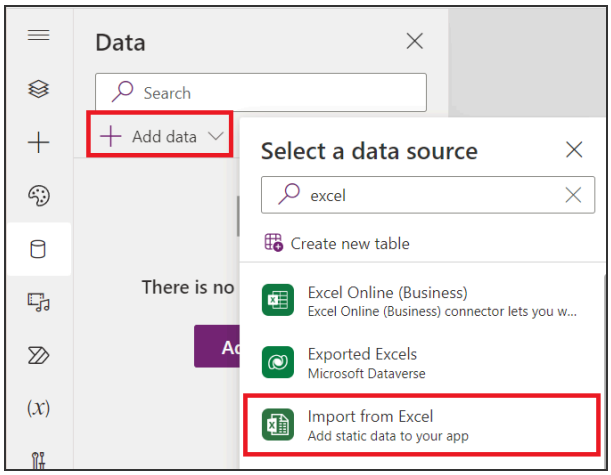
Task 1: Import static data from Excel

In this task you will learn how to import static data from Excel spreadsheet containing device data for the solution.

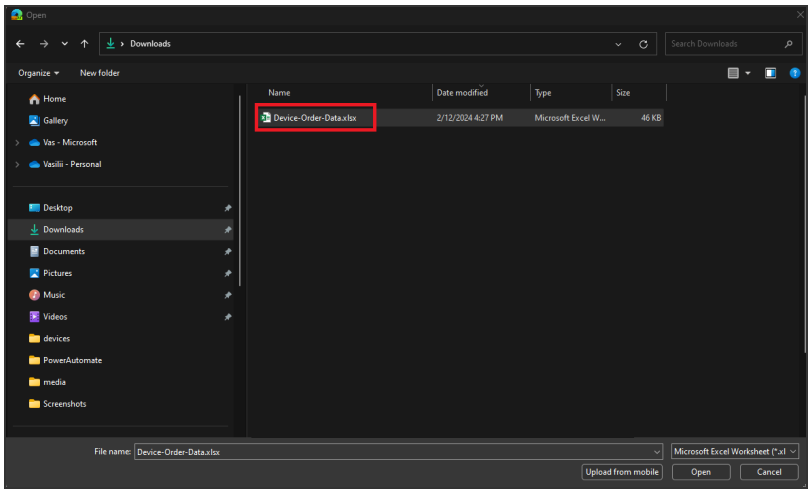
1. In the app authoring pane, click the **Data** icon



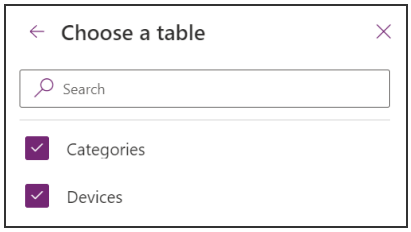
2. Click **+Add data** dropdown, in popup menu type **Excel** into the **search** box



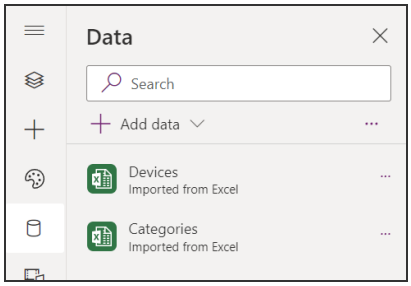
3. Click **Import from Excel**
4. The Open dialog will appear, browse to the location where the spreadsheet *'Device-Order-Data.xlsx'* is saved (as in Task one of the Prerequisites)
5. Click **Open**



6. In the right-hand pane, from the **Choose a table** dialog, select both the **Devices** and **Categories** tables



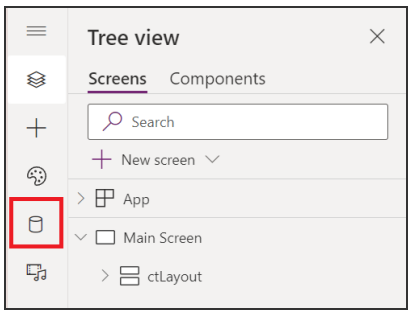
7. Click **Connect**
8. **Devices** and **Categories** connections should be listed in the **Data** pane



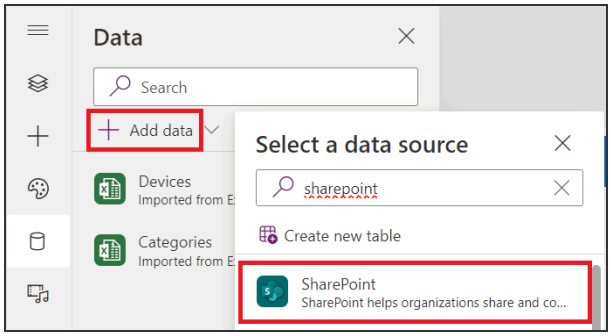
Task 2: Create a connection to store device orders

Depending on selected data source, you will create a connection to store device orders in SharePoint list.

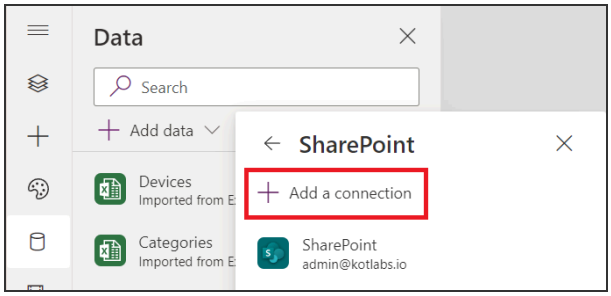
1. In the app authoring pane, click the **Data** icon



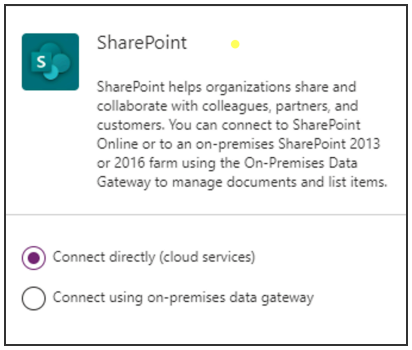
2. Click **+Add data** dropdown, in popup menu type **SharePoint** into the search box
3. Select **SharePoint**



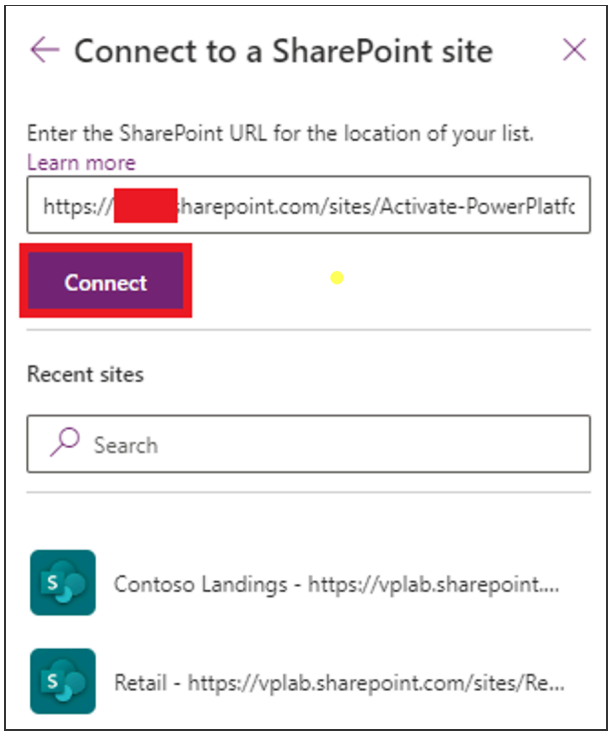
4. If this dialog is displayed, click **Add a connection**



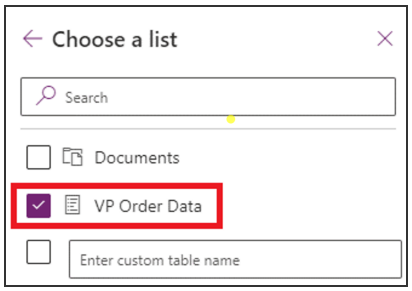
5. Ensure that **Connect directly (cloud services)** is selected



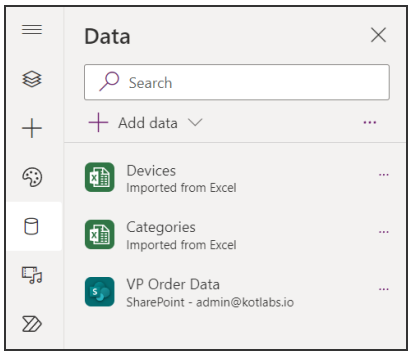
6. Click **Connect**
7. In the 'Enter the SharePoint URL for the location of your list' input box, paste the **URL** of your SharePoint site, or select the sire URL from the list of recent sites and click **Connect**



8. In the **Choose a list** dialog, select the **'<your initials> Order Data'** list (created in task four of the Prerequisites)



9. Click **Connect**
10. **'<your initials>\ Order Data'** connection will appear in the list of connections in the **Data** pane



Exercise 3: Add a device gallery to the Main Screen

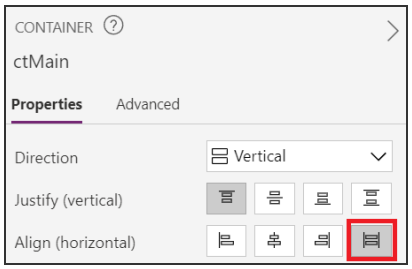
In this exercise, you will add a gallery containing all available devices making it easy for users to browse the list and get a quick overview of the devices available.

> **Note:** What is a gallery? A gallery control can show multiple records from a data source, and each record can contain multiple types of data. For example, a Gallery control can show multiple contacts with each item showing contact information that includes a name, address, and a phone number for each contact.

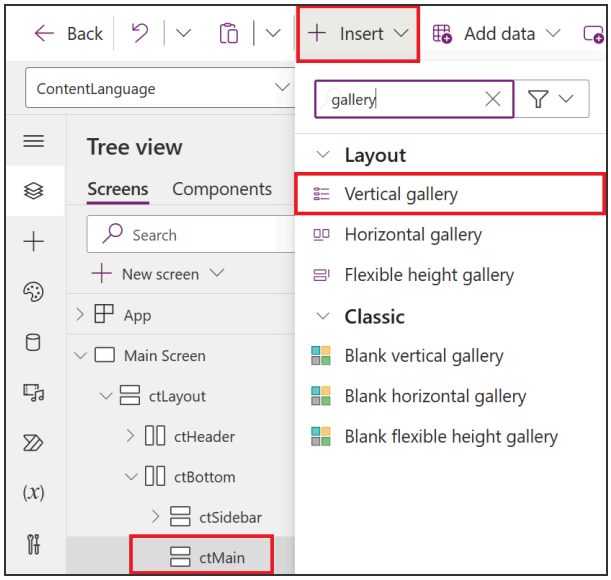
Task 1: Add a device gallery

In this task you will learn how to add a gallery control to your app containing a list of all devices.

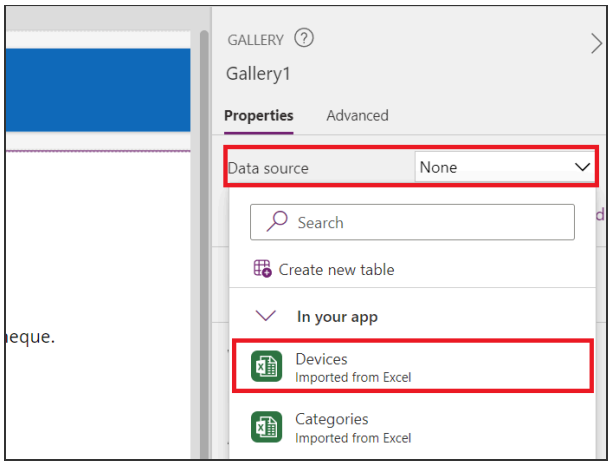
1. Select **ctMain** and in Properties pane, set **Align (horizontal)** property as **Stretch**



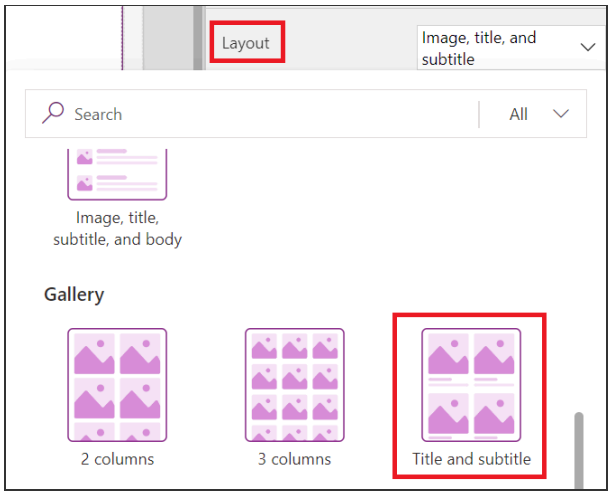
2. Ensure **ctMain** is still selected and in **Insert** dialog search for **Vertical gallery**
   Add it to **ctMain**



3. Go to the **Properties pane**, click on the **Data source** drop-down (or click **Data** in gallery menu)
4. Select **'Devices'** from the list of data connections under the 'In your app' section



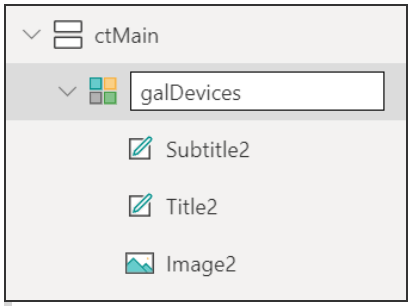
5. In the **Properties pane**, click on the **Layout** drop-down and select **Title and subtitle** (or click **Layout** in gallery menu)



6. Check that **Align in container** property is *Set by container* and set **Flexible Height** as *On*

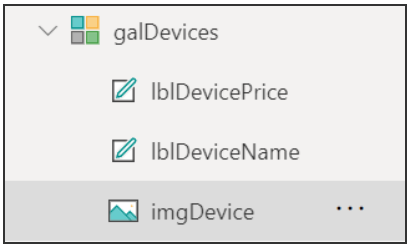
7. **Rename** Gallery1 to **galDevices**.

   Notice in **Tree View**: this gallery with three controls within it – two text labels (title, subtitle) and an image.



> **Note:** Galleries provide a powerful way to visualize tabular data in Power Apps. It is important to become familiar with customizing a gallery. Key components of a gallery include: the gallery control, the template cell (the first cell), and controls within the template cell.

8. In Tree View, expand **galDevices** and rename controls this way:
   - **Title1** -> **lblDeviceName**
   - **Subtitle1** -> **lblDevicePrice**
   - **Image1** -> **imgDevice**



9. **Select** the **galDevices** and set **TemplateSize** as *300*

> **Note:** To select the entire gallery – click on the gallery in the Tree view on the left or click on the second or third cell inside the gallery. Clicking any cell that is NOT the first cell of the gallery will select the entire gallery. Now you can specify properties that apply to the entire gallery, such as the TemplateSize.

```
TemplateSize = 300
```

10. To make the gallery responsive and display multiple columns of devices, depending on the screen size, set the **WrapCount** property to:

```
WrapCount = Switch(
 App.ActiveScreen.Size,
 ScreenSize.Small,
 1,
 ScreenSize.Medium,
 2,
 4
)
```

> **Note:** For the locales with decimal delimeter "," (comma), formula will be slightly different. Please, notice that localized formulas given further for your convenience.

Localized formula:

```
WrapCount = Switch(
 App.ActiveScreen.Size;
 ScreenSize.Small;
 1;
```

```
 ScreenSize.Medium;
 2;
 4
)
```

11. Select **lblDeviceName** and set **Text** property to **ThisItem.Title**

```
Text = fx ThisItem.Title
```

12. Select **lblDevicePrice** and set **Text** property to **ThisItem.Price**

```
Text = fx ThisItem.Price
```

13. Select **imgDevice** and set **ImagePosition** property to **ImagePosition.Fit**

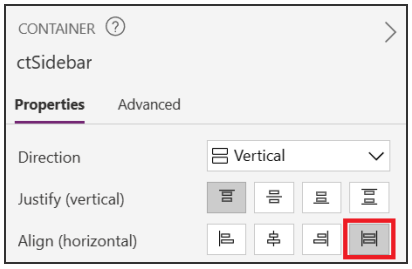
```
ImagePosition = fx ImagePosition.Fit
```

14. Resulting **galDevices** gallery should look like this:

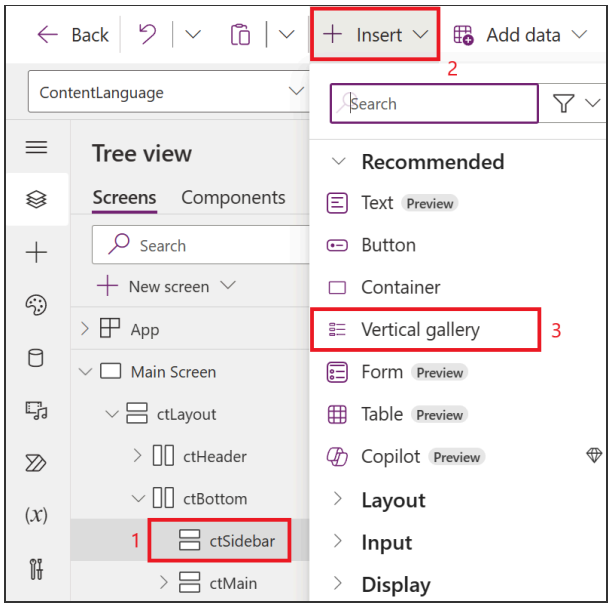
Task 2: Add a gallery to show device categories

In this task you will add a second gallery that lists the various device categories, or product lines. This will be a single column vertical gallery positioned on the left side of the screen, with each cell displaying device category name, and its photo. This gallery will later be used as a filter for the device gallery created above.

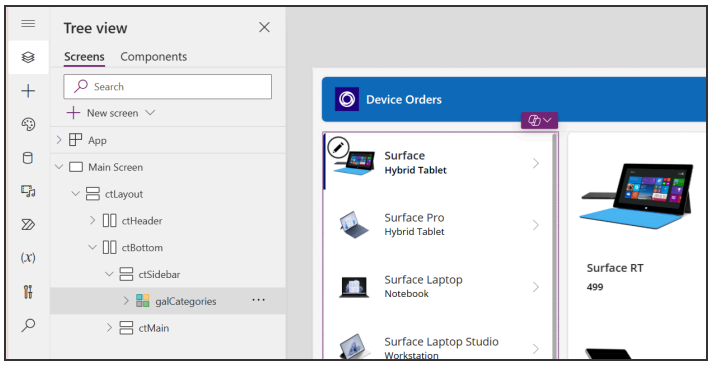
1. Select **ctSidebar** as set **Align (horizontal)** property as **Stretch**



2. Select **ctSidebar** and in **Insert** dialog search for **Vertical** gallery
   Add it to **ctSidebar**



3. In the **'Select a data source'** dialog, select **Categories**
4. **Rename** the gallery to **galCategories**
5. The app should look like the image below



6. Click **Save** button in the toolbar to save the app

Task 3: Highlight the selected item in the gallery

In this task, you will learn how to use the TemplateFill property of the manufacturer gallery to specify a highlight color for the item.

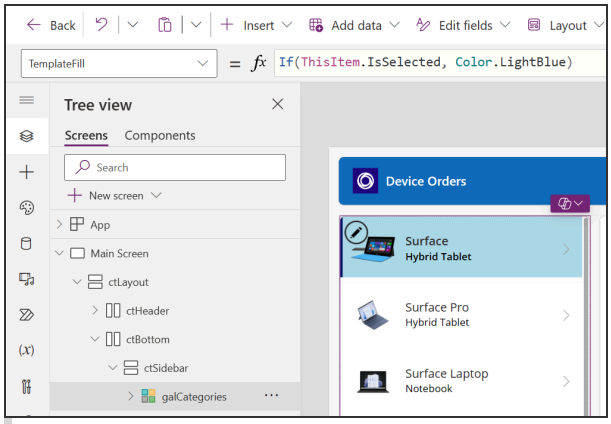
1. Select the **galCategories**
2. From the properties drop-down, select the **TemplateFill** property
3. In the **formula bar**, enter the following:

```
TemplateFill = fx  If(ThisItem.IsSelected,
                   Color.LightBlue)
```

Localized formula:

```
TemplateFill = fx  If(ThisItem.IsSelected;
                   Color.LightBlue)
```

4. Use the **preview mode** to perform a quick test of the highlighting. You can enable preview mode by holding down the Alt key and clicking a few different manufacturers in the gallery. Preview mode ends when you stop holding the Alt key.



> **Note:** You could alternatively click the **Play** button to enter preview mode, and to exit this you would click the X in the upper right corner or use the Esc key.

Task 4: Filter the devices based on the selected category

In this task, you will learn to use the **Filter()** function to filter the items in the **galDevices** to only display devices that match the selected item in the **galCategories**.

1. Select the **galDevices**
2. From the properties drop-down, select the **Items** property
3. Enter the following expression in the formula bar:

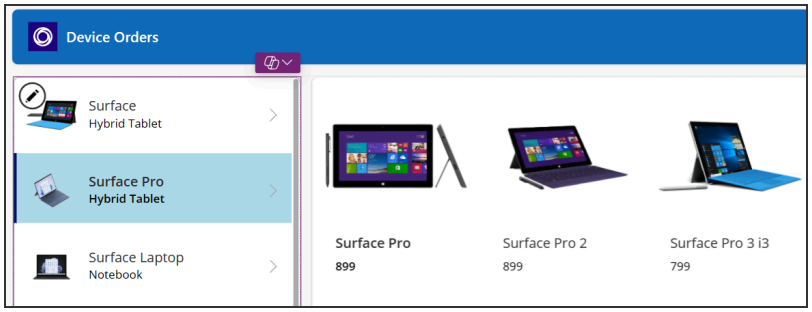
```
Items = fx
Filter(
 Devices,
 CategoryID =
galCategories.Selected.CategoryID
)
```

Localized formula:

```
Items = fx
Filter(
 Devices;
 CategoryID =
galCategories.Selected.CategoryID
)
```

This will filter the device gallery to only display items that match the selected category based on CategoryID.

1. Enter preview mode
2. Select a different item in the **galCategories** gallery on the left, and you will notice the device gallery will update accordingly



Task 5: Add price formatting and highlight prices above $1,500

In this task you will learn how to configure text label format, using Text() function, and how to implement conditional formatting.

1. In Tree View, expand **galDevices** gallery
2. Select the **lblDevicePrice** control
3. To add the currency formatting, set **Text** property as:

```
Text = Text(ThisItem.Price, "[$-en-US]$##,###.00")
```

Localized formula:

```
Text = Text(ThisItem.Price; "[$-en-US]$##,###.00")
```

> **Note:** After you enter the above expression in the formula bar, it will automatically resolve to include your locale, e.g. [$en-US]. If you see an error here, it might be because your local is not yet supported, in which case as a workaround, manually change it to [$-en-US].

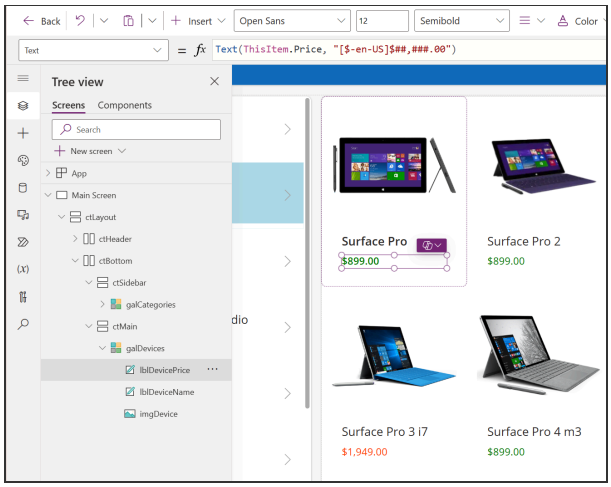
4. To add conditional formatting, set **Color** property as:

```
Color = If(ThisItem.Price>1500,
 Color.OrangeRed,
 Color.Green
)
```

Localized formula:

```
Color = If(ThisItem.Price>1500;
 Color.OrangeRed;
 Color.Green
)
```

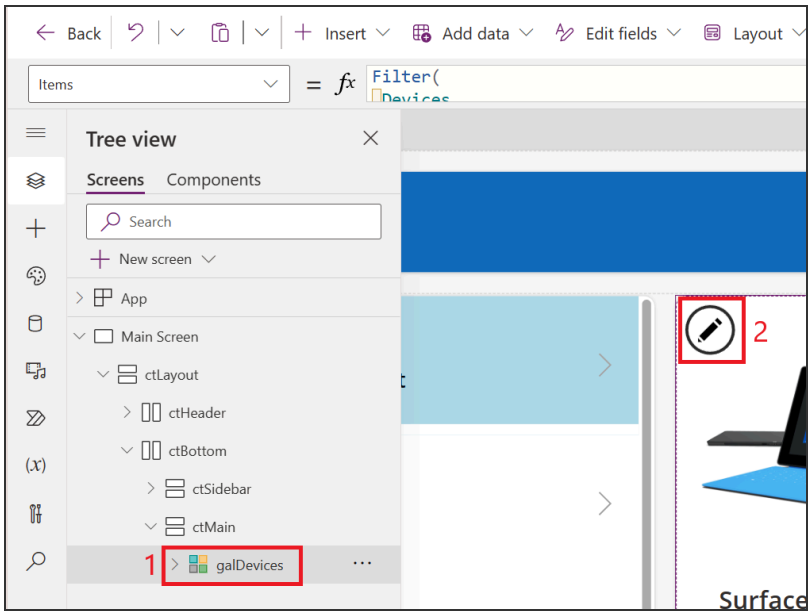
5. **Save** the app, check if it looks similar


Back
Insert
Open Sans
12
Semibold
Color
Text
Text(ThisItem.Price, "[$-en-US]$##,###.00")
Tree view
Screens
Components
Search
New screen
App
Main Screen
ctLayout
ctHeader
ctBottom
ctSidebar
galCategories
ctMain
galDevices
lblDevicePrice
lblDeviceName
imgDevice
Surface Pro
$899.00
Surface Pro 2
$899.00
Surface Pro 3 i7
$1,949.00
Surface Pro 4 m3
$899.00

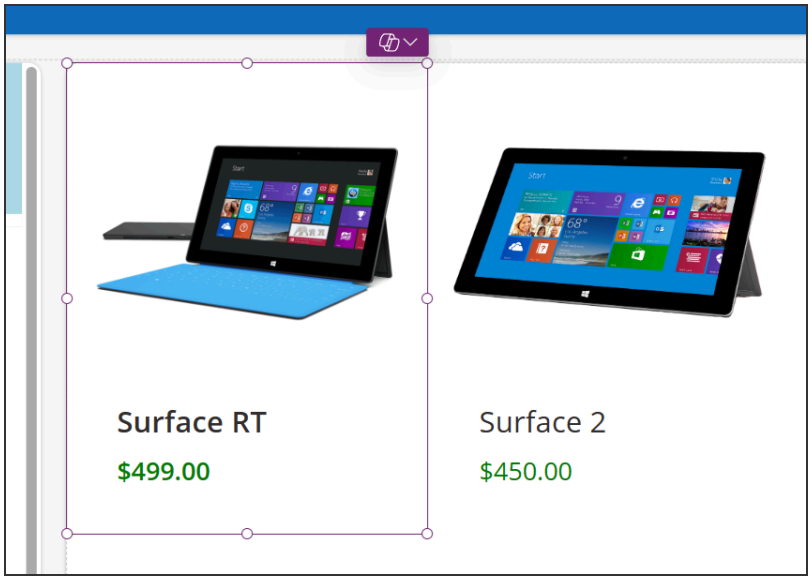
Task 6: Add a checkbox to add a device to Compare list

We want to be able to allow users to compare devices before making an order, in this task you will learn how.

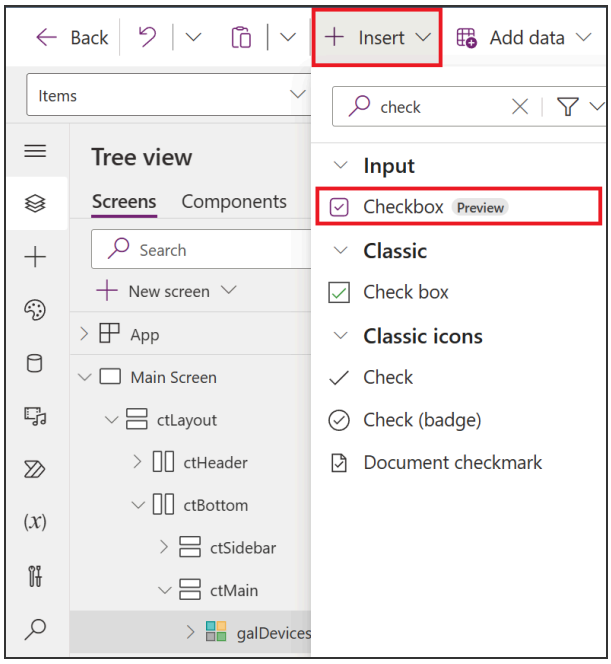
1. Select the **galDevices**
2. Click the **pencil icon** in the top left of the gallery to select the template cell



3. Make sure that only the first item in the gallery is selected



4. Add modern **Checkbox** control using **Insert** pane or button in toolbar



5. Now, let's define coordinates of this checkbox to make it aligned with title and price of the device.

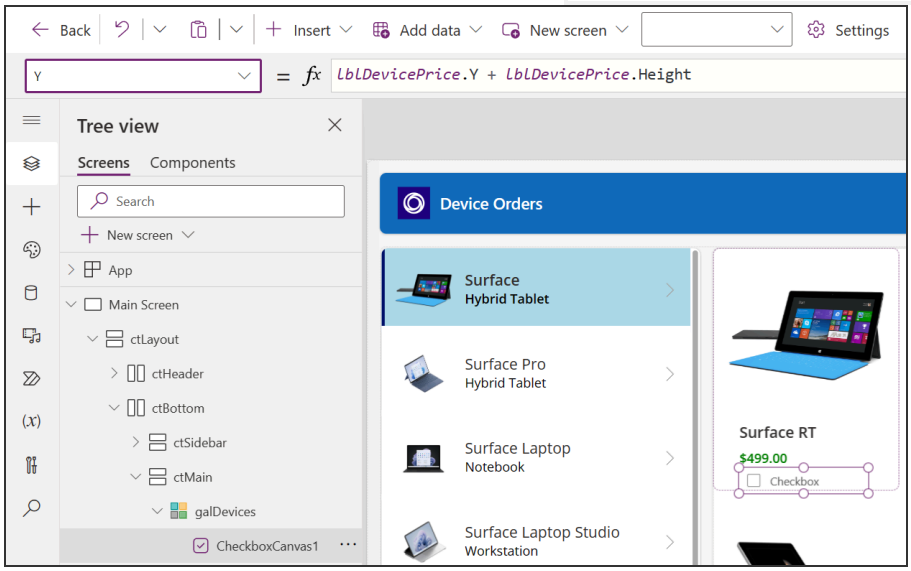
> **Note:** When building an app with responsive UI, containers are a huge help in organizing controls, but not everything can be done automatically. Sometimes you need to manually define relative X/Y position and width/height of controls.
>
> **More details:** Positioning formulas for dynamic layouts

Select the **Checkbox** control and set the **X** property as **lblDevicePrice.X**

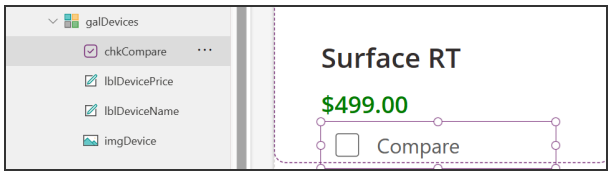
```
X = lblDevicePrice.X
```

Set the **Y** property as **lblDevicePrice.Y + lblDevicePrice.Height**

```
Y = lblDevicePrice.Y+
    lblDevicePrice.Height
```



6. Rename **Checkbox** control to **chkCompare** and change its **Label** property to **"Compare"**


galDevices
chkCompare
lblDevicePrice
lblDeviceName
imgDevice
Surface RT
$499.00
Compare

Task 7: Create a collection for the selected devices

In this task you will create a collection called CompareList. This will allow users to compare devices before deciding to place an order.

> **Note:** What is the collection? A collection is a group of objects that are similar, such as devices in a device list

1. Select previously created **chkCompare** checkbox control
2. Find **OnCheck** property and type the following expression in the formula bar:

   OnCheck = fx `Collect(CompareList, ThisItem)`

   Localized formula:

   OnCheck = fx `Collect(CompareList; ThisItem)`

   This will add a device to the **CompareList** collection when the checkbox is checked.
3. Click the **OnUncheck** value and type the following expression in the formula bar

   OnUncheck = fx `Remove(CompareList, ThisItem)`

   Localized formula:

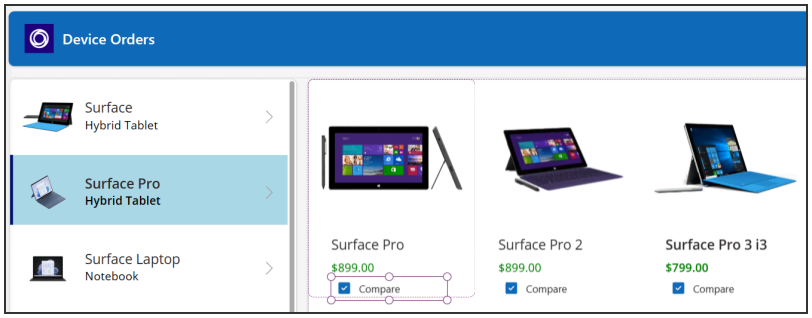
   OnUncheck = fx `Remove(CompareList; ThisItem)`

   This will remove the device from the **CompareList** collection when the checkbox is unchecked.
4. From the properties drop-down, select the **Default** property
5. Set **Checked** property as follows:

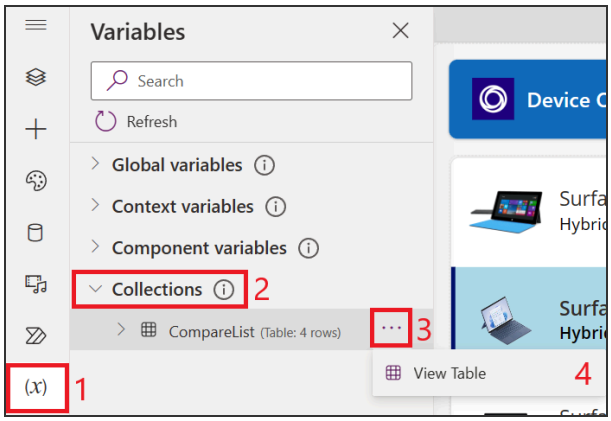
   Checked = fx `ThisItem in CompareList`

   > **Note:** Property **Checked** is a Boolean (true of false) value that determines if the checkbox should be checked or not. Setting it to this formula will ensure that the checkbox is checked by default if the item has already been added to the collection since the result will be true, i.e., this item is in the CompareList collection.

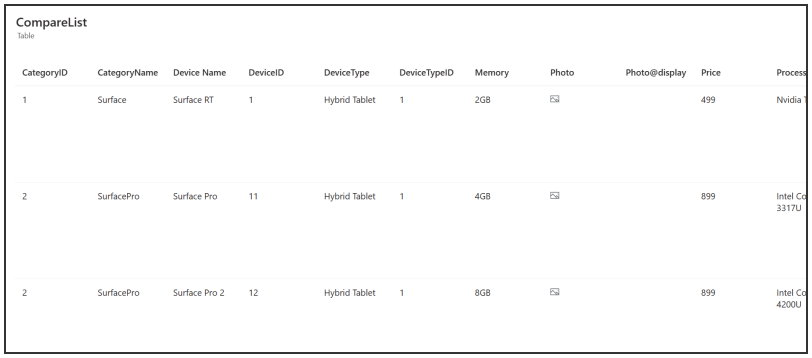
6. Test the checkbox functionality. Run the app in preview mode (F5) or by clicking the preview button, click on checkboxes of three items



7. **Close** preview mode (clicking X)
8. In left navigation pane, open **Variables**(1), then expand **Collections**(2) and view the content of CompareList by clicking ellipsis **"..."**(3) and selecting **View Table**(4)



9. You should see the **CompareList** collection and the three items that you selected



10. Run the app in **preview mode (F5)** or by clicking the preview button, click on checkboxes of three items to uncheck them
11. **Close** preview mode (clicking **X**)
12. Check that all items were removed from the **CompareList** collection

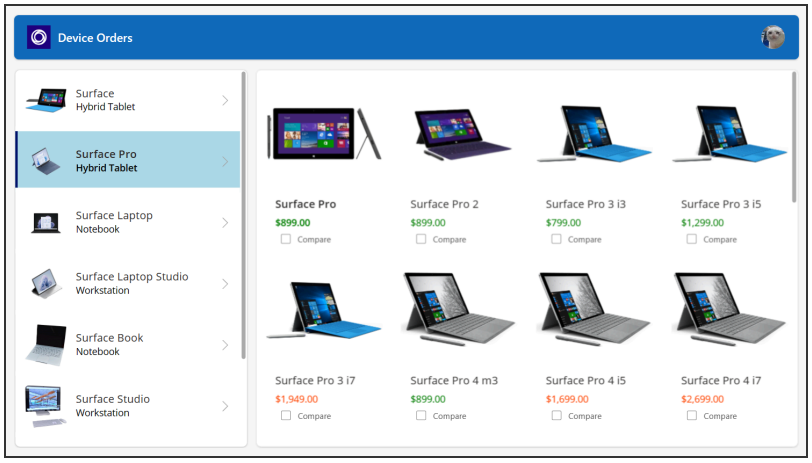
Task 8: Set the default selection to the first category

In this task, you will learn how to ensure that the app doesn't display a blank list of devices when the app starts by setting the default item in the Categories gallery to be the first item.

1. **Select** the **galCategories** in the tree view in the left pane
2. From the **properties** drop-down, select the **Default** property
3. Enter the following **expression** in the formula bar

```
Default = First(Categories)
```

4. Run the app in **preview mode** (F5)
5. The app should look like the image below



6. **Close** preview mode (clicking X)
7. **Save** the app

Exercise 4: Add a compare screen

The compare screen is where users will compare the selected devices and then choose the device that they wish to submit for approval.

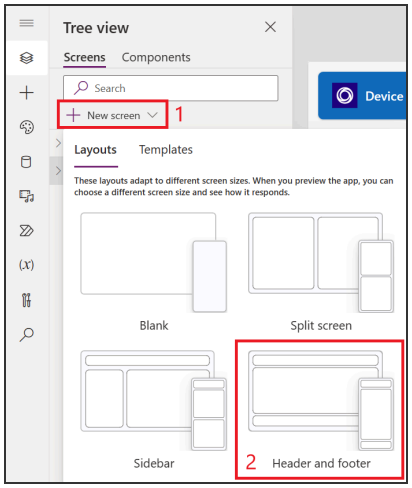
This screen will include:

- Navigation back to the Main Screen
- A list of selected devices for comparison
- Additional details for each device
- Highlighting the selected device

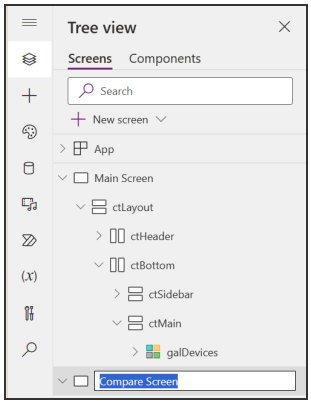
Task 1: Add a screen

In this task you will add the Compare Screen to the app.

1. Click **New screen**
2. Select **Header and footer**



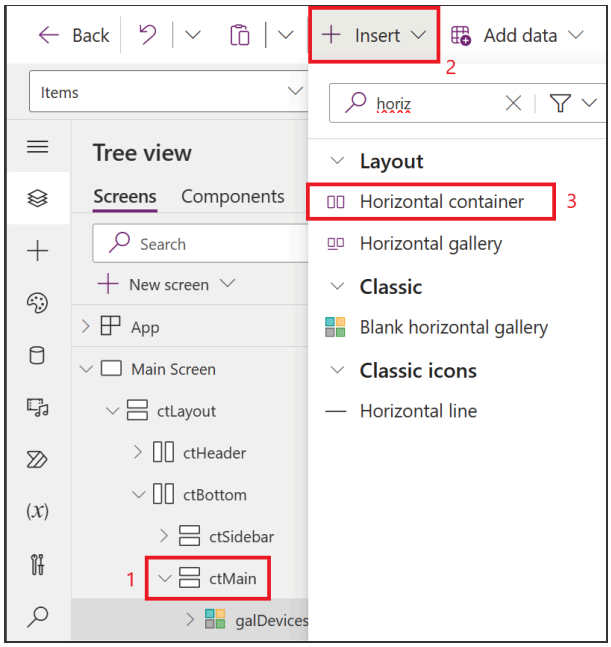
3. Rename the screen to **Compare Screen**



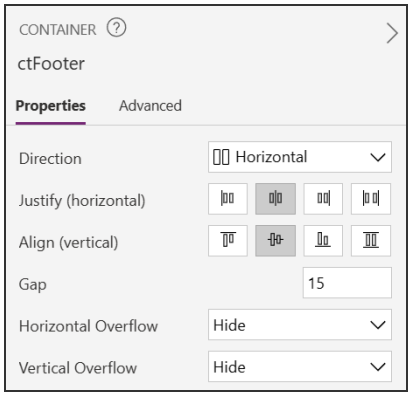
Task 2: Add Compare and Clear buttons to Main Screen

In this task you will learn how to add buttons to the Main Screen that add functionality to the app to navigate to the Compare Screen and clear the selected devices for comparison

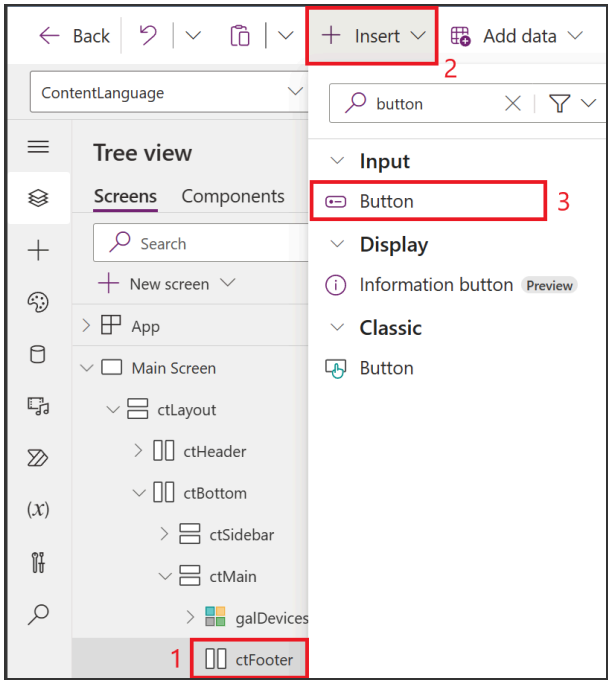
1. In the Tree View, switch to **Main Screen**
2. Select **ctMain** container, click the Insert button and add **Horizontal** container



3. Rename it as **ctFooter**
4. Set ctFooter properties as:
   a. Justify (horizontal) -> **Center**
   b. Align (vertical) -> **Center**
   c. Gap -> **15**
   d. Wrap -> **Off**
   e. Flexible Height -> **Off**
   f. Height -> **50**



5. Select **ctFooter** and insert modern **Button** control



6. For the button, in formula bar set **Text** property as:

```
Text = fx "Compare " &
CountRows(CompareList) &
" item(s)"
```

7. Resize the button so the text fits without wrapping (e.g., Width -> 200)
8. Select the button and set the **DisplayMode** property expression to:

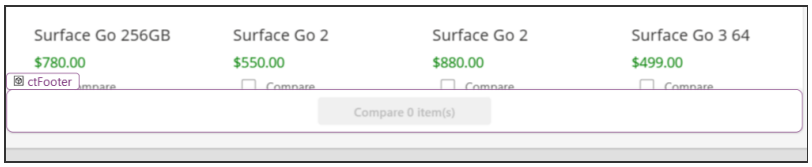
```
DisplayMode = fx If(CountRows(CompareList) > 0,
 DisplayMode.Edit,
 DisplayMode.Disabled
)
```

Localized formula:

```
DisplayMode = fx If(CountRows(CompareList) > 0,
 DisplayMode.Edit;
 DisplayMode.Disabled
)
```

9. Run the app in **preview mode** (F5)
10. **Select** a few devices
11. Check that the button displays the number of devices selected for comparison
12. **Close** preview mode (clicking X)

13. Run the app in **preview mode** (F5)
14. **Unselect** the devices
15. Check that the button is **disabled**



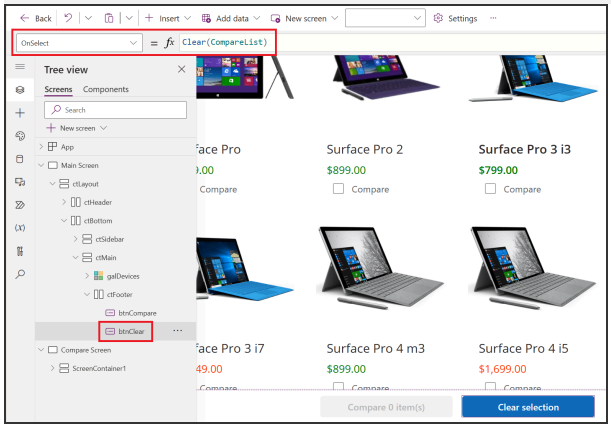
16. **Close** preview mode (clicking X)
17. In **Tree View**, rename this button as **btnCompare**
18. Insert one more modern **Button** control to **ctFooter**
19. Change the **Text** property to **"Clear selection"**
    Change the **Width** property to **200**
20. In **Tree View**, rename this button as **btnClear**
21. Set the **OnSelect** property of the button to the following expression:

```
OnSelect = fx Clear(CompareList)
```

> **Note:** This will remove all the items in the CompareList collection when the user clicks the button

22. Select the **btnCompare** button
    For **btnCompare**, set **OnSelect** property as:

```
OnSelect = fx Navigate('Compare Screen')
```

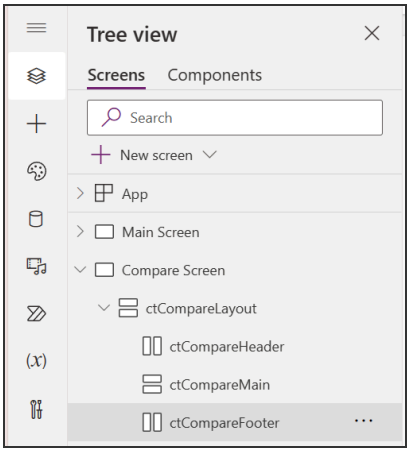


23. Click **Preview** and select a couple of devices and click the Compare button
24. **Verify** that it takes you to the Compare Screen, then close Preview (clicking X)

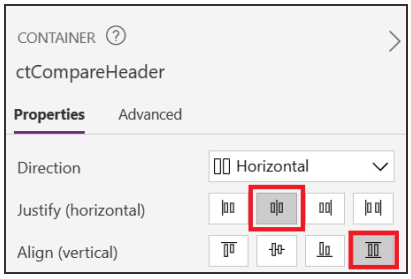
Task 3: Add controls to the Compare Screen

In this task, you will copy the various controls that you want to display on both the Main Screen and Compare Screen.

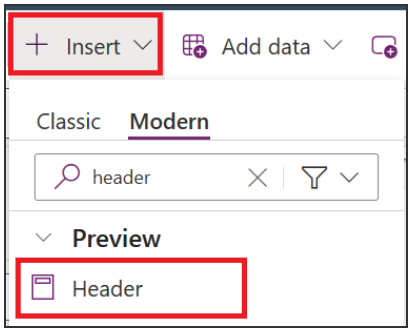
1. In **Tree View** pane, navigate to **Compare Screen**
2. In **Tree View** pane, **Rename** containers according to naming conventions:
   - ScreenContainer1 -> ctCompareLayout
   - HeaderContainer1 -> ctCompareHeader
   - MainContainer1 -> ctCompareMain
   - FooterContainer1 -> ctCompareFooter



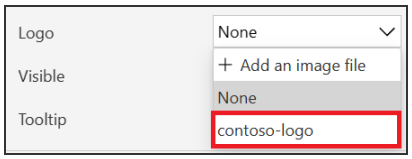
3. Select **ctCompareHeader** container
   a. Justify (horizontal) -> **Center**
   b. Align (vertical) -> **Stretch**



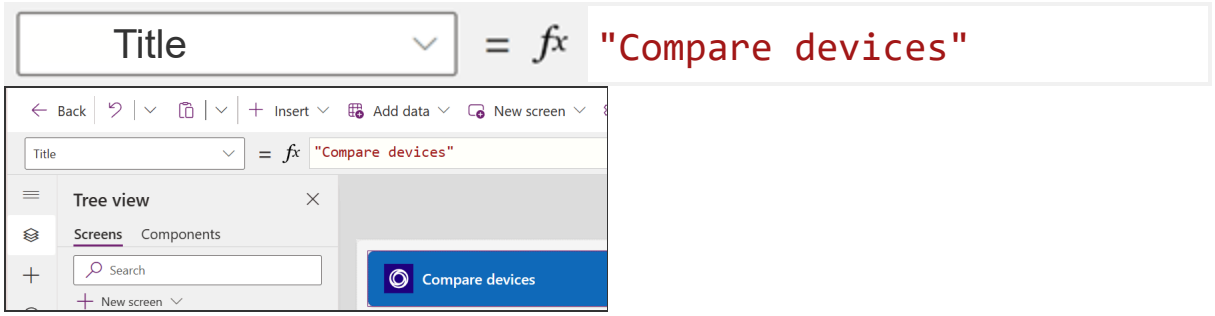
4. Select the **Insert** button in the toolbar, navigate to **Modern** controls, and select **Header**



5. In Properties pane, find **Logo** property and select *contoso-logo.png* that was uploaded before.



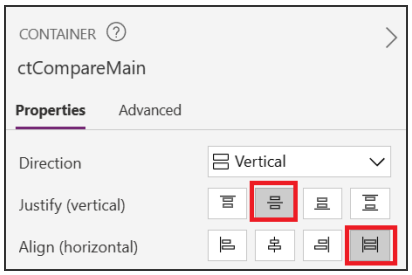
6. Set **Title** property to **Compare devices**



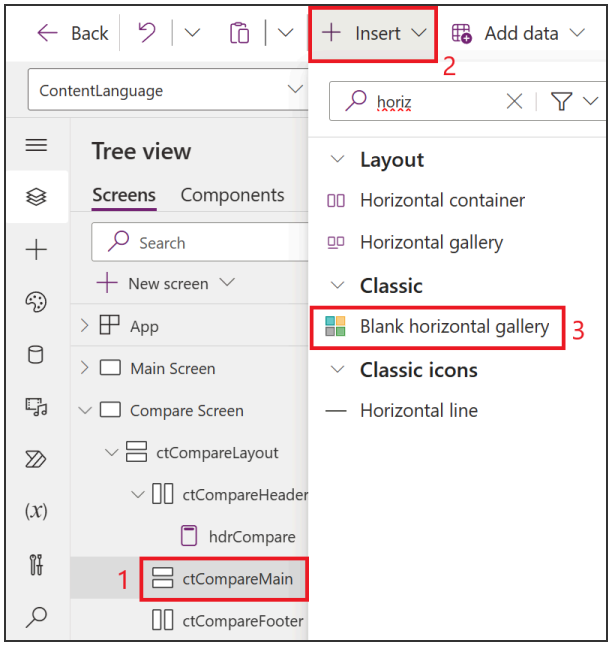
7. Change the **Header** control name to **hdrCompare**
8. Select **ctCompareMain** container and set its properties:

   a. Justify (vertical) -> **Center**

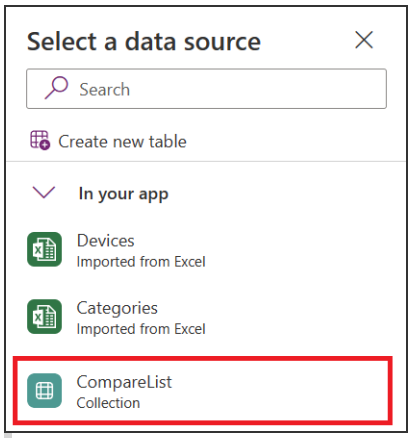
   b. Align (horizontal) -> **Stretch**



9. **Insert** classic **Blank horizontal gallery** control to **ctCompareMain**

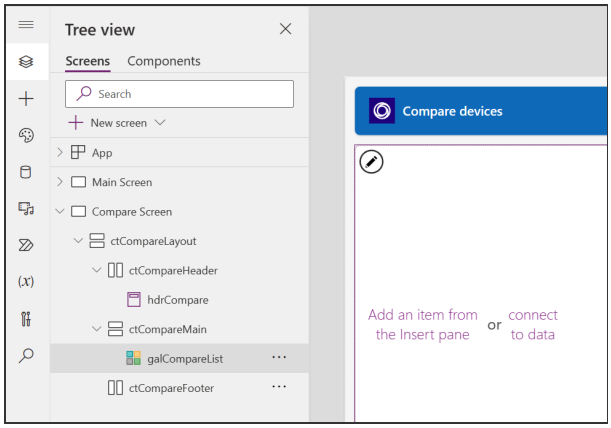


10. Select **CompareList** as a data source for this gallery



> **Note:** If you accidentally selected not BLANK horizontal gallery, you can change Layout property to Blank and set WrapCount to 1

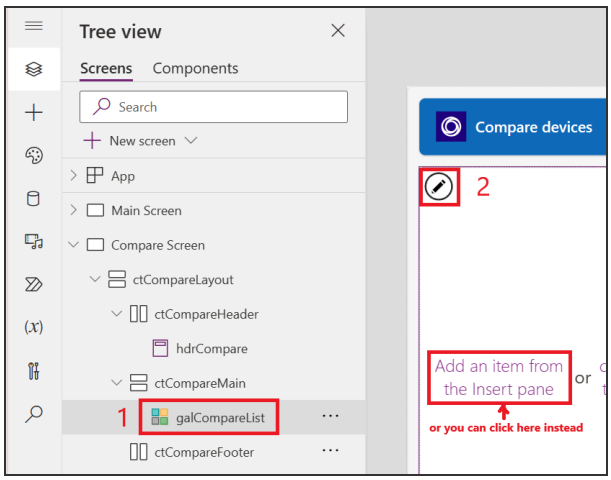
11. Rename this Gallery to **galCompareList**



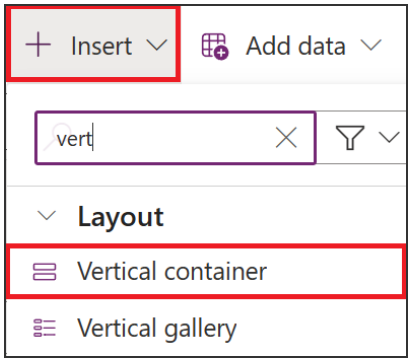
Task 4: Configure galCompareList gallery

In this task, we will use technique of adding containters into the gallery template to organize the layout of the gallery items.

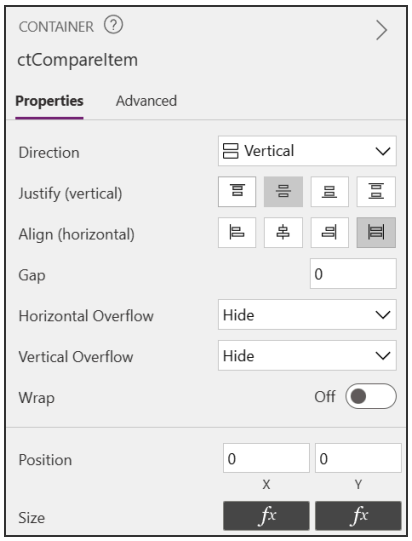
1. Select **galCompareList** and click on **pencil icon** in the top-left corner to select the template cell



2. Add **Vertical** container to the template cell



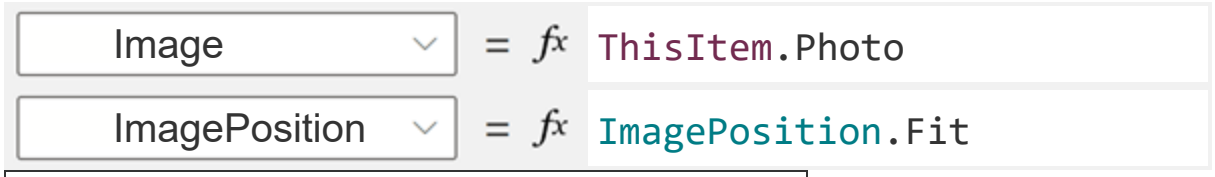
3. Rename this container as **ctCompareItem**
4. Set **ctCompareItem** container properties as:
   a. Justify (vertical) -> **Center**
   b. Align (horizontal) -> **Stretch**
   c. Gap -> **5**
   d. X -> **0**
   e. Y -> **0**
   f. Height -> **Parent.TemplateHeight**
   g. Width -> **Parent.TemplateWidth**

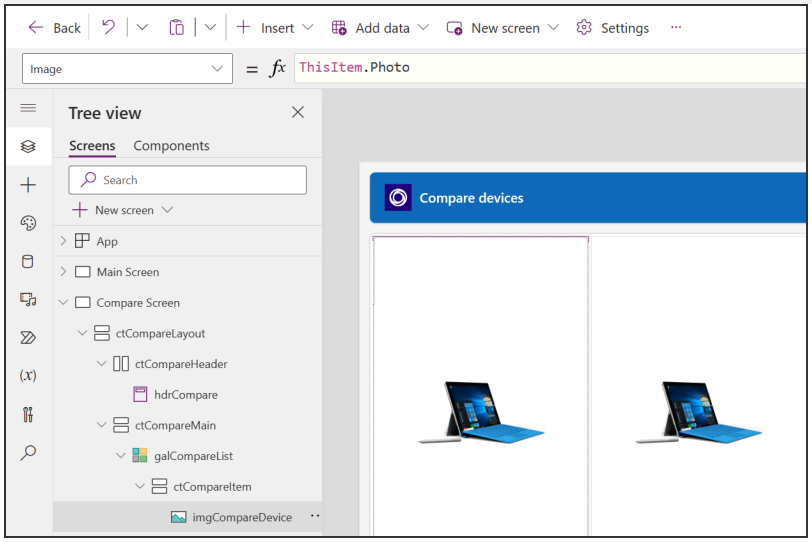


5. Select **ctCompareItem** and insert **Image** control

   Rename it as **imgCompareDevice**

6. Set **Image** property as **ThisItem.Photo** and **ImagePosition** as **ImagePosition.Fit**

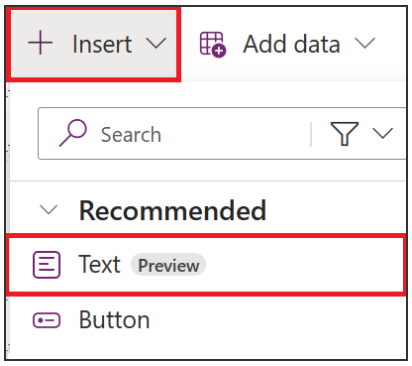




   If you don't see the images, make sure you've marked some checkboxes on **Main Screen** for device comparison.

7. Select **ctCompareItem** and insert modern **Text** control

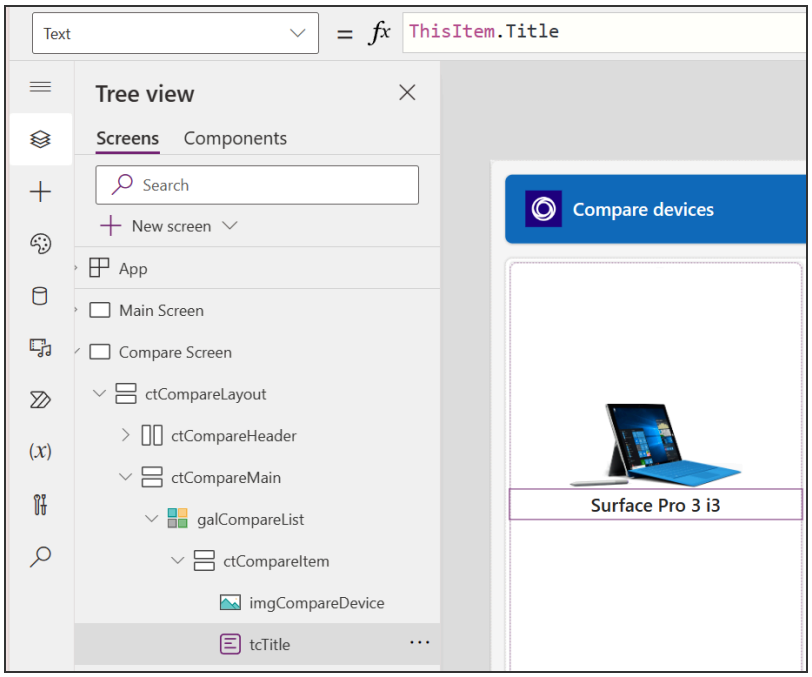
   > **Note:** You will need to repeat this and a few following steps multiple times later in this lab.



8. Rename it as **tcTitle**
9. Select **tcTitle** and set **Text** property as **ThisItem.Title**

10. Adjust formatting of the Text control to make it look better.

    For example, set **Weight** to **'TextCanvas.Weight'.Bold** and **Size** to **18**, and align text to the center.



11. Add and repeat the process for a few more Text controls:

Control name	Text property
tcPrice	Text(ThisItem.Price, "[$-en-US]$##,###.00")
tcProcessor	ThisItem.Processor
tcMemory	ThisItem.Memory
tcStorage	ThisItem.Storage
tcScreenSize	ThisItem.ScreenSize

12. **Insert** a modern **Button** control to **ctCompareItem** container, and rename it as **btnRemove**

13. For this button, set **Text** property as **"Remove from comparison"**

```
Text = "Remove from comparison"
```

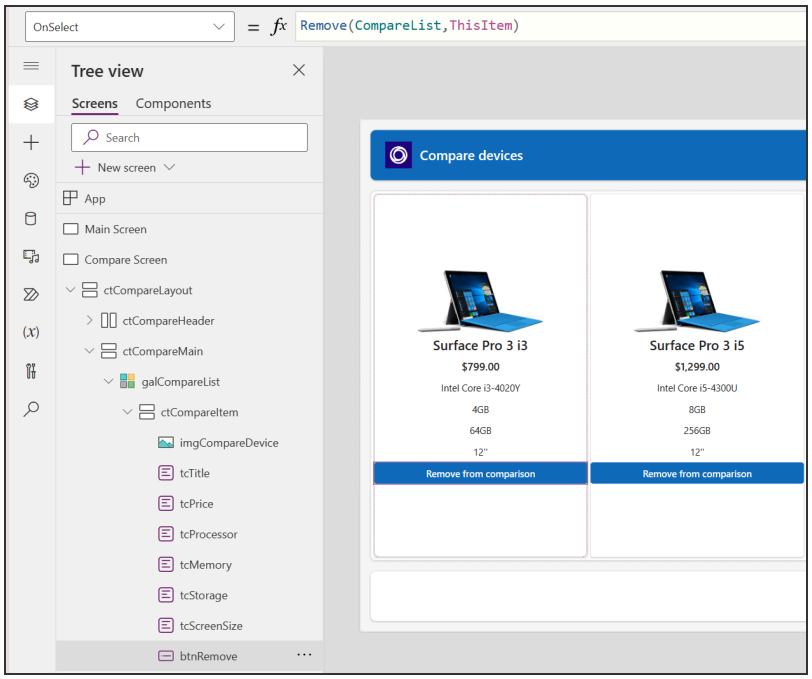
14. Set **OnSelect** property as **Remove(CompareList, ThisItem)**

```
OnSelect = Remove(CompareList, ThisItem)
```

Localized formula:

```
OnSelect = Remove(CompareList; ThisItem)
```

15. Your gallery should look like this now



Task 5: Highlight the selected device

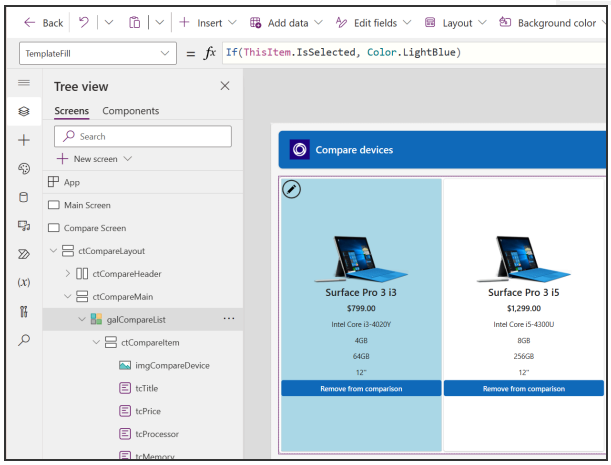
In this task you will recreate the behavior of the selected category in the Main Screen. When a user clicks on a device, you will set the template fill to show that it is highlighted.

1. Select the **galCompareList**
2. From the properties drop-down, select the **TemplateFill** property
3. In the formula bar, enter the following expression:

```
TemplateFill = fx  If(ThisItem.IsSelected,
                   Color.LightBlue)
```

Localized formula:

```
TemplateFill = fx  If(ThisItem.IsSelected;
                   Color.LightBlue)
```

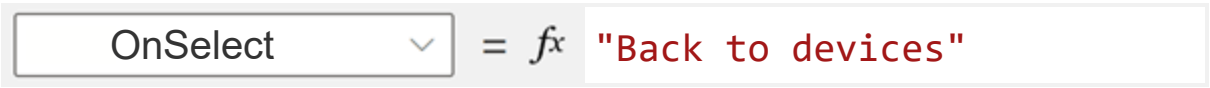


This is conditionally setting the Fill color if the cell is selected. Click a few different items in the gallery, notice the selected item is highlighted in a light blue color.

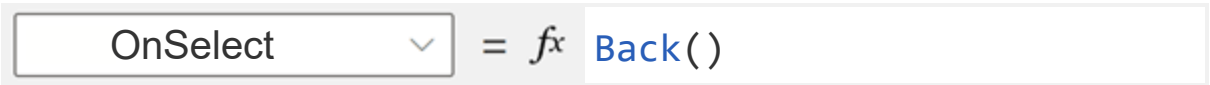
Task 6: Configure galCompareList gallery

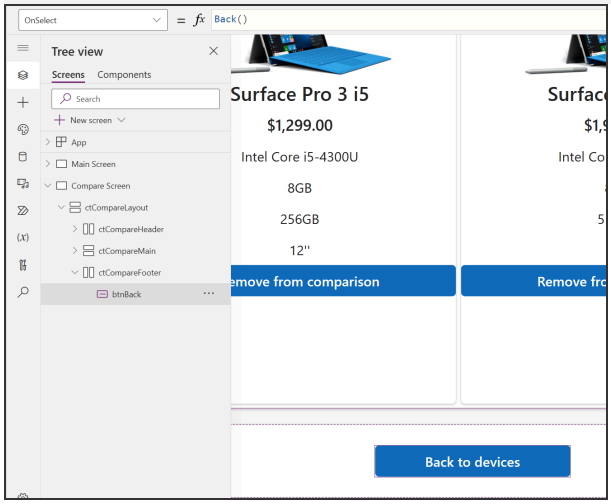
In this task you will learn how to add a button to navigate back to the Main Screen.

1. On **Compare Screen** select **ctCompareFooter** container
2. Set **ctCompareFooter** properties as:
   a. Justify (horizontal) -> **Center**
   b. Align (vertical) -> **Center**
   c. Gap -> **15**
   d. Wrap -> **On**
3. Click **Insert** in toolbar and add modern **Button** control to **ctCompareFooter** container
4. Rename this button as **btnBack**, change its **Width** to **200**
5. Set **Text** property as **"Back to devices"**



6. From the properties drop-down select the **OnSelect** property and in the formula bar enter the following expression:





> **Note:** This will cause navigation back to the previous screen.

Exercise 5: Ordering a device for approval

So far, you have built the app. Users can browse and compare devices. In this exercise you will add additional functionality so that users can create an order/ submit a device for approval. In this lab, device orders will be managed in SharePoint list (created as part of the prerequisites). Power Automate flow will be created in subsequent exercises to manage the approval process.

Task 1: Add an order button

In this task, you will add an order button to the **galCompareList** template cell.

1. Select the **ctCompareFooter**
2. Click the **Insert** button in the toolbar and add a modern **Button** control to **ctCompareFooter**
3. Rename the button to **btnPlaceOrder**
4. From the properties drop-down select the **Text** property and change it from "Button" to **"Order"**:

   Text = fx `"Order"`
5. Button should look like this:

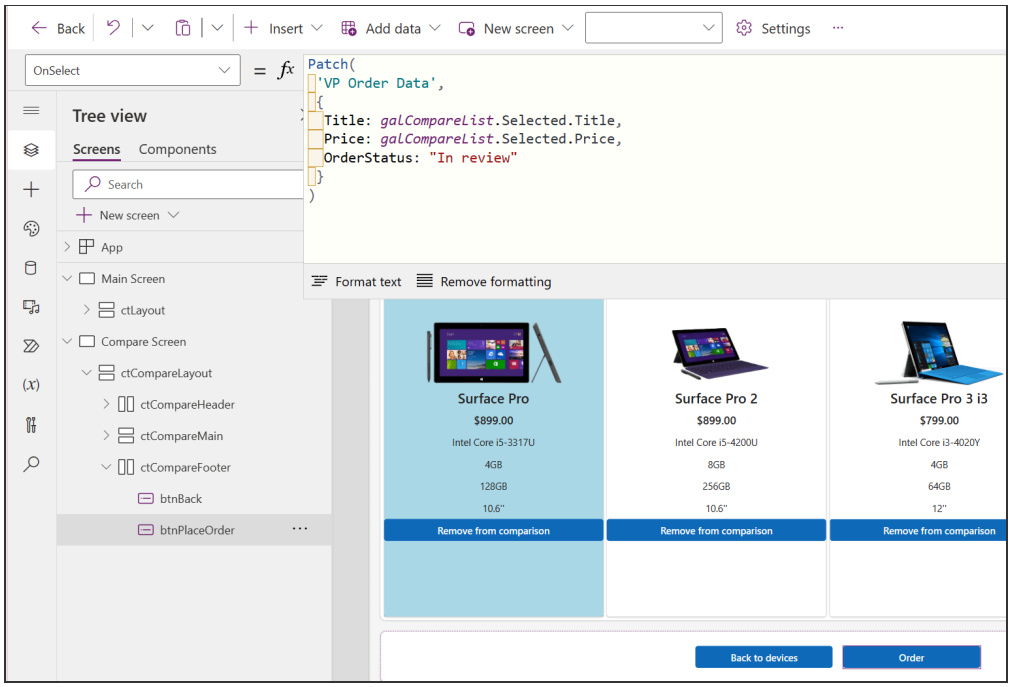
   

6. Select **OnSelect** property and in the formula bar, enter the following expression:

   > **Note:** Patch() function creates or modifies a record. In the expression below, we are creating a new record in the SharePoint list.

   OnSelect = fx

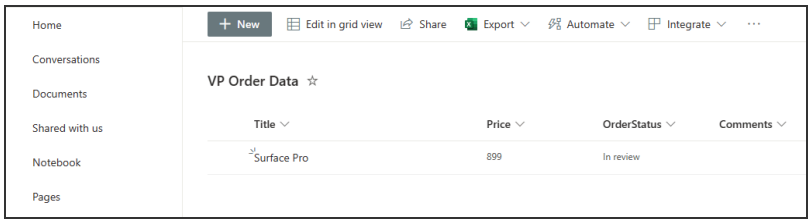
```
Patch(
 '<YOUR INITIALS> Order Data',
 {
  Title: galCompareList.Selected.Title,
  Price: galCompareList.Selected.Price,
  OrderStatus: "In review"
 }
)
```



Localized formula:


```
Patch(
 '<YOUR INITIALS> Order Data';
 {
  Title:
galCompareList.Selected.Title,
  Price:
galCompareList.Selected.Price,
  OrderStatus: "In review"
 }
)
```

7. Test the app. Enter **preview mode** (F5) and click the **Order** button for one of the laptops. Check that the order information was created in SharePoint list:



Task 2: Notifying the user

In this task, when an order has been placed, you will create a notification to communicate that the order has been placed to the user.

1. Select the button **btnPlaceOrder**
2. From the properties drop-down, select **OnSelect**
3. At the end of the expression (created in task one) add a chaining operator: semicolon '**;**'

   > **Note:** In alternate/European locales: chaining operator is double semicolon "**;;**"

4. Press Shift+Enter
5. On the new line, append the following expression:

   OnSelect = fx

   ```
   Patch({...});
   Notify("Thank you for placing an order",
   NotificationType.Success)
   ```

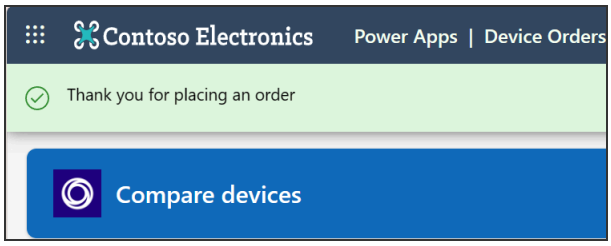
   Localized formula:

   OnSelect = fx

   ```
   Patch({...});;
   Notify("Thank you for placing an order";
   NotificationType.Success)
   ```

6. Test the app, enter **preview mode** (F5), and click the **Order** button.

   You should see a toast notification shown in the image below:

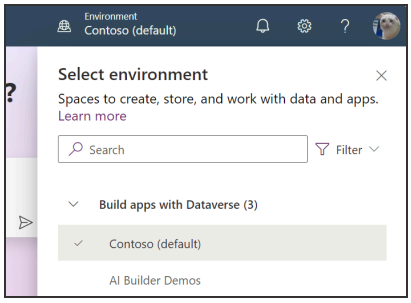


Exercise 6: Creating an approval flow

In this exercise, you will be introduced to Microsoft Power Automate.
You will learn how to build a basic approval flow for the device ordering process.

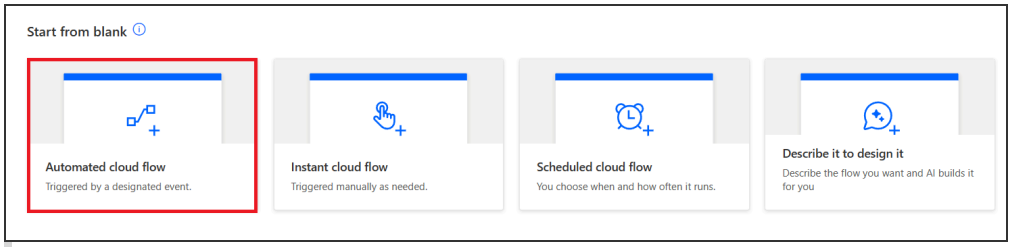
Task 1: Create a new flow

1. In the browser, go to https://make.powerautomate.com/
2. In the top right of the screen, click **Sign In**
3. Before creating a flow, switch to the correct environment (where your organization permits development apps to be created).
   Click the **Environments** drop-down in the top right of the screen to switch to the correct environment.
   Ask the instructor if you have any questions.



4. In the left navigation, click **Create**

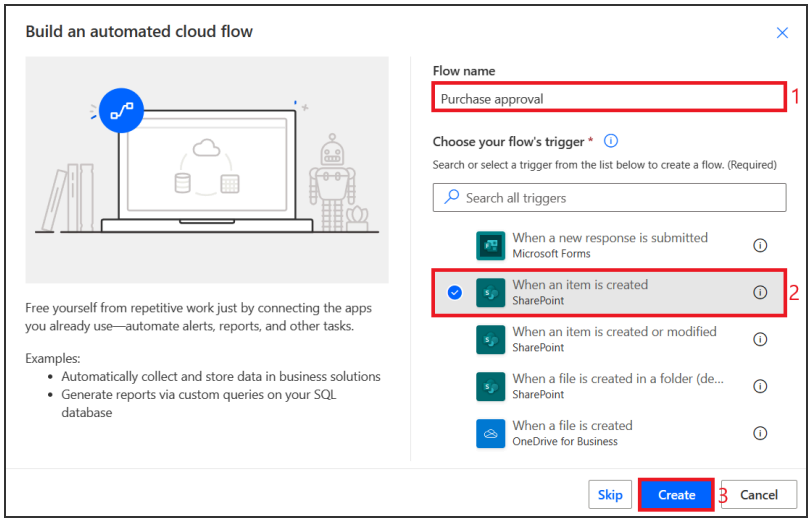
5. In the Create screen, select the **Automated cloud flow** option



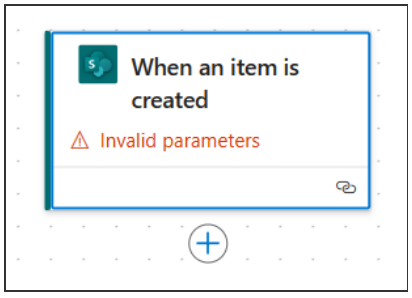
> **Note:** Flows are started with a 'Trigger'.
> There are three basic triggers to start a flow:

> - **Automated**: designated events will start the Flow; for example: a new item created in a SharePoint list or a new record in Dataverse table
> - **Instant**: Flows are started manually by the user
> - **Scheduled**: Flows start at a frequency specified; for example: every day at 10:00 hours

6. In the *'Build an automate cloud flow'* dialog, enter **Purchase approval** in the Flow name box select **'When an item is created'** (SharePoint) and, finally, click **Create**

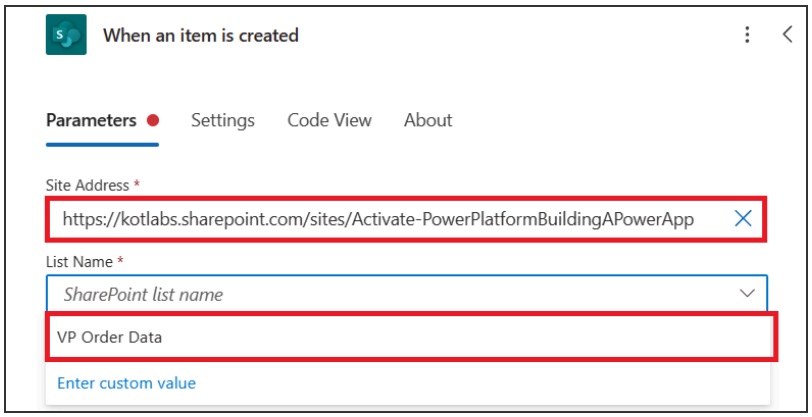


7. When flow designer experience opens, click on the trigger to set its parameters



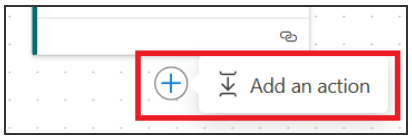
8. In side pane, click in the **Site address** input box, select **Enter custom value** and **type/ paste the URL of the SharePoint site** that you created to store order data.

   Then **List name** drop down, select the **'<your initials> Order Data'** list.

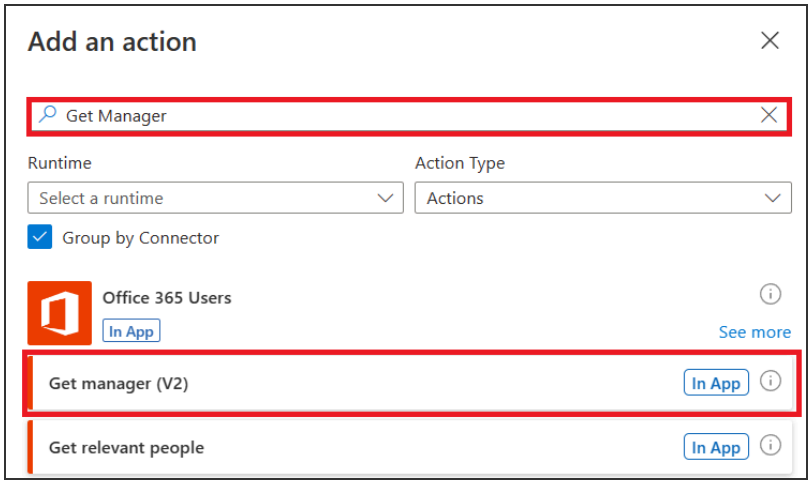


Task 2: Lookup the requestors line manager

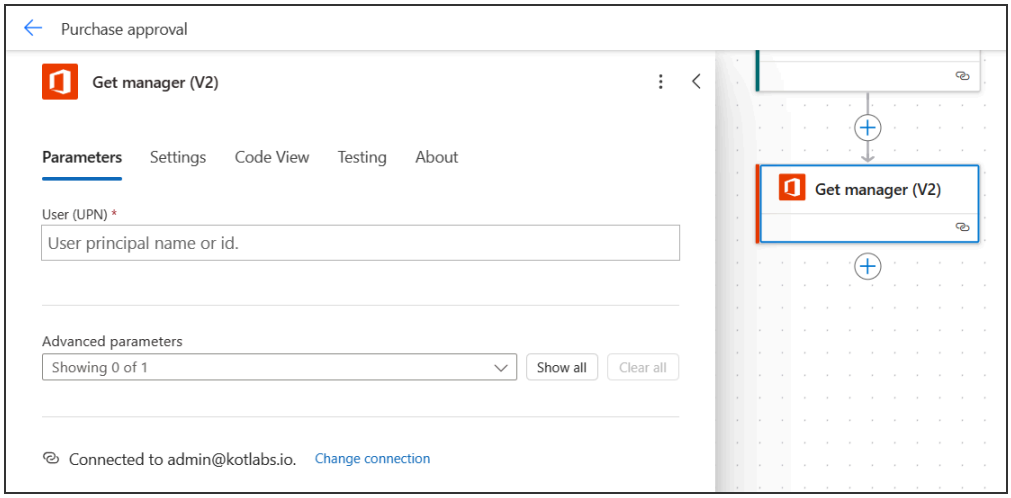
1. Click + icon below the trigger and select **Add an action**



2. In **Add an action** side pane, enter **Get manager** in the search box. Then select **Get manager (v2)** action from **Office 365 Users** connector.

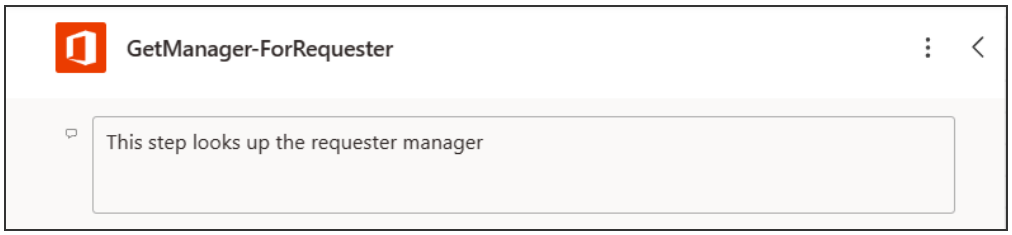


Action will be added right after the trigger, and Parameters pane will open on the left side.

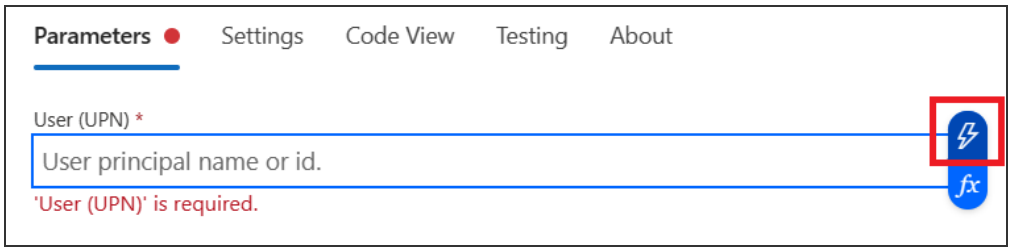


3. Click on Get Manager (V2) title and rename action as **GetManager-ForRequester**
4. Click the **ellipsis (…)**, then select **'Add a note'**. Enter **'This step looks up the requester manager'**

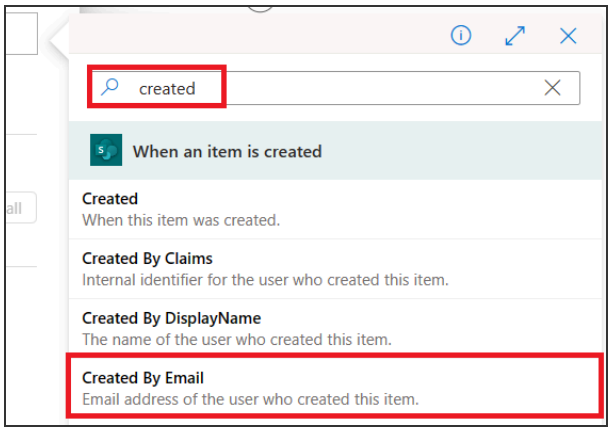
> **Note:** Renaming actions and providing notes is good practice as it ensures that co-owners of the flow understand the context and purpose of specific actions, especially in complex flows. When renaming action, please, preserve original action name as a prefix.
> We will use this action to show the example of renaming and adding notes.



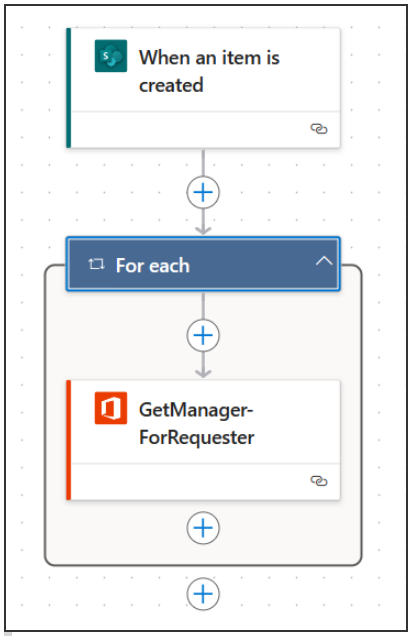
5. Click inside the **User (UPN)** parameter box and select **'Add a dynamic content'** (lightning) icon on the right



6. Search for **Created by Email** property retrieved from SharePoint trigger (click on **See more** if you don't see it)

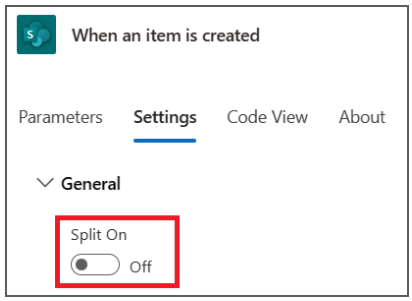


7. The flow should now look like the screenshot below



> **Note: For each** loop is automatically added as trigger can return multiple new items from the list. This behavior can be controlled using trigger's **Split on** setting but we do not need to

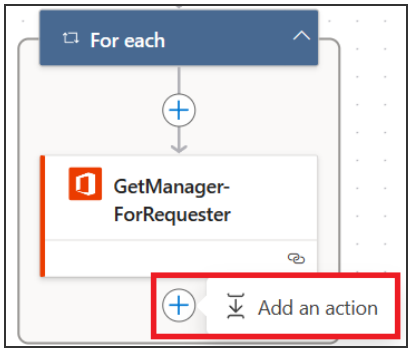
> change that now. We need to create an approval for each new request, that's why we will continue adding actions inside the loop.



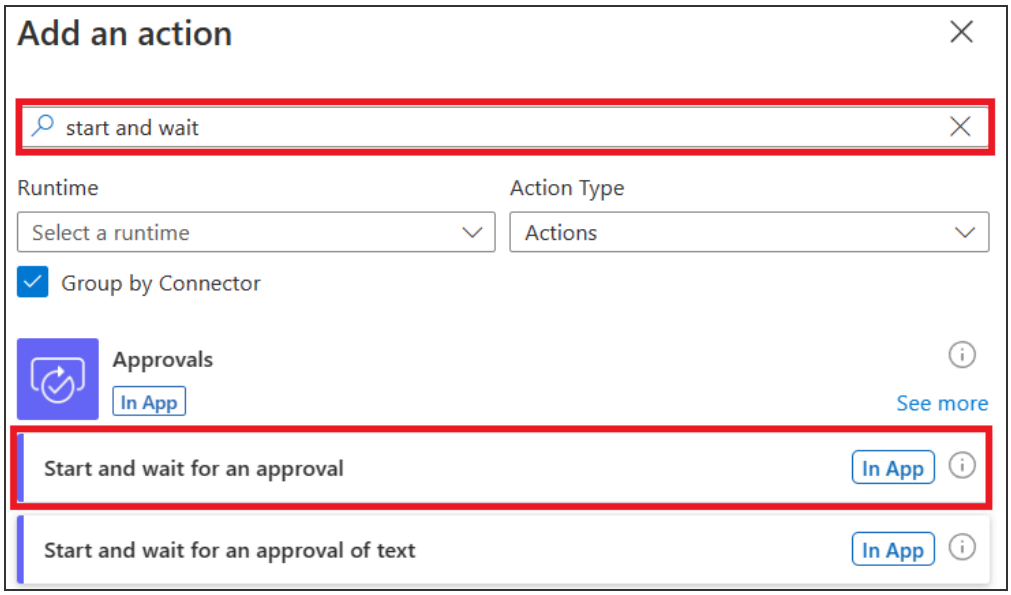
Task 3: Start and wait for approval

In this task you will learn how to use the Approvals connector.

1. Right after **GetManager-ForRequester** action, click **+** icon and **Add an action**



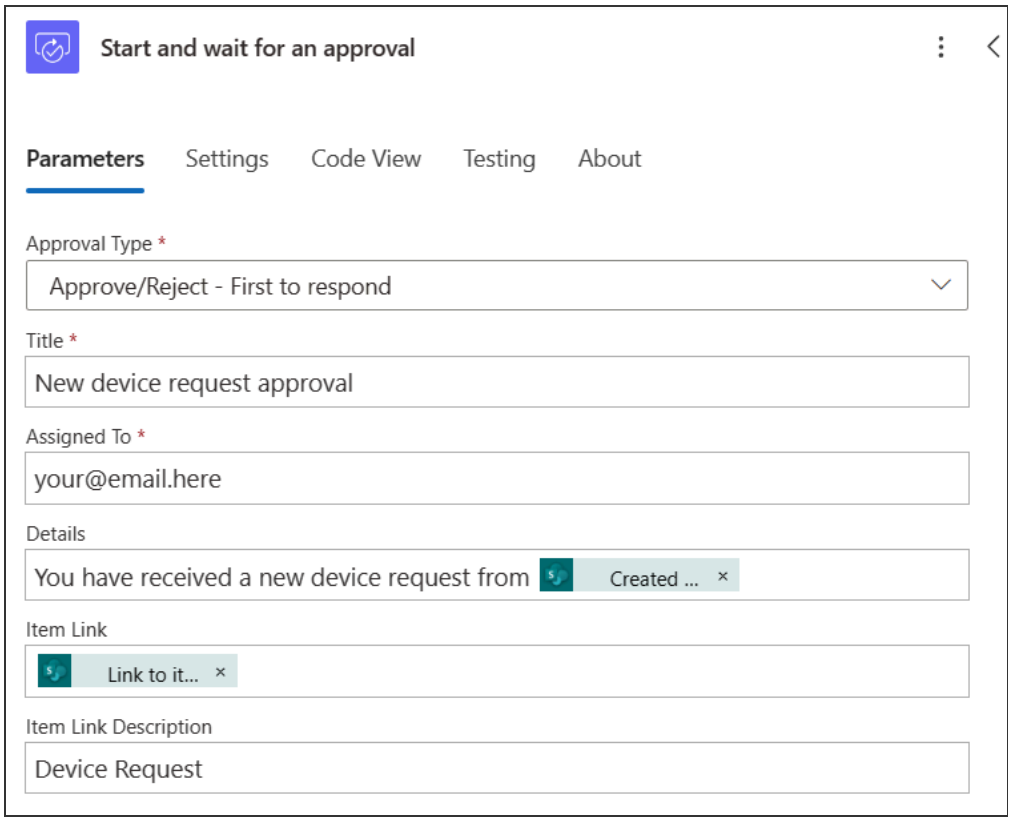
2. In Add an action pane's search box, type **Start and wait for approval** and then select corresponding action below



3. In the Approval Type drop-down, select 'Approve/Reject – First to respond'
4. Set up other parameters as below:
   - **Title**: New device request approval
   - **Assigned to**: Enter **YOUR OWN** email

     > **Note:** In real-life scenario, you would assign the approval to the mail of the manager from the previous step.

   - **Details**: You have received a new device request from (select 'Created By DisplayName' from the When an item is created trigger in dynamic content)
   - **Item link**: Select Link to item from the 'When an item is created' trigger
   - **Item link description**: Enter any link description text
5. **Start and wait for approval** action parameters should look the same as the screenshot below:

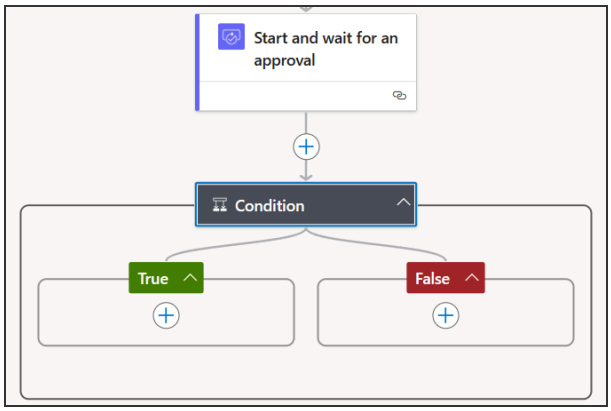


> **Note:** Consider removing previous **GetManager-ForRequester** action to avoid getting errors if Manager field is not defined in profile.

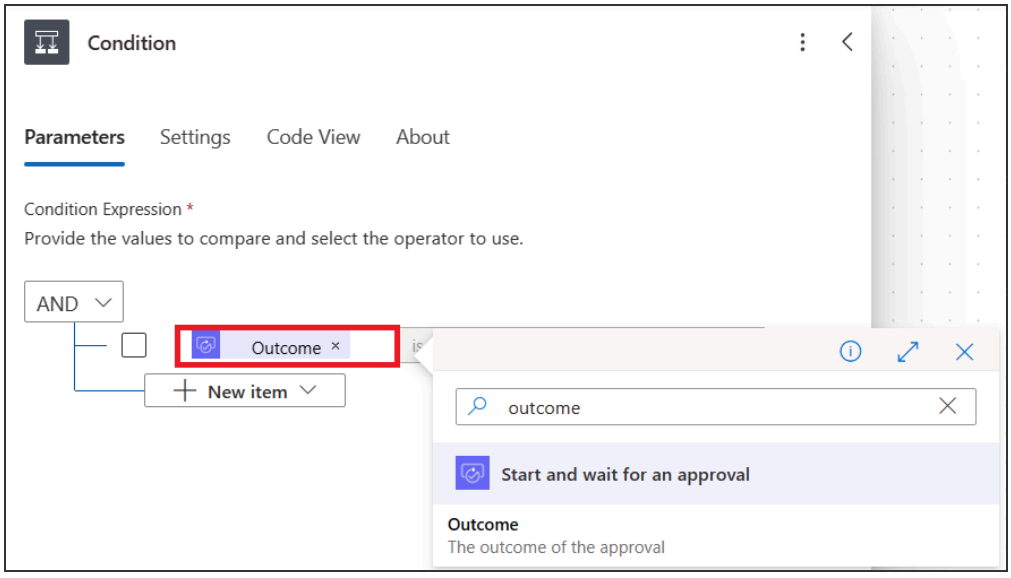
Task 4: Add a condition

In this task you will learn how to add in a condition. A condition action handles specific conditions in your Flow, such as 'If a property equals a value', or 'If a property is greater than a value'.

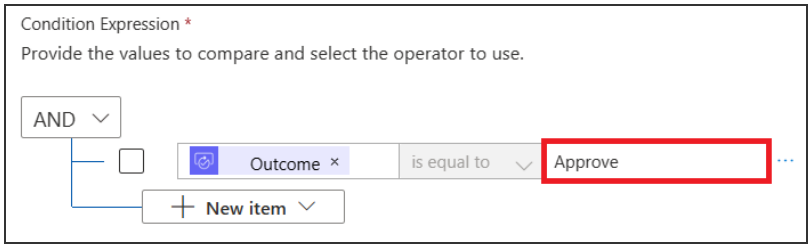
1. Right after **Start and wait for approval** action, click **+** icon and **Add an action**
2. In the actions window, search and select '**Condition**' from Control connector



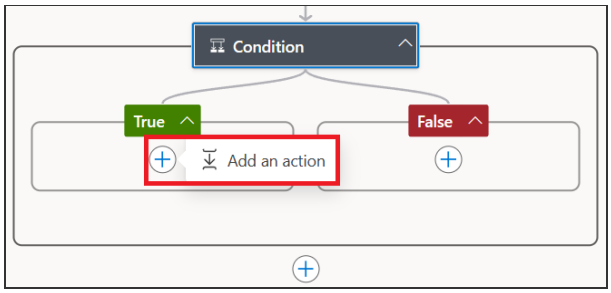
3. Let's set parameters for Condition.
   In the left side of condition expression select **'Outcome'** from dynamic content of **Start and wait for approval** action



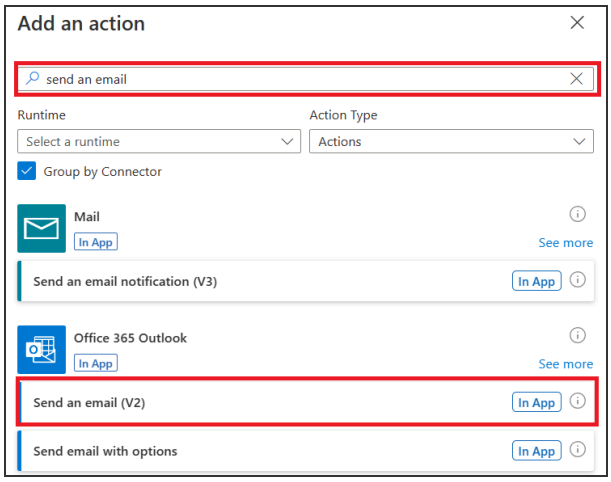
   Check that **'is equal to'** is selected, and in the right side of expression type *Approve* (it's just a plain text)



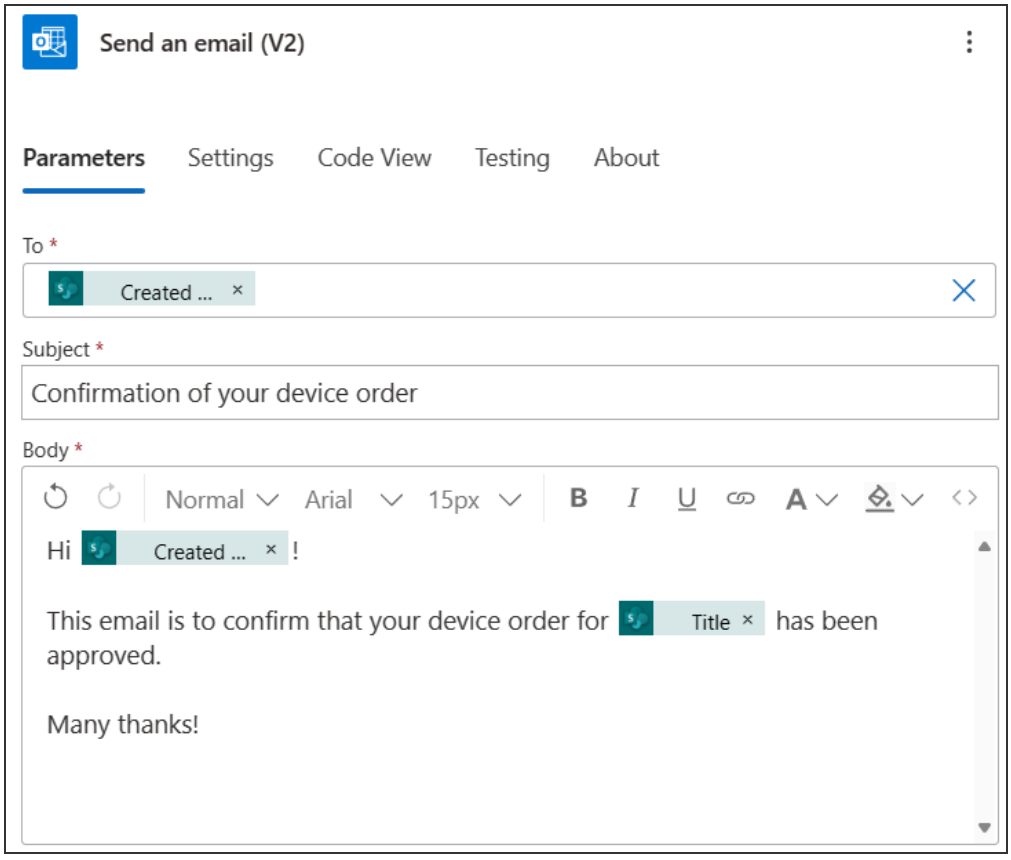
4. In the **'True'** branch of the condition, click **Add an action**



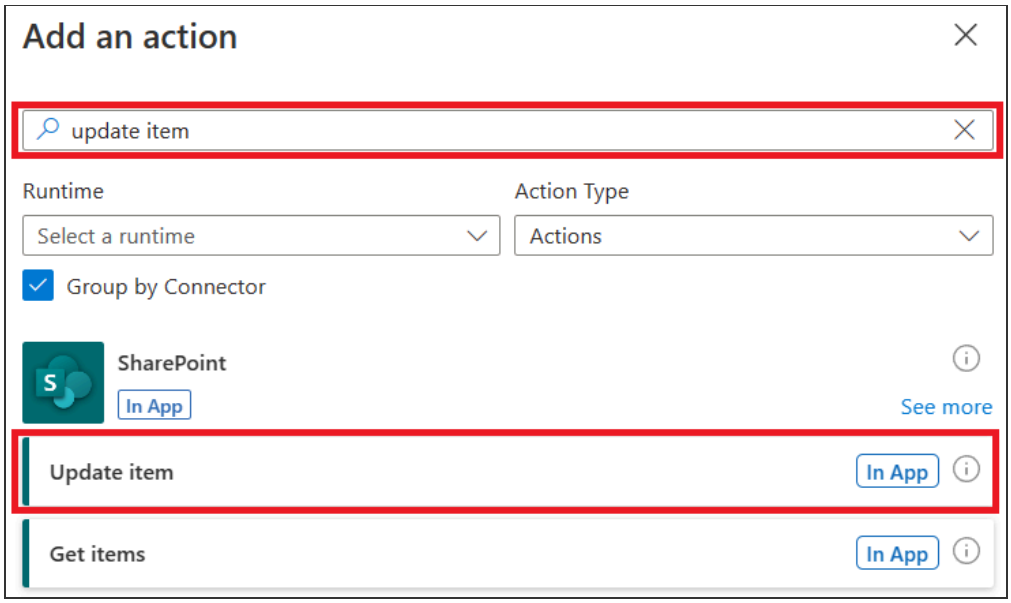
5. In the actions window, search and select **'Send an email (V2)'** from Office 365 Outloock connector



6. Set **Send an Email (V2)** parameters as below:
   - **To**: 'Created by Email' (from dynamic content of trigger, select **Enter custom value** to see it)
   - **Subject**: *Confirmation of your device order* (plain text)
   - **Body**: *Hi* **'Created by DisplayName'***! This email is to confirm that your device order for* **'Title'** *has been approved. Many thanks!* (use dynamic content from trigger)

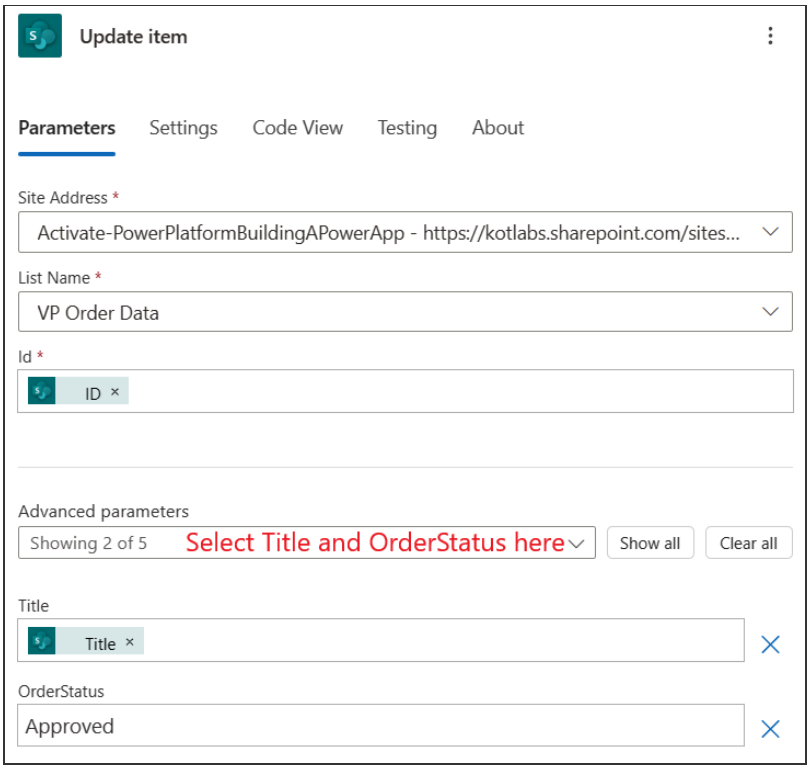


7. After **Send an email (V2)** action, click **+** icon and **Add an action**
8. In the actions window, search and select **'Update item'** from SharePoint connector



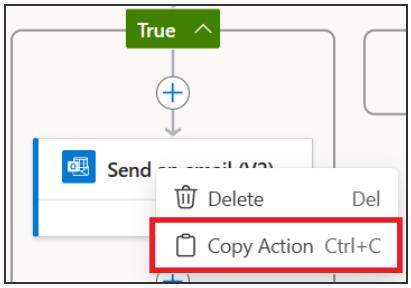
9. Set **Update item** parameters as below:
   - **Site Address**: URL of SharePoint site where the list is located
   - **List Name**: '<your initials> Order Data'
   - **Id**: ID (dynamic content from trigger)

     In Advanced parameters below, select **Title** and **OrderStatus**, set these parameters as below:
   - **Title**: Title (dynamic content from trigger)
   - **OrderStatus**: *Approved* (plain text)

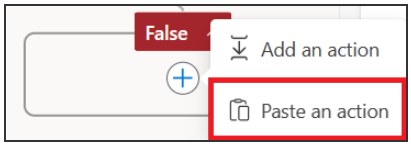


10. Now, when **'True'** branch of the condition is done, let's copy actions to the **'False'** branch and modify them accordingly.

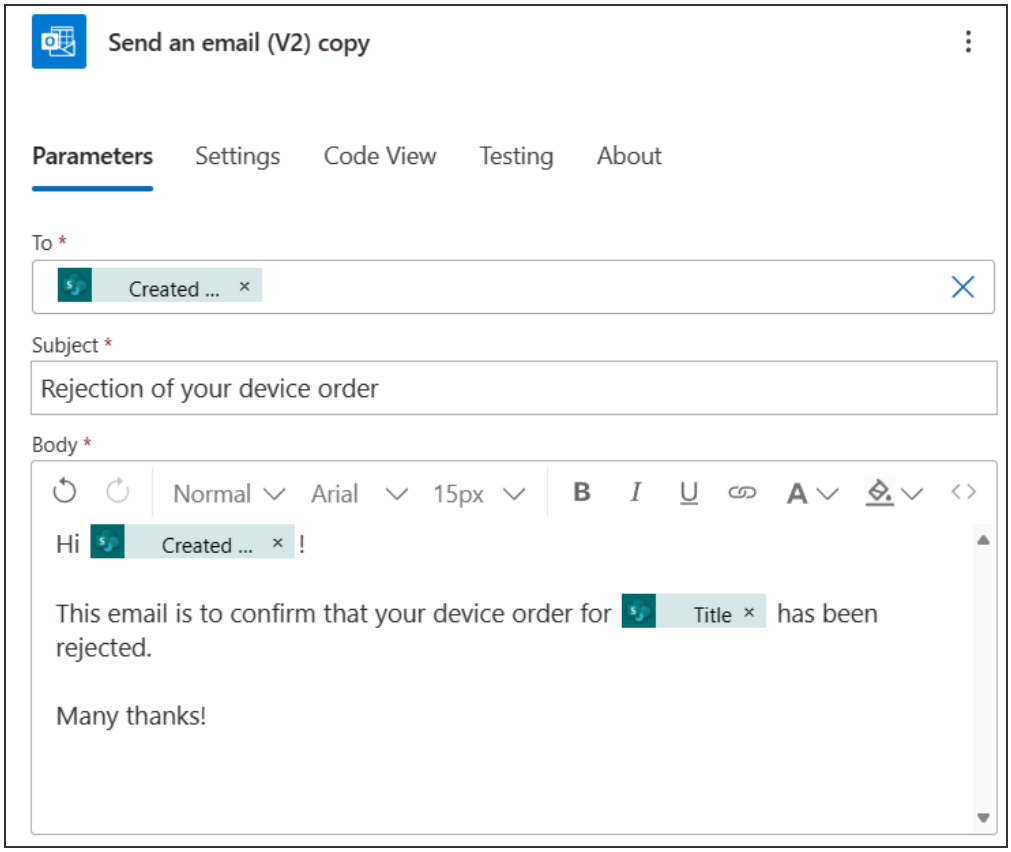
Right-click on **Send an email (V2)** action and select **Copy action**



11. In 'False' branch of the condition, click **+** icon and select **Paste an action**

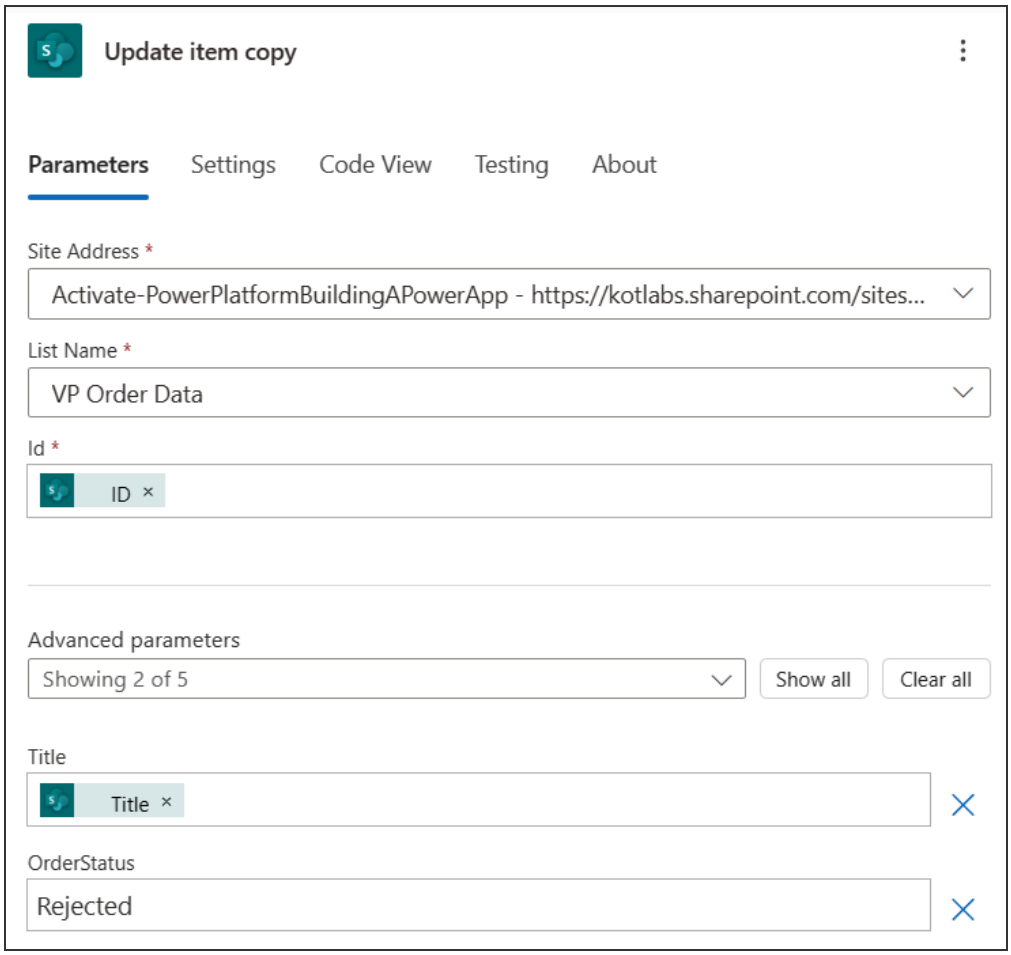


12. Modify parameters of **Send an email (V2) copy** to send a rejection email.



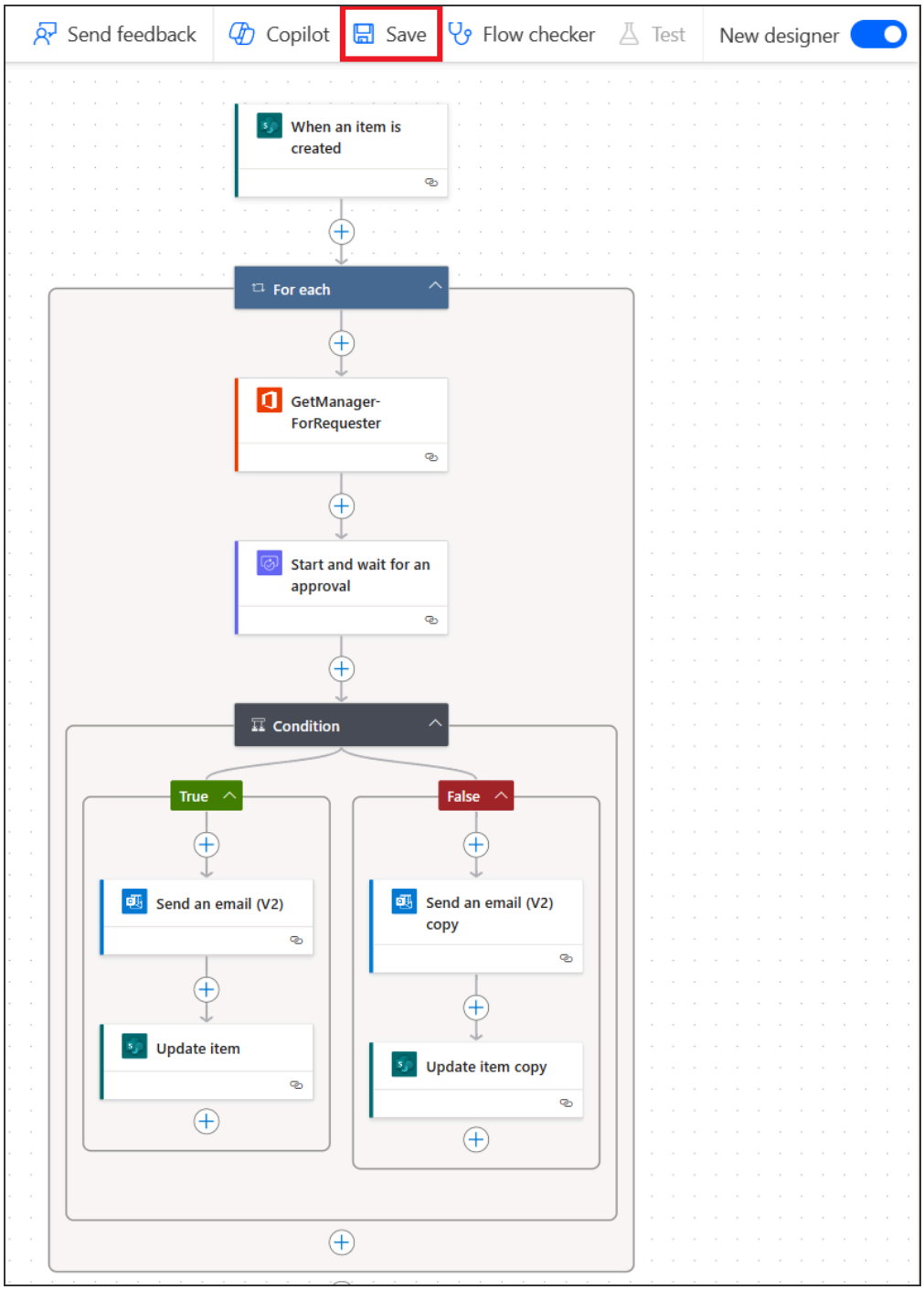
13. Repeat the same actions to copy and paste **Update item** action to the **'False'** branch of the condition.

    Set its parameters to update the item with the **OrderStatus** - *Rejected*.


Update item copy
Parameters
Settings
Code View
Testing
About
Site Address *
Activate-PowerPlatformBuildingAPowerApp - https://kotlabs.sharepoint.com/sites...
List Name *
VP Order Data
Id *
ID
Advanced parameters
Showing 2 of 5
Show all
Clear all
Title
Title
OrderStatus
Rejected

1. Your flow should look like the screenshot below.

   Save the flow by clicking **Save** in the top right corner of the screen.



Exercise 7: Test the app and flow

It is a good idea to save and test your app regularly. There are a few ways in which you can test your app:

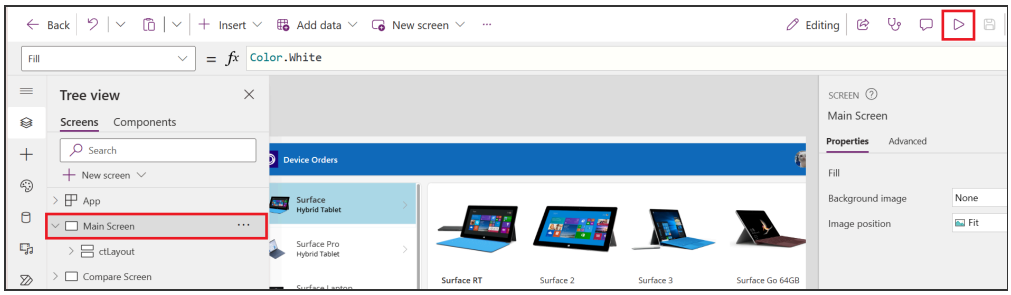
- Hold the Alt-key down and click on controls to activate the functionality
- Enter preview mode (F5)
- Play the app

In this exercise you will run through some tests of your app

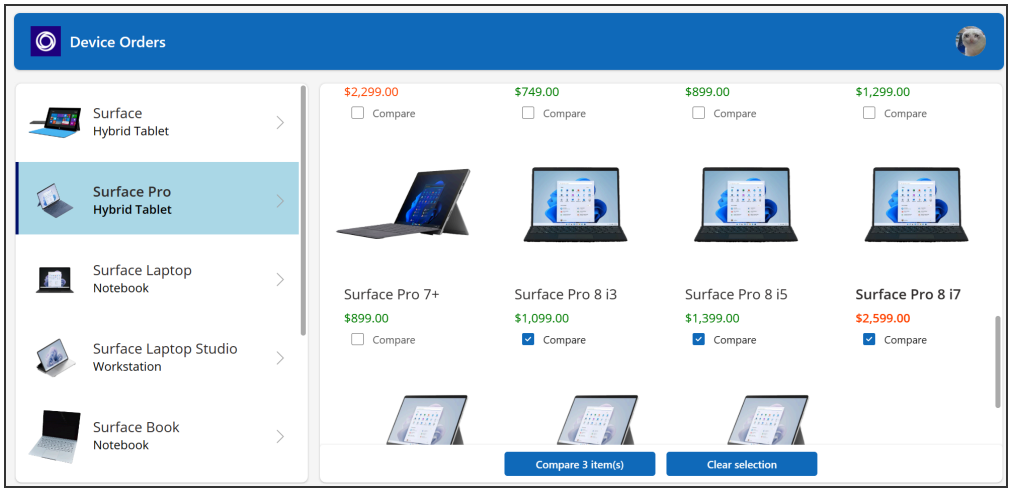
Task 1: Testing the app

In this task you will perform some tests in your app.

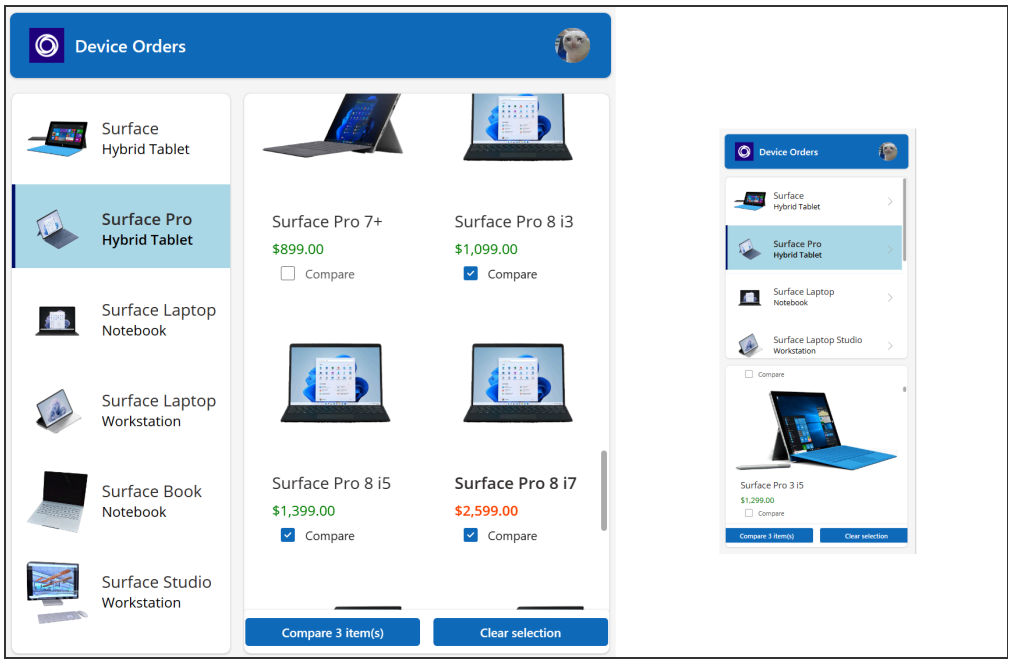
1. Navigate to https://make.powerapps.com and open your app (if it is not already open)
2. Go to the **Main Screen** and **Preview** the app by clicking the **Play** button in the top-right



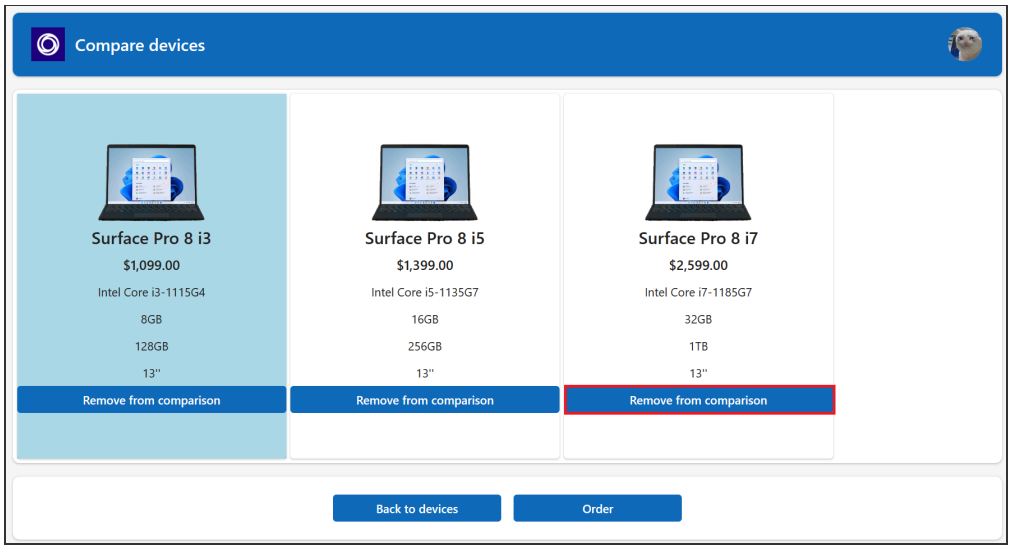
3. Uncheck if there are any checked devices
4. Select **'Surface Pro'** category on the left to show a filtered set of devices
5. Check the compare box on a few devices on the main screen from a few different categories



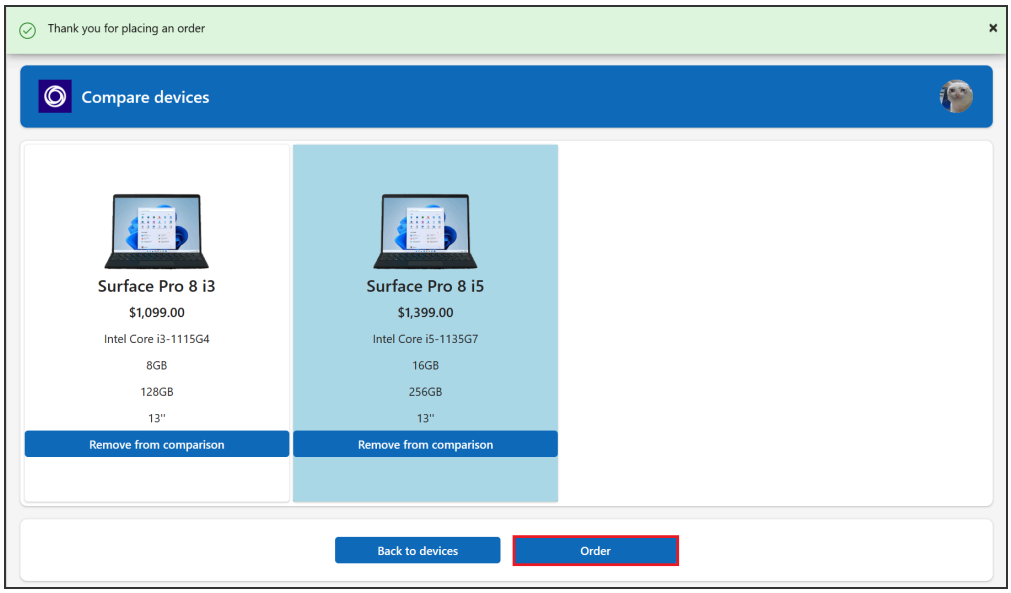
6. Resize the browser window to see how the app behaves in a smaller screen



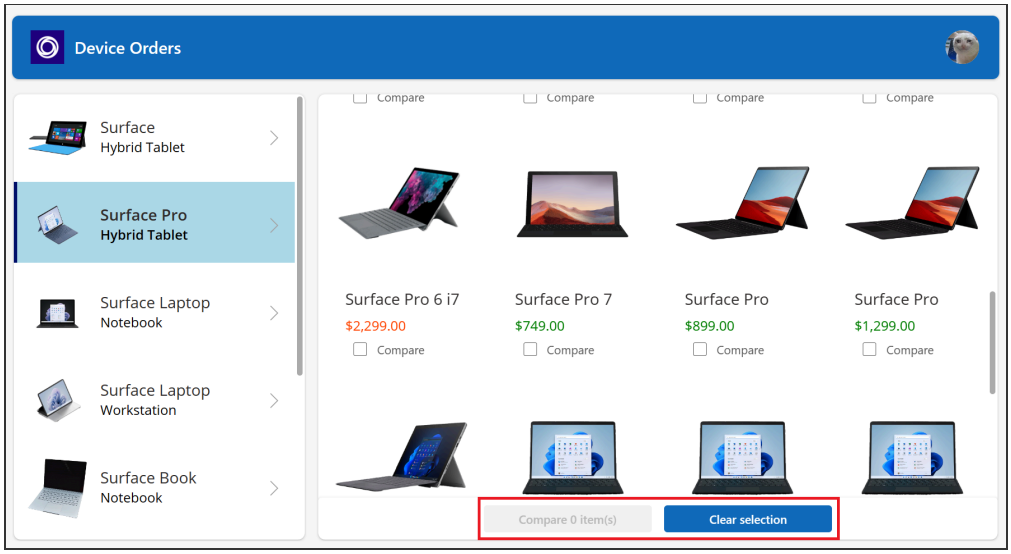
7. Click the **Compare** button to navigate to comparison and order screen
8. Remove a device from the comparison list by clicking the **Remove from comparison** button



9. Select preferred device and click the **Order** button. Notice toast notification to appear on top of the app.



10. Click **Back to devices** button to navigate back to the Main Screen
11. Click **Clear selection** button to clear the selected devices, see how **Compare** button is disabled now

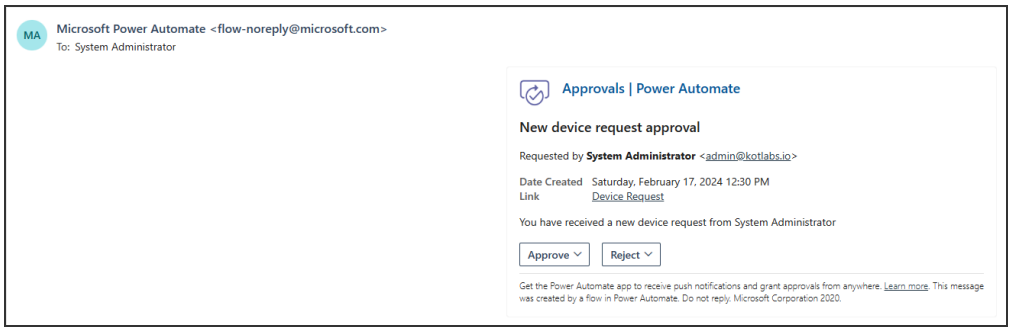


12. Close the preview

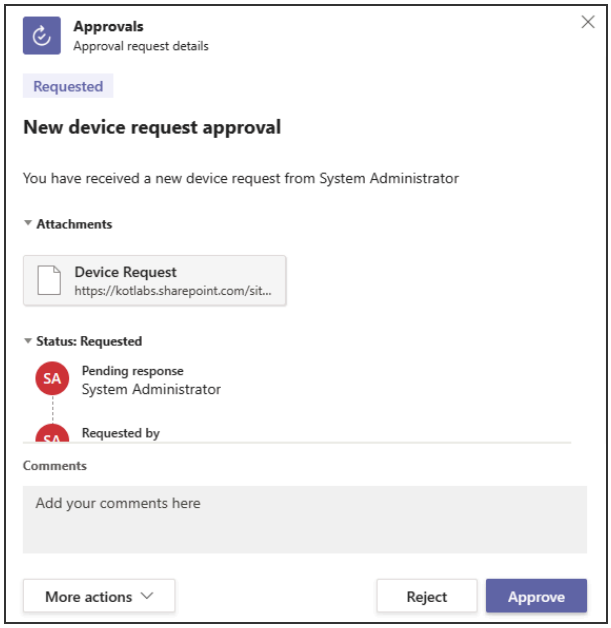
Task 2: Testing the flow

In this task, you will test the flow. When a device order is placed, a corresponding item is created in SharePoint. Creation of this list item will trigger the flow.

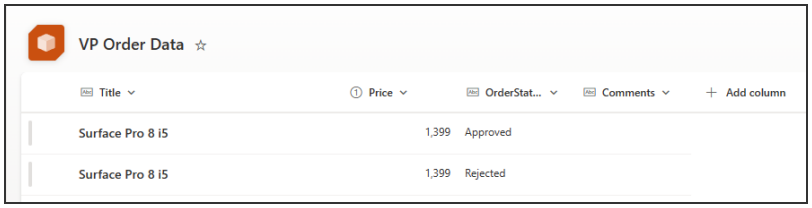
1. If you ordered a device in the previous task, proceed to the next step. Otherwise, ensure you ordered a device in the app.
2. Go to https://outlook.office.com and confirm that when you place a request for a new device, you receive the following email:



3. In your Microsoft Teams client open **Approvals** app and confirm that you have a new approval request



4. Approve or reject the request using any preferred tool.
5. Go to the **SharePoint site** and confirm that the request **OrderStatus** column has been updated to *'Approved'*
6. Go back to **Outlook** and check that you have received a confirmation email.
7. Follow from **Step 1** again, this time reject the request.



> **Note:** If your flow is not working as expected, check the flow run history for errors.
> Navigate to https://make.powerautomate.com/ and open your flow. Click on **Run history** in the left pane to see the details of the flow runs.

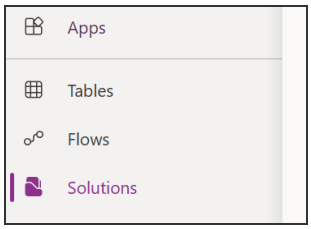
Primary lab module is complete!
Please, give an update to your instructor and proceed to optional tasks if you want to do them and if you have time remaining.

Optional exercise 1: Packaging as a Power Platform solution

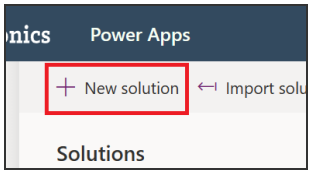
> **Note:** Please, consult with the instructor if you can proceed to this task.
> This exercise will rely on Power Platform features, that may be not included into your licensing or role permissions.

Task 1: Create a solution

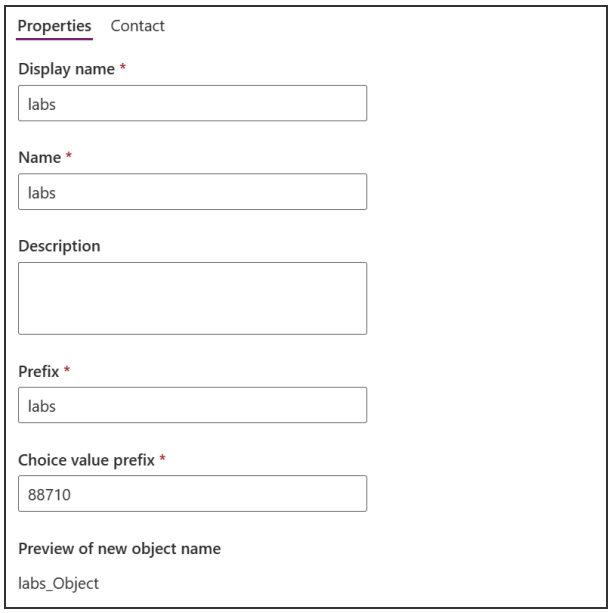
1. Navigate to https://make.powerapps.com
2. Make sure that preferred Power Platform environment is selected
3. In left navigation click **Solutions** (look for this option in **More...** if it's not displayed)



4. In toolbar click **+ New solution**



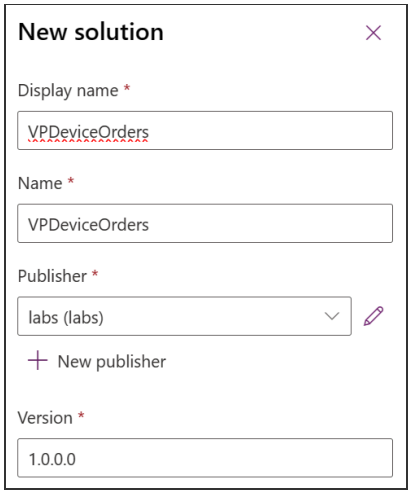
5. Check if **labs** publisher exists.
   If not, click **+ New Publisher** and create publisher with **Display name**, **Name**, and **Prefix** set as *labs*



Click **Save** button below

6. Set both **Display name** and **Name** as *YOUR INITIALS DeviceOrders* (e.g., VPDeviceOrders)

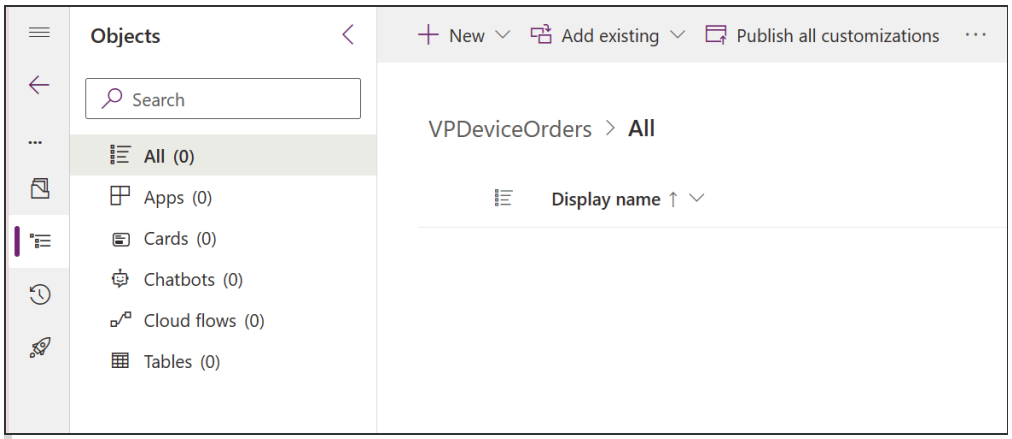
   Set **labs** created in previous step as a **Publisher**



   Click **Create** button below

Task 2: Create a Dataverse table to store the requests

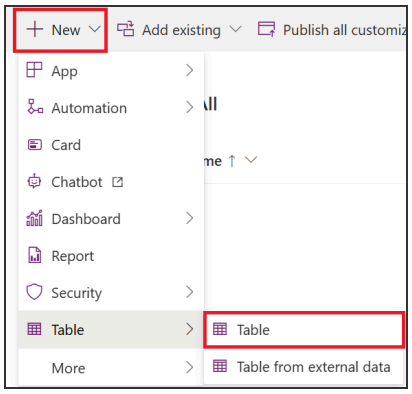
1. Your solution is currently empty, and its contents look like this



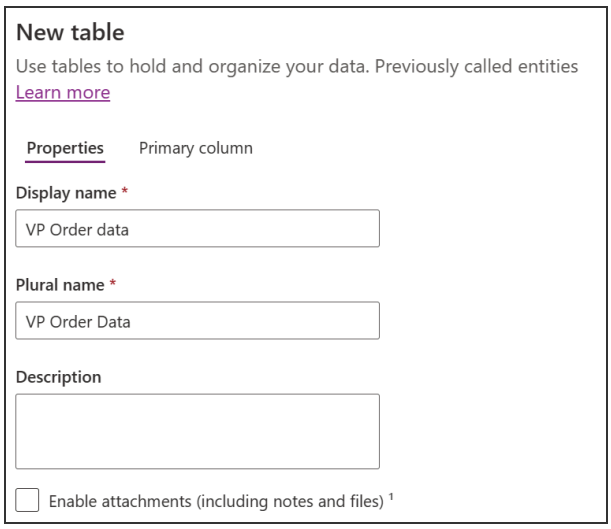
> **Note:** You can quickly navigate there from https://make.powerapps.com -> Solutions -> Click on your solution in the list

2. In this exercise you will switch the app from SharePoint list to Dataverse table that will be part of your solution

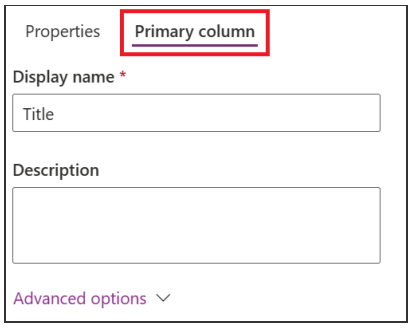
   Click **+ New** , then **Table** , and again **Table**



3. Name this table just like your SharePoint list: **<YOUR INITIALS> Order Data** (e.g., *VP Order Data*).



4. Select **Primary column** tab and rename it as **Title**

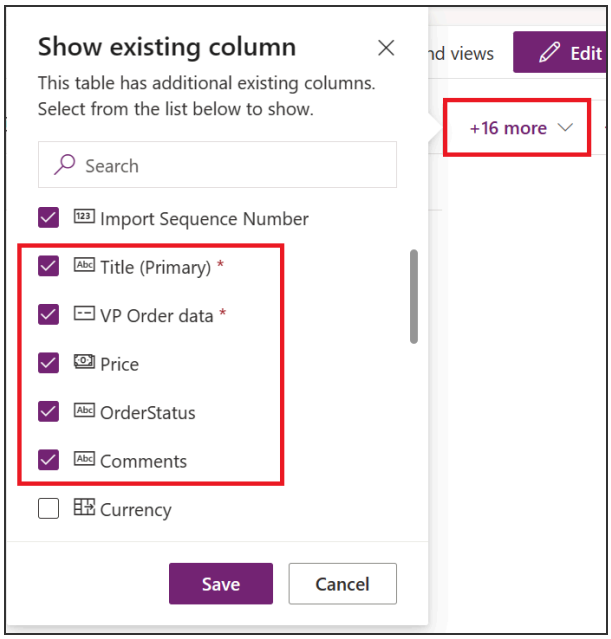


5. Click **Save** button below to create the table
6. Click **+ New** , then **Column** to add a column to the table

   Add the following columns to the list:

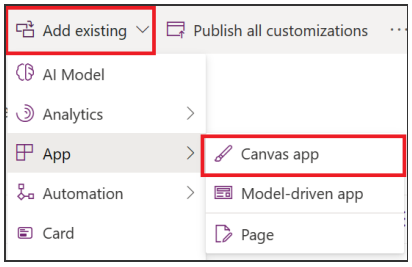
   - **Price**: Currency
   - **OrderStatus**: Single line of text
   - **Comments**: Single line of text

     Notice, that we have the columns with the same name, like we had in SharePoint list



Task 3: Add existing Device Orders app to the solution

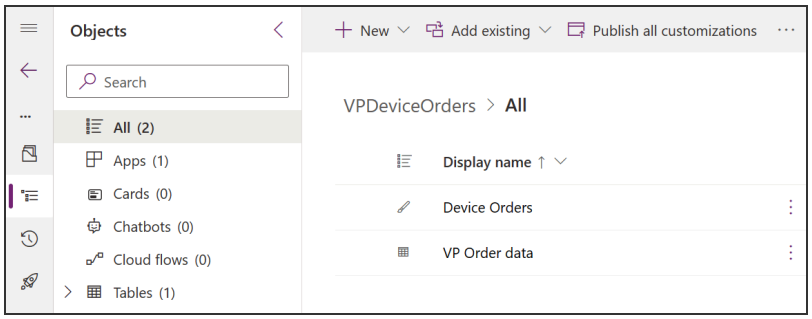
1. Navigate back to your solultion contents
2. In the toolbar, click **Add existing**, then **App**, then **Canvas App**



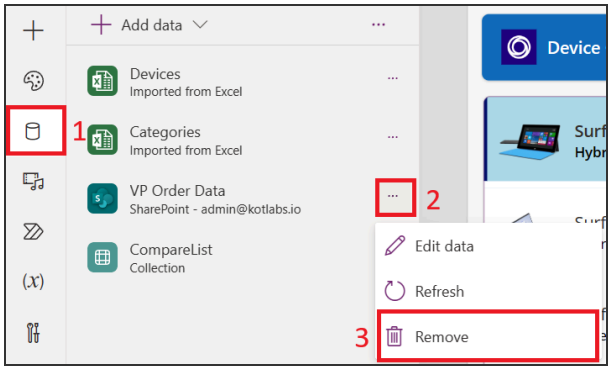
3. Switch to **Outside Dataverse** tab and select your **Device Orders** app

Click **Add** button below. It may take a few minutes to add an app to the solution.

4. Your solution contents should show both table and the app now:

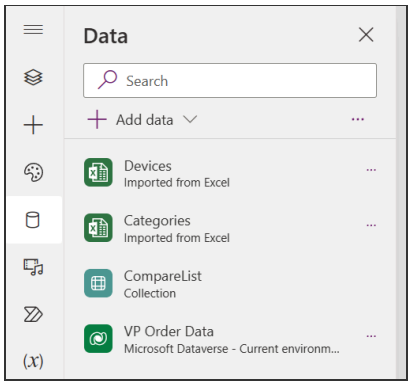


5. Click on **Device Orders** app within solution to start editing it
6. In app authoring pane, select **Data** to see data connections
7. **Remove** SharePoint list from the data connections



8. Click **+Add data**, and add previously created Dataverse table to the app

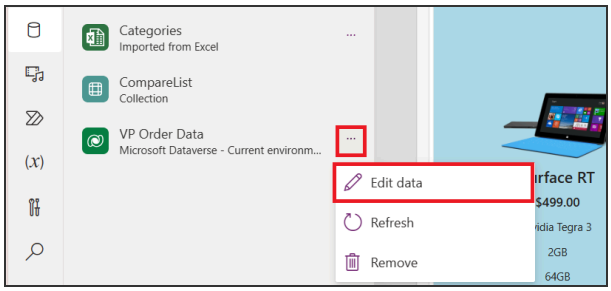
Now your connections list should look like this:



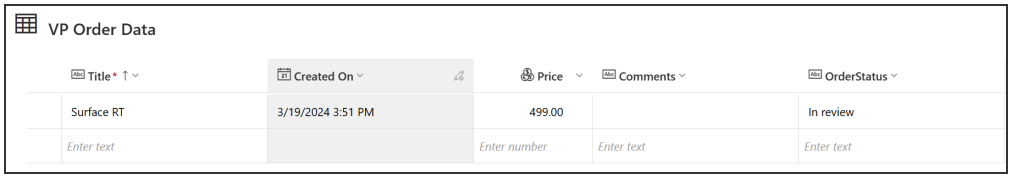
9. Because table has the same name as SharePoint list and the same column names, there will be no need to change formulas - everything should work as is. You can **test the app** now.

   > **Note:** If the table name or column names differ from SharePoint list, then, please, follow App Checker instructions to update Patch() formula to use correct data source and column names.

10. Select any device and order it. In Data pane, you can click **Edit data** to see if table contents have changed.



New entry should be displayed in the table. You can modify the view to see values in the other columns.

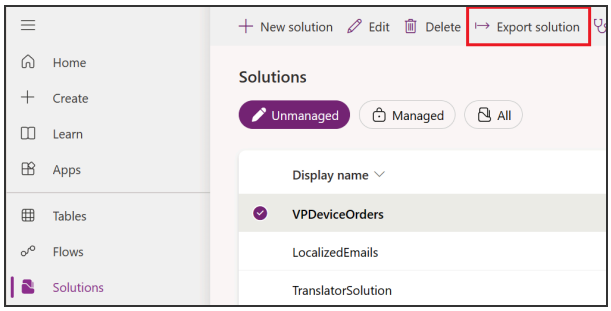


11. **Save** and **Publish** the app

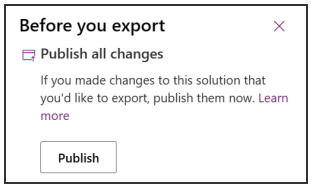
Task 4: Export solution

1. Now your solution can be exported or deployed to another environment using Power Platform pipelines.

   Navigate to https://make.powerapps.com, then **Solutions**
2. Select your solution and click **Export solution** in the toolbar



3. Before exporting you may need to **Publish** all changes



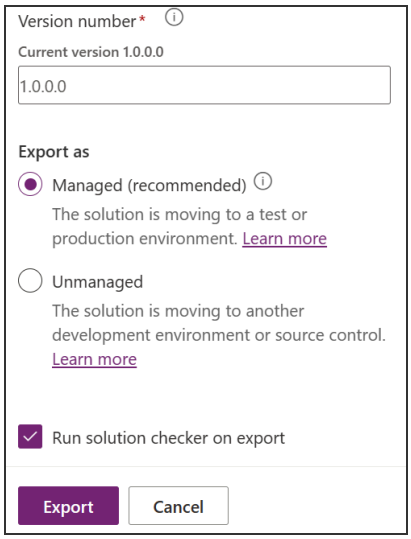
4. Click **Next**
5. Define solution version number (*major.minor.build.revision* format)

   Select solution packaging type:

   - **Managed**: ready for production, Device Orders app cannot be changed after deployment

   - **Unmanaged**: objects in the solution can be edited, good for further development or dev collaboration

   Consider **Run solution checker on export** to detect potential problems



6. Click **Export**, it may take some time.

   Solution package can be downloaded as a ZIP file.

Optional task: Add flow to the solution

> **Challenge:** Technically, **Purchase approval** flow should be part of the solution too. You can either:
> - create a new flow within solution (rely on Exercise 6 but use Dataverse connector);
> - or, add existing **Purchase approval** flow and replace SharePoint trigger and actions with Dataverse.

Optional exercise 2: Integration with Microsoft Teams

Coming soon