

Building solutions with Dataverse for Teams

Lab 2 – Advanced data models and apps

Workshop Version: 1.4, Published: 02-2023

Table of contents

Exercise 1: Advance data model	2
Task 1: Navigate to Dataverse for Teams content.....	3
Task 2: Create Registration table and relationship.....	5
Task 3: Set table permissions	8
Optional task: Create Points table to store award points balance	10
Exercise 2: Build Power App from scratch	12
Task 1: Create a new Power App and connect to data	13
Task 2: Create layout and learn naming conventions	15
Task 3: Setting up Initiatives gallery	22
Task 4: Setting up Initiative display form	26
Task 5: Setting up Registrations gallery	29
Task 6: Add and delete Registrations.....	35
Appendix: Localized formulas	38

Exercise 1:

Advance data model



Important – Working with lab tenant

- All the labs in this course require you to use the latest version of Edge or Chrome in Incognito/InPrivate mode.
- Use Office 365 credentials retrieved from Skillable (Labs on Demand) in Lab 0.
- Always remember to replace M365XXXXXXX with your lab tenant prefix.
- If you are experiencing any problems with working in your lab tenant – please, notify your instructor as soon as possible.

Objectives:

- Create additional tables in Dataverse for Teams
- Define relationships between tables
- Set up table permissions

Estimated time:

20 minutes

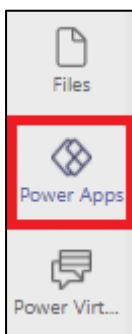
Task 1: Navigate to Dataverse for Teams content



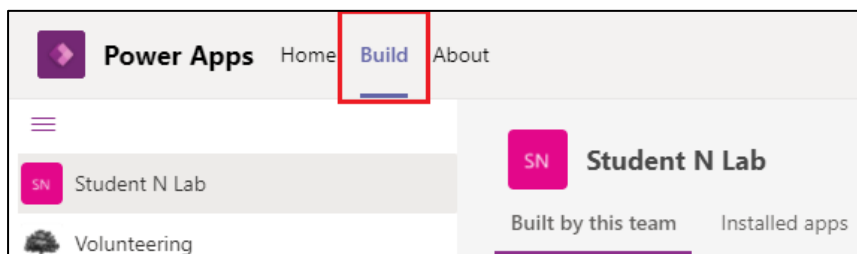
Important

This routine will be referenced as “Navigate to Dataverse for Teams content” and will be repeated many times during this course.

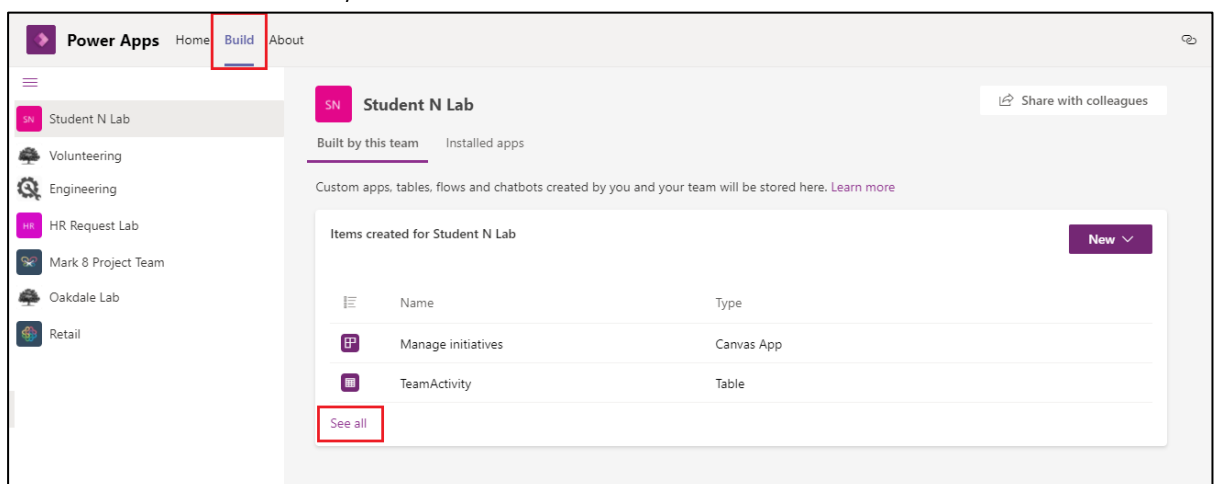
1. In **InPrivate/Incognito** browser mode, while logged in into your **M365xXXXXXX lab tenant**, navigate to <https://teams.microsoft.com>
2. Open **Power Apps** app using the left bar



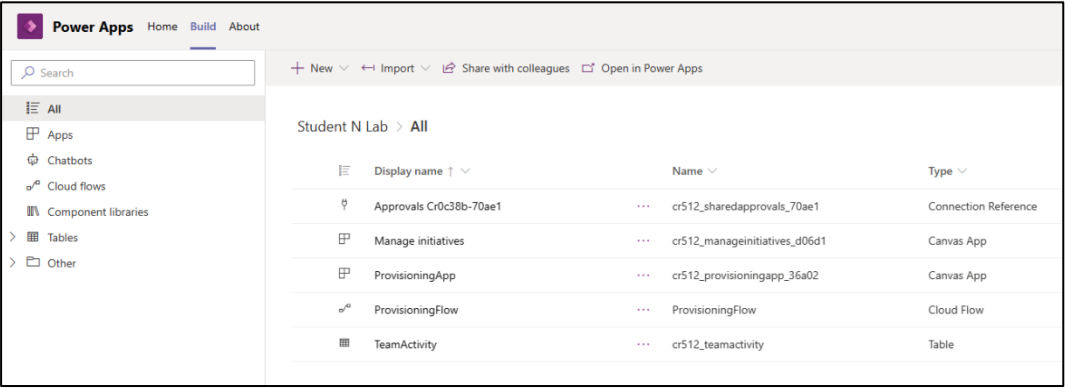
3. Select **Build** tab, then find and select your Team



4. In **Items created for ...** , click **See all** link



5. In this interface, you can see all the content created in your Team’s **Dataverse** **for Teams** environment:

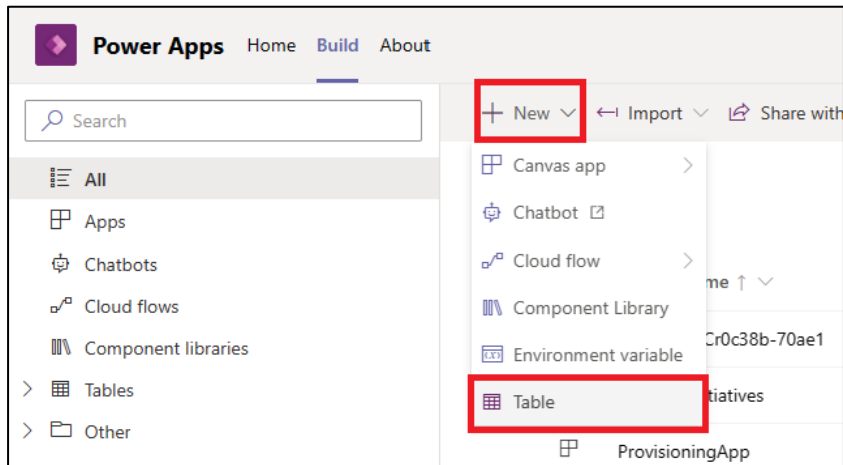


The screenshot shows the Power Apps interface. On the left is a navigation pane with a search bar and a list of categories: All, Apps, Chatbots, Cloud flows, Component libraries, Tables, and Other. The main area displays a list of applications under the heading 'Student N Lab > All'. The list has columns for 'Display name', 'Name', and 'Type'. Each row includes an icon, a display name, a three-dot menu, a name, and a type.

	Display name ↑ ↓		Name ↓	Type ↓
📄	Approvals Cr0c38b-70ae1	...	cr512_sharedapprovals_70ae1	Connection Reference
📄	Manage initiatives	...	cr512_manageinitiatives_d06d1	Canvas App
📄	ProvisioningApp	...	cr512_provisioningapp_36a02	Canvas App
📄	ProvisioningFlow	...	ProvisioningFlow	Cloud Flow
📄	TeamActivity	...	cr512_teamactivity	Table

Task 2: Create Registration table and relationship

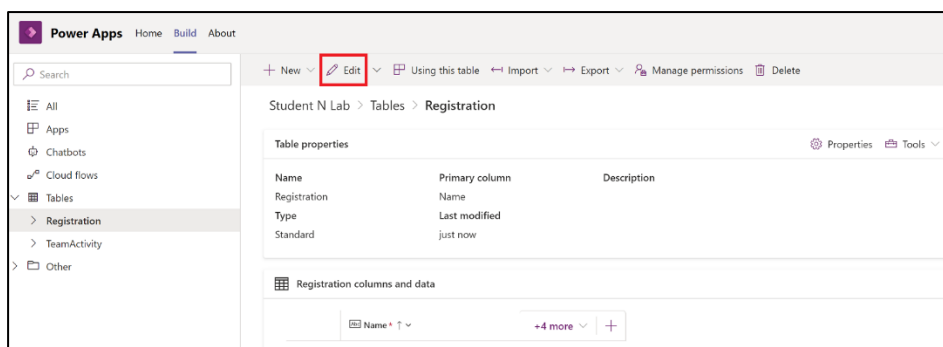
1. **Navigate to Dataverse for Teams content** of your Team
2. Click **+New** and select **Table** in a dropdown menu:



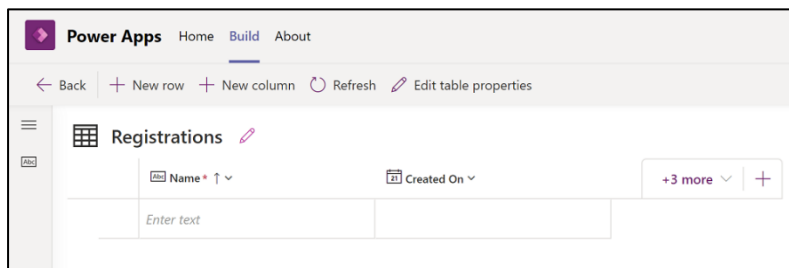
3. Set **Display Name – Registration** and click **Save** button

A screenshot of the 'New table' form in Power Apps. The form has a title 'New table' and a subtitle 'Use tables to hold and organize your data. Previously called entities'. Below the subtitle is a 'Learn more' link. The form has two tabs: 'Properties' and 'Primary column'. The 'Properties' tab is active. It contains three input fields: 'Display name *' with the value 'Registration', 'Plural name *' with the value 'Registrations', and 'Description'. Below these fields is an 'Advanced options' section with a dropdown arrow. At the bottom of the form are two buttons: 'Save' and 'Cancel'. The 'Save' button is highlighted with a red box.

4. Select newly created **Registration** table details will get displayed.
In a toolbar, click **Edit** to open table editor




5. Now you can modify table in the same way, you did in the previous lab.



6. Let's add a few columns:

Name	Type	Format	Advanced Settings
UserEmail	Single line of text	Email	Max char count: 100
ApprovalStatus	Choice	Sync with global choice? - No	Choices: - In waitlist - Approved - Rejected

For **ApprovalStatus** column select **Sync with global choice?** as 'No'

Click  icon if you want to set a colour for the choice.

Set **Default choice** as 'In waitlist'

7. Now we will add a many-to-one relationship between **TeamActivity** and **Registration** by creating a **Lookup** column:

Display name: relTeamActivity

Data type: Lookup

Related table: TeamActivity

Click **Save** button

New column
Previously called fields. [Learn more](#)

Display name *
relTeamActivity

Description ⓘ

Data type * ⓘ
Lookup

Required ⓘ
Optional

☒ Searchable ⓘ

Related table *
TeamActivity

Advanced options ▾

Save Cancel

This table will be used later to track user registrations for Team initiatives.

8. Add two sample entries (*don't forget to replace xxxxxxxx*):

a. **Name:** Adele Vance

UserEmail: adelev@m365xxxxxx.onmicrosoft.com

ApprovalStatus: In waitlist

relTeamActivity: Team Building CY2023

b. **Name:** Megan Bowen

UserEmail: meganb@m365xxxxxx.onmicrosoft.com

ApprovalStatus: Approved

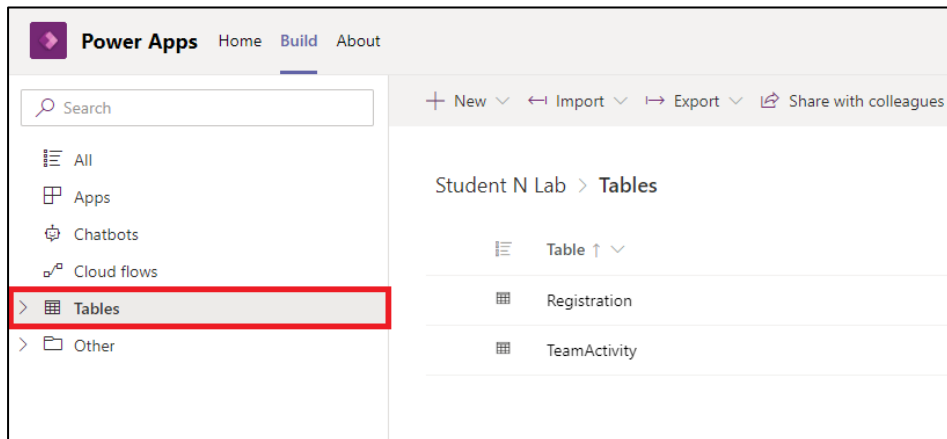
relTeamActivity: Team Building CY2023

Registrations			
Name * ↑ ▾	UserEmail ▾	ApprovalStatus ▾	relTeamActivity ▾
Adele Vance	adelev@m365xxxxxx.onmicrosoft.com	In waitlist	Team Building CY2023
Megan Bowen	meganb@m365xxxxxx.onmicrosoft.com	Approved	Team Building CY2023
Enter text	Enter email	Select option	Select lookup

9. Task is completed.

Task 3: Set table permissions

1. If you just finished previous task and are still in the table editor, click **Back**. Otherwise, **Navigate to Dataverse for Teams content** (see Task 1)
2. Select **Tables** in left navigation menu:



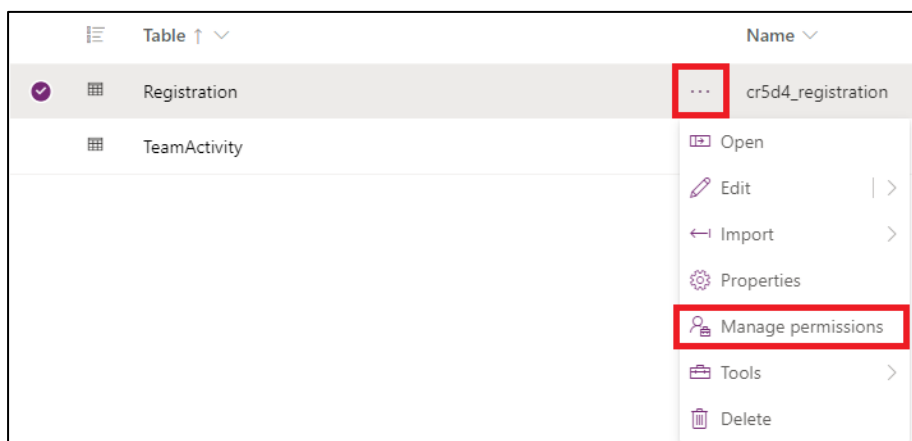
3. To comply with solution security requirements, we need to define table permissions:

Team Owners can perform any operations with table in Dataverse for Teams environment of their team.

Team Members should have limited permissions for certain tables:

- *TeamActivity* – only team owners can create, update, delete, but other team members should be able to read-only
- *Registration* – team members can only create requests, but not modify them

4. Click **ellipsis (...)** to open context menu for **Registration** table and select **Manage permissions** option:



5. Set **Registration** table permissions for **Members** as **Collaborate**
Click **Save** button

Manage Registration table permissions [X]

Set permissions for each type of user group. [Learn more](#)

Student N Lab team

- Owner
Full access
- Member**
Private
- Guest
Private

Colleagues with access

- This isn't shared with any colleagues
None

Full access

- ✓ Create new records
- ✓ Read all records
- ✓ Update or delete all records

Collaborate

- ✓ Create new records
- ✓ Read all records
- 🔒 Update or delete their own records

Reference

- ✗ Can't create new records
- ✓ Read all records
- ✗ Can't update or delete records

Private

- ✓ Create new records
- 🔒 Read their own records
- 🔒 Update or delete their own records

None

- ✗ Can't access records at all

Save Cancel

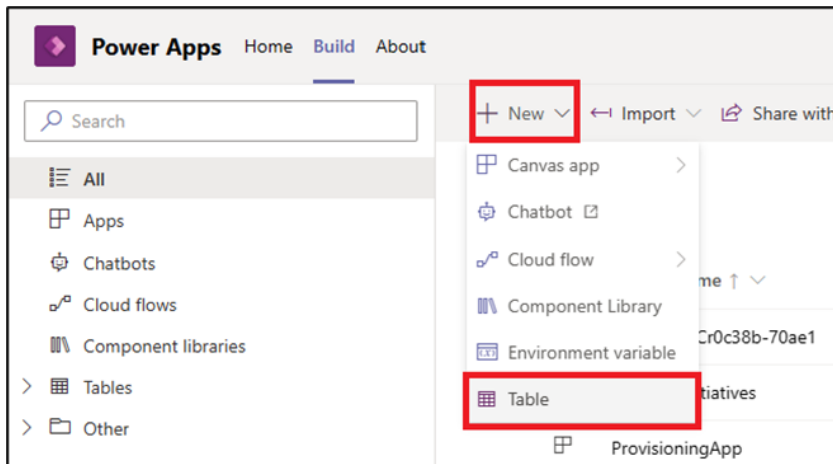
6. **Repeat steps 2-5** to set up permissions for **TeamActivity** table:

TeamActivity → Members – Reference

7. Task is completed.

Optional task: Create Points table to store award points balance

1. **Navigate to Dataverse for Teams content** of your Team
2. Click **+New** and select **Table** in a dropdown menu:



3. Set **Display Name – Points** and click **Save** button

A screenshot of the 'New table' dialog box in Power Apps. The dialog box has a title bar 'New table' and a close button. Below the title bar is a description: 'Use tables to hold and organize your data. Previously called entities' and a link 'Learn more'. There are two tabs: 'Properties' and 'Primary column'. The 'Properties' tab is selected. It contains three input fields: 'Display name *' (with 'Points' entered), 'Plural name *' (with 'Points' entered), and 'Description'. Below these fields is an 'Advanced options' section with a dropdown arrow. At the bottom are 'Save' and 'Cancel' buttons. The 'Display name' field is highlighted with a red box.

4. Click **Edit** in toolbar to start the table editor
5. Let's add a few columns:

Name	Type	Format	Advanced Settings
UserEmail	Single line of text	Email	Max char count: 100
Balance	Whole number	None	Minimum value: 0

6. Add a few sample entries (*don't forget to replace xXXXXXX*):

a. **Name:** Adele Vance










UserEmail: adelev@m365xxxxxxx.onmicrosoft.com

Balance: 1000

b. **Name:** System Administrator

UserEmail: admin@m365xxxxxxx.onmicrosoft.com

Balance: 750

Points 			
	 Name *  	 UserEmail 	 Balance 
	Adele Vance	adelev@m365xxxxxxx.OnMicrosoft.com	1,000
	System Administrator	admin@m365xxxxxxx.OnMicrosoft.com	750
	<i>Enter text</i>	<i>Enter email</i>	<i>Enter number</i>

7. Task is completed.

Exercise 2:

Build Power App from scratch

Objectives:

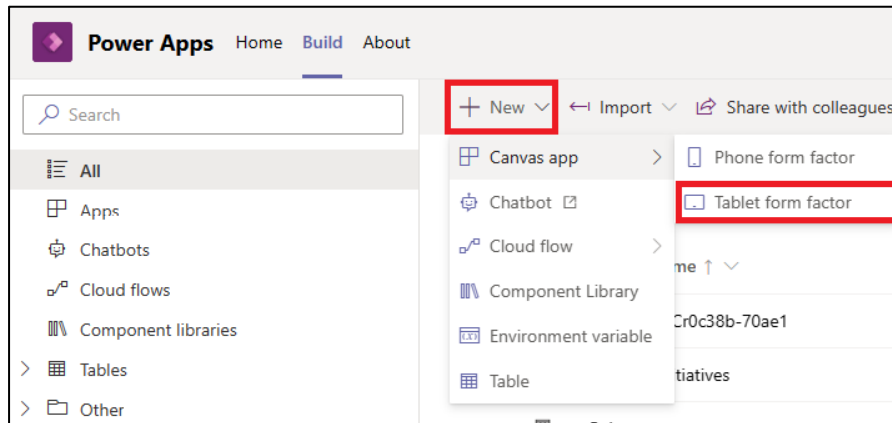
- Learn how to use galleries to display and select data entries
- Use the advantages of Dataverse for Teams relational database
- Utilize Power Apps formula language to implement business logic

Estimated time:

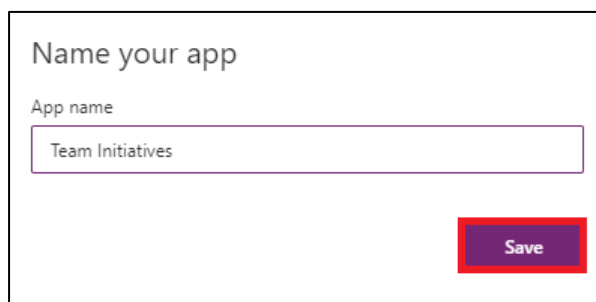
40 minutes


Task 1: Create a new Power App and connect to data

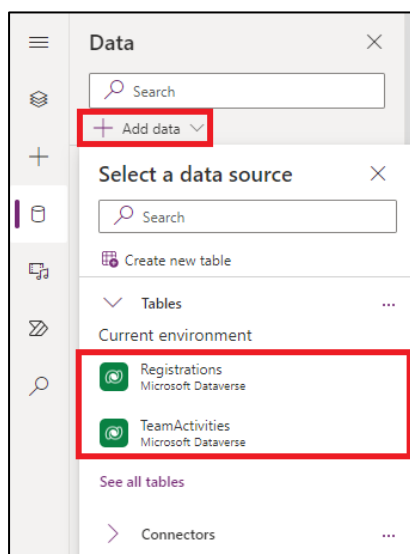
1. **Navigate to Dataverse for Teams content** of your Team
2. Create a new Power App, click **+New** -> **Canvas App** -> **Tablet form factor**




3. Name a new app as **Team Initiatives** and click **Save**

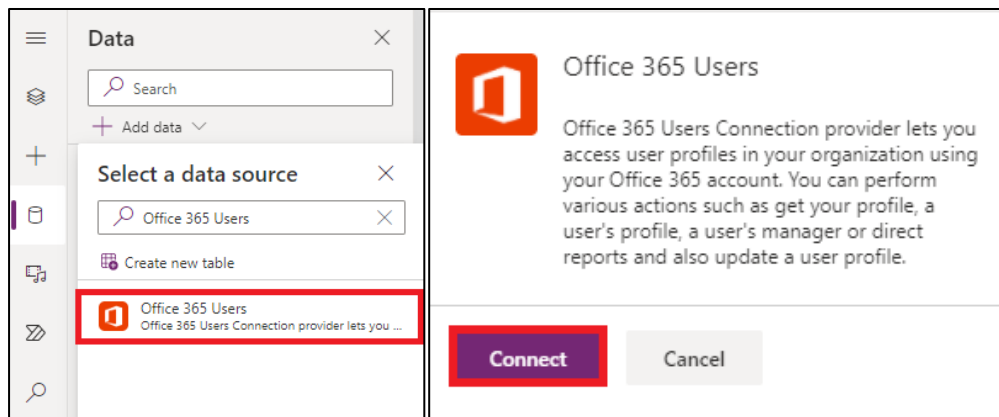


4. In Data  panel, click **+Add data** and add **TeamActivities** and **Registrations** tables to the app:



5. In Data  panel, click **+Add data** and in **Search** type *Office 365 Users*. Add **Office 365 Users** connector to the app by clicking on it. We will use this connector to query user profile (e.g., extract user photo from directory)

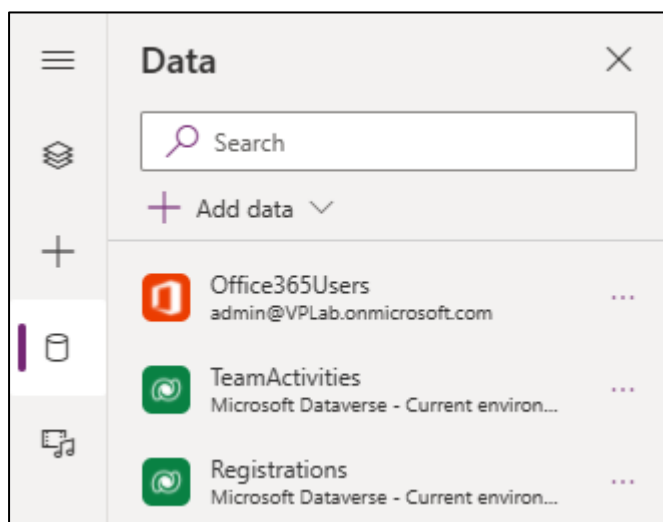
Click **Connect** in the right panel.



Localized connector names

Office 365 Users connector may have different name in your language if you have different locale settings on your computer.

6. The list of connected data sources should look like this:



Order of data sources is NOT important.

Task is completed. Proceed to the next task.

Task 2: Create layout and learn naming conventions



Before we start -> About Power Apps formula language

Please, pay attention that Power Apps formula language (PowerFx) is **case-sensitive**.

Also, Power Apps formula language (PowerFx) depends on locale settings of your computer.

Documentation: [Operators and Identifiers - Power Apps | Microsoft Docs](#)

For example, consider the following formula expressed in a language and region that uses dot or period as the decimal separator, such as US, Japan or the United Kingdom:

```
If( Slider1.Value > 12.59;  
    Notify( "Valid!", Success ); Navigate( "NextScreen", None );  
    Notify( "Invalid, try again", Error )  
)
```

Now view this same formula in a language and region where a comma is used for the decimal separator, such as France or Spain:

```
If( Slider1.Value > 12,59;  
    Notify( "Valid!"; Success );; Navigate( "NextScreen"; None );  
    Notify( "Invalid, try again"; Error )  
)
```

Internally the formula doesn't change, all that changes is how it's displayed and edited by the author. Two different authors using two different languages can view and edit the same formula, with each seeing the appropriate separators and operators for their language.

The lab document contains cross-references to Appendix with localized formulas.

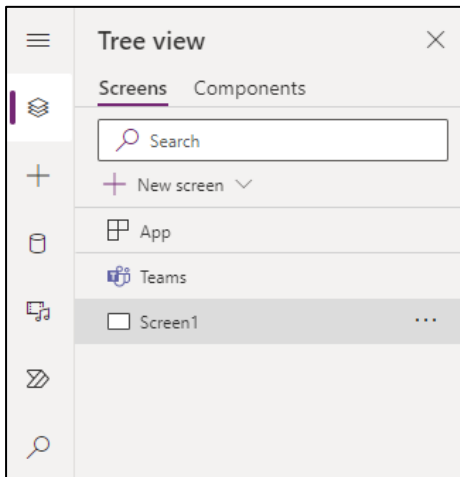





Canvas Apps Coding Standards and Guidelines

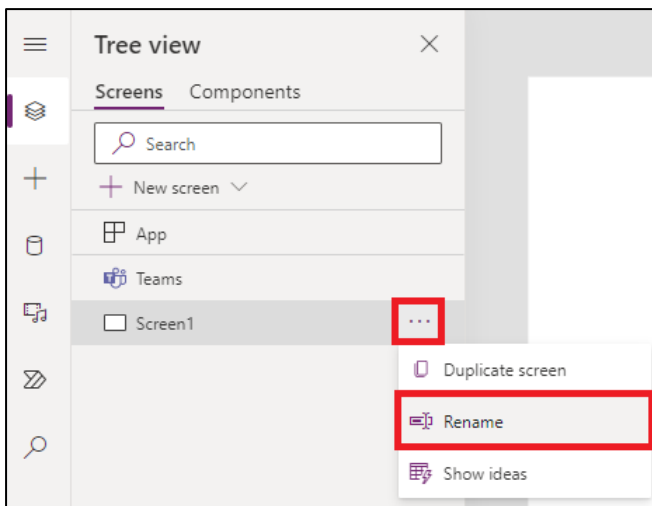
In this lab, we will rely on the best practices and naming conventions defined in Canvas apps coding standards and guidelines.

Full document link: <https://aka.ms/powerappscanvasguidelines>

1. Switch left panel to **Tree View**  :



2. Notice **App** and **Teams** objects:
 - a.  **App** is a top-level object, that represents our application. **OnStart** property, is especially interesting for us – it allows us to define, what commands should be executed when the app starts.
 - b.  **Teams** is an integration object unique for apps created in Dataverse for Teams. It contains the details of the current context in Microsoft Teams: current team, channel, colour theme information etc
3. Right-click on **Screen1** in **Tree View**  and select **Rename** it as **Main Screen**





Screen naming

Use plain language to name your screens, and that the names include spaces and no abbreviations. Also, we recommend that you end the name with the word "Screen," so that the context is understood when the name is announced.


Good examples:

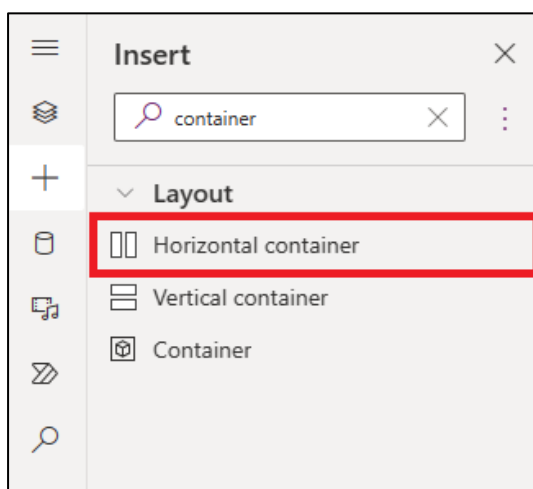
- Home Screen
- Thrive Help Screen

Bad examples:

- Home
- LoaderScreen
- EmpProfDetails
- Thrive Help

4. Let's create responsive layout for the app.

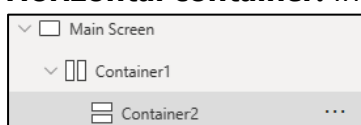
Open **Insert**  pane and add **Horizontal container**



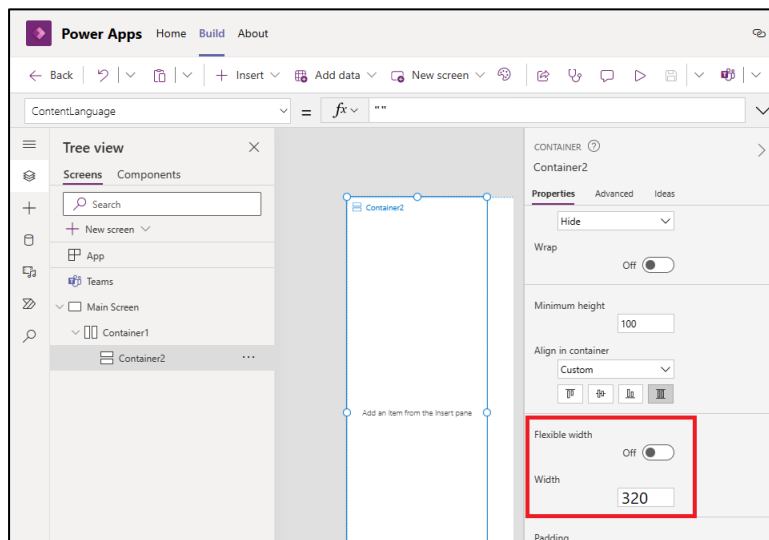
We will utilize this method to add controls to the screen during this lab. Use **Search** to find control that you need, then select it.

5. **Resize** container to fit the screen canvas size.

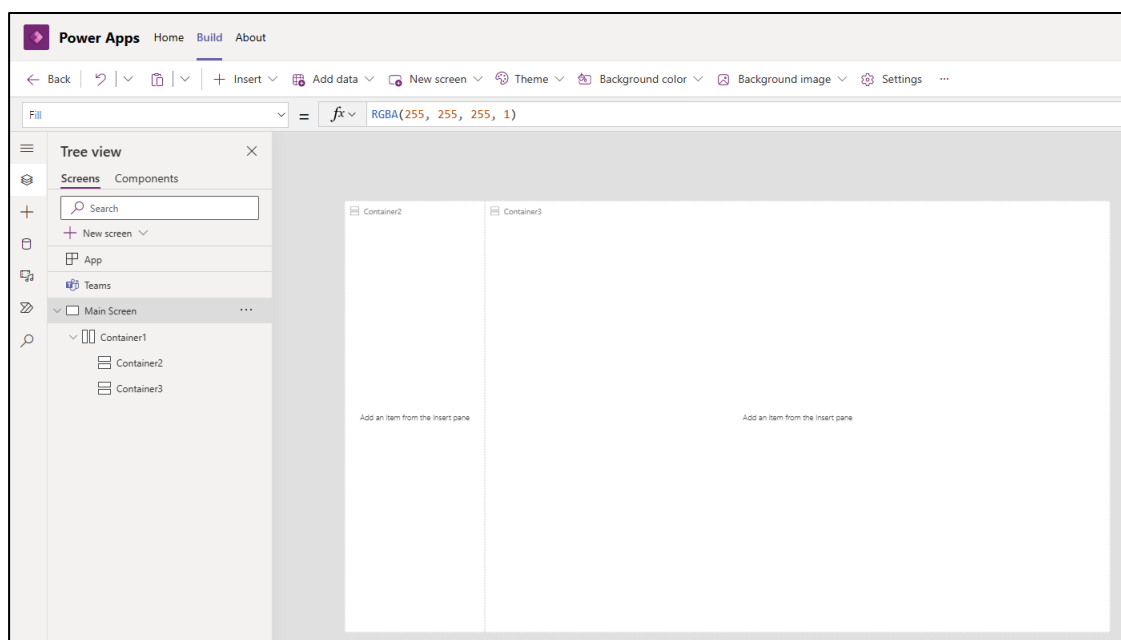
6. Open **Insert**  pane and add a **Vertical container** into the existing **Horizontal container**. In **Tree View**  it should look like this:



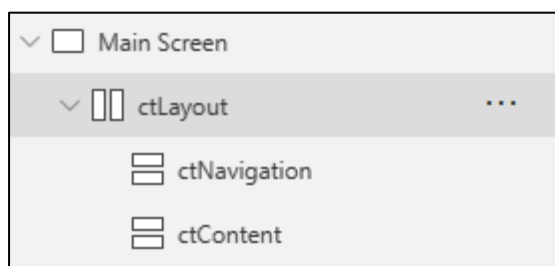
7. Select **Container2** (Vertical), set **Flexible width - Off** and **Width – 320**



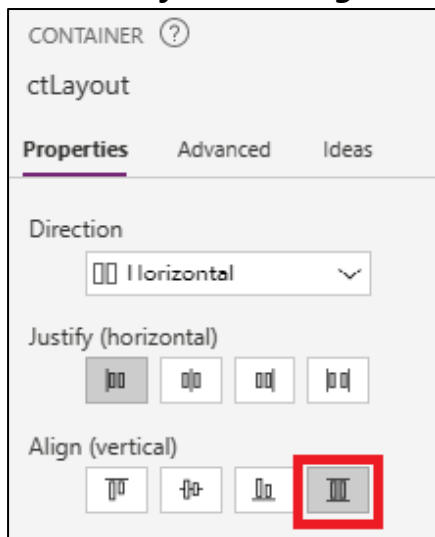
8. From **Insert** + pane, add one more **Vertical container** to **Container1** (Horizontal). Your app screen should look like this:



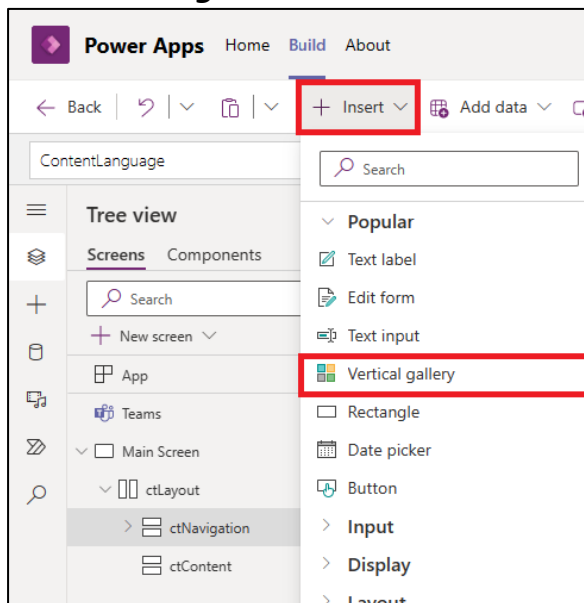
9. Let's rename containers as: **ctLayout**, **ctNavigation**, **ctContent** (in order of appearance). In **Tree View** it should look like this:



10. Select **ctLayout**, set **Align (vertical)** property as **Stretch**



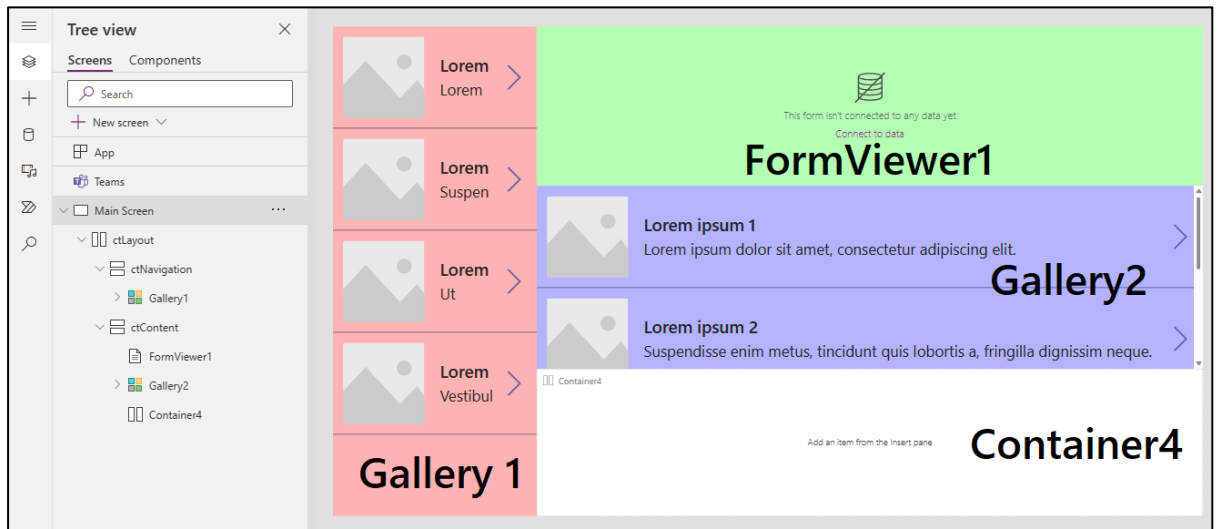
11. Select **ctNavigation** and **+Insert** one **Vertical gallery** into this container



12. Select **ctContent** and **+Insert** multiple controls into this container:

- Display form
- Vertical gallery
- Horizontal container

13. Your **Main Screen** layout should look like this now:

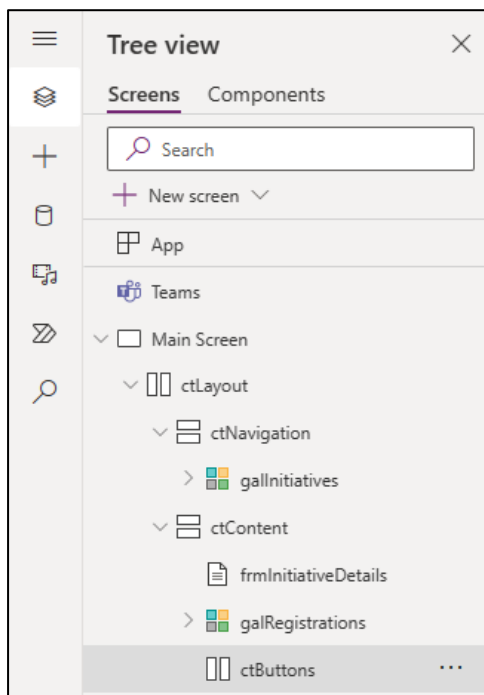


Note: Not actual colours – just a layout example.

14. In **Tree View** pane, **Rename** controls according to naming conventions:

- Gallery 1 -> galInitiatives
- Gallery 2 -> galRegistrations
- FormViewer1 -> frmInitiativeDetails
- Container4 -> ctButtons

Tree View pane should look like this as a result:





Controls naming

All control names on the canvas should use camel case. They should begin with a three-character type descriptor, followed by the purpose of the control. This approach helps identify the type of control and makes it easier to build formulas and search.

Here are good examples: `lblUserName`, `btnSend`, `imgLogo`

3-character type descriptors for the controls in this lab:

- `lbl` - label
- `btn` - button
- `frm` - form
- `gal` - gallery
- `img` - image
- `tim` - timer

Don't forget to save your app.

Task is completed.

Task 3: Setting up Initiatives gallery

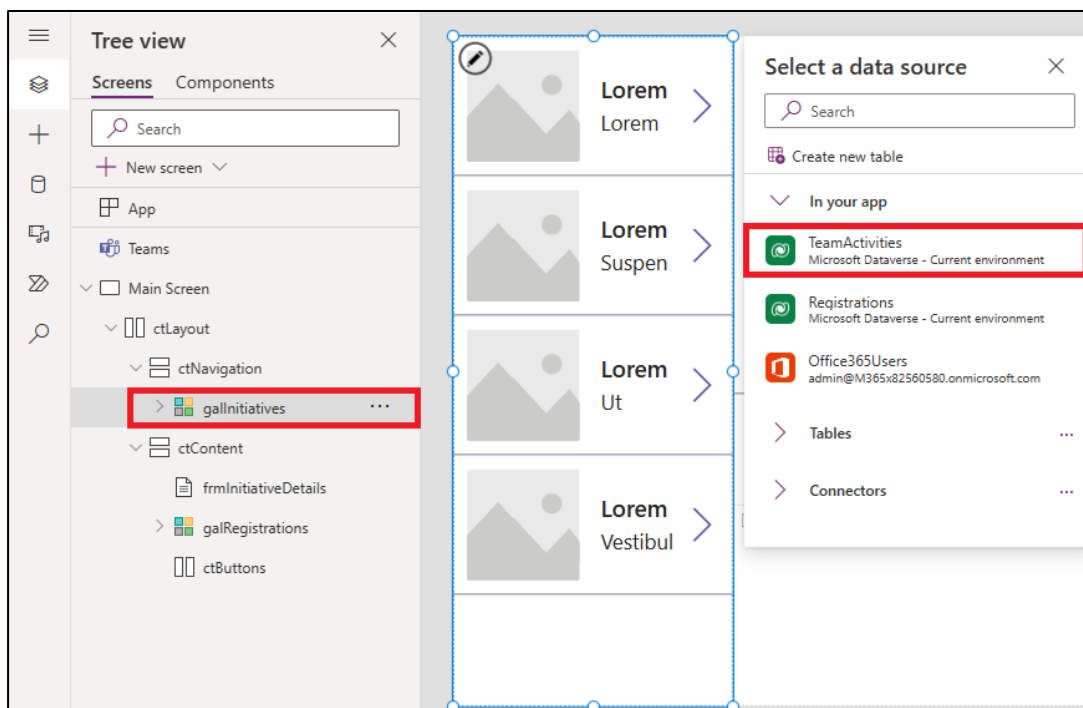


Gallery control

Gallery is one of the most useful Power Apps controls, when we are working with data or collections of objects.

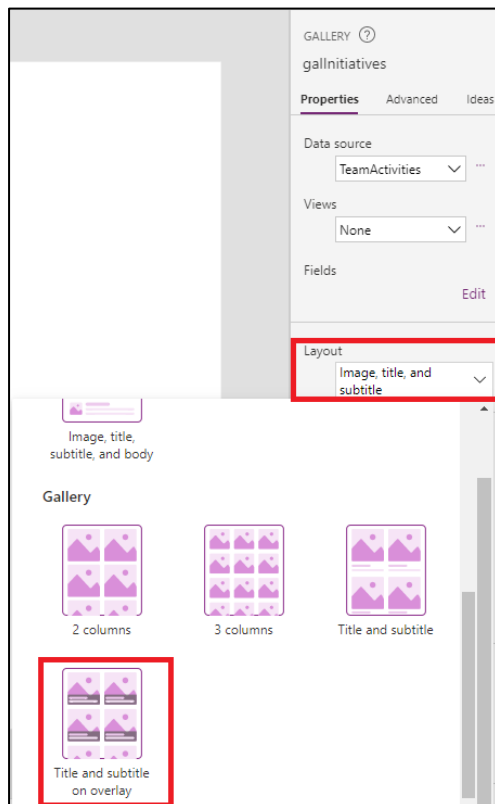
Gallery displays the repetitive data, based on pre-defined template.
Template can contain not only labels but any controls that you want.

1. Select **galInitiatives** and connect it to **TeamActivities** data source



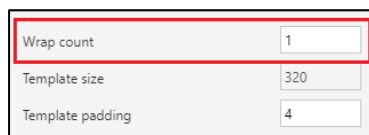
- Let's use some pre-defined gallery templates.

In **Properties** pane, change **Layout** property, set it as *"Title and subtitle on overlay"*



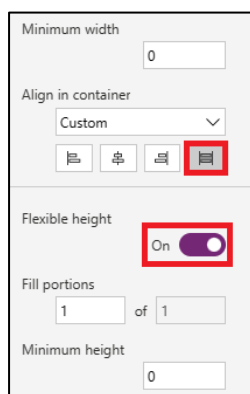
- Change one more property for **gallInitiatives**, set **Wrap count** property as **1**


You can also modify template size (height) and template padding if you want.

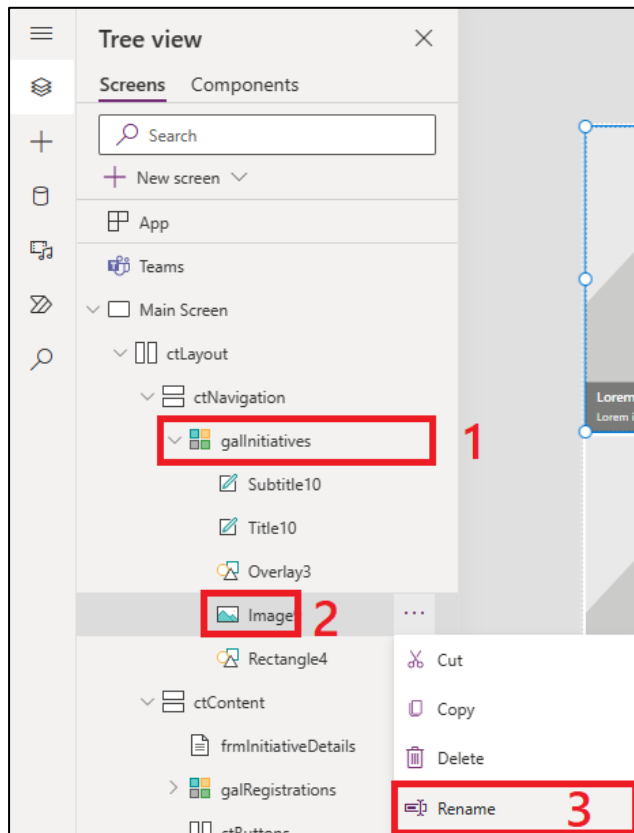


Set **Flexible Height** property as **On**

Set **Width** as Align in container -> **Stretch**

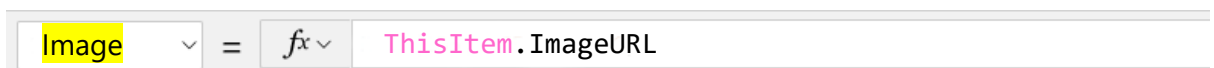


4. In **Tree View**  expand **galInitiative** and select an image within it, rename it as **imgInitiative**



Controls within gallery are parts of its template.

5. For **imgInitiative**, set **Image** property as `ThisItem.ImageURL`

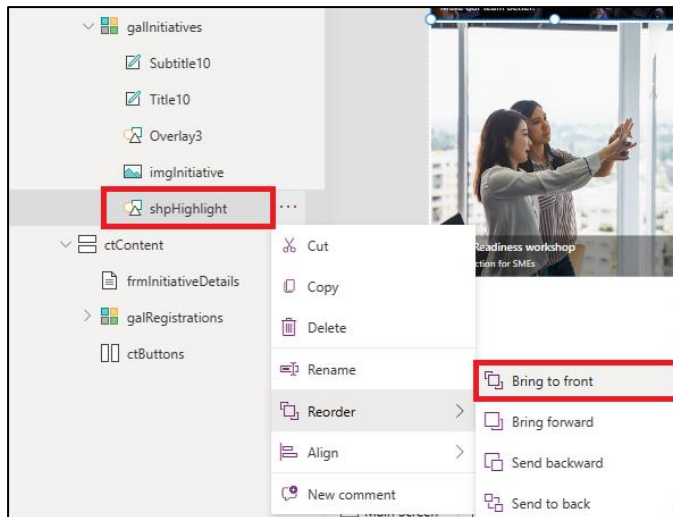


If the images are still not displayed in the gallery – double-check ImageURL links in your TeamActivities table.

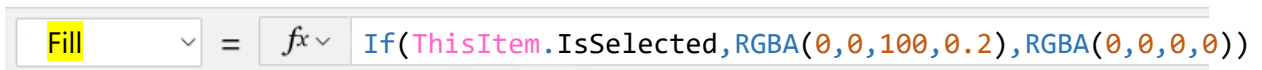
6. In **Tree View**  expand **galInitiative** and select a Rectangle within it.

Rename it as **shpHighlight**

Use **Reorder -> Bring to front** to put this invisible rectangle on top of gallery template.



- For **shpHighlight**, set **Fill** property as [\(go to localized formula\)](#)



Conditions and conditional formatting

If(condition, then, else) – works in the same way as in Excel.

In this case, we are using it to implement conditional formatting.

If the initiative is selected, then it should be highlighted (fill of rectangle changed).
Technically, you can apply this method to any property of the control.

RGBA(red, green, blue, alpha) - function defines colour.

RGBA(0, 0, 100, 0.2) – transparent light blue colour

RGBA(0, 0, 0, 0) – absolutely invisible black colour

Now, try to Alt-click on initiatives in gallery and colour should slightly change if the item is selected.

Don't forget to save your app. Task is completed.

Task 4: Setting up Initiative display form

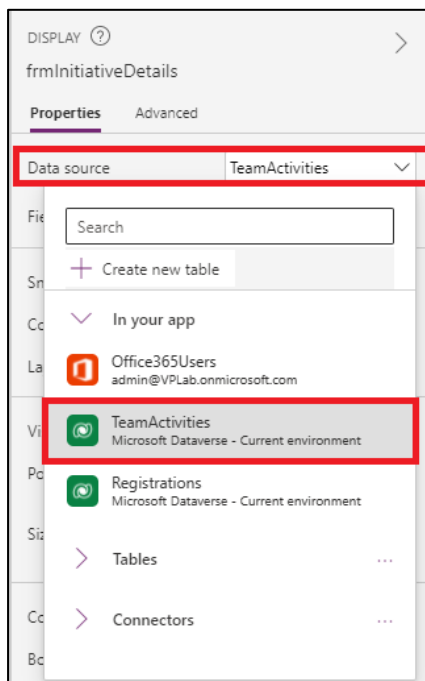


Displaying complex data

Using **Display Form** is just one of the ways to display data. It's a quick way to automatically generate datacards based on your data source. We'll use it to save time. On the other hand, display forms are far from being flexible.

Alternatives: single item gallery OR manually built custom group of controls.

1. Select **frmInitiativeDetails** and set **Data source** property as **TeamActivities**



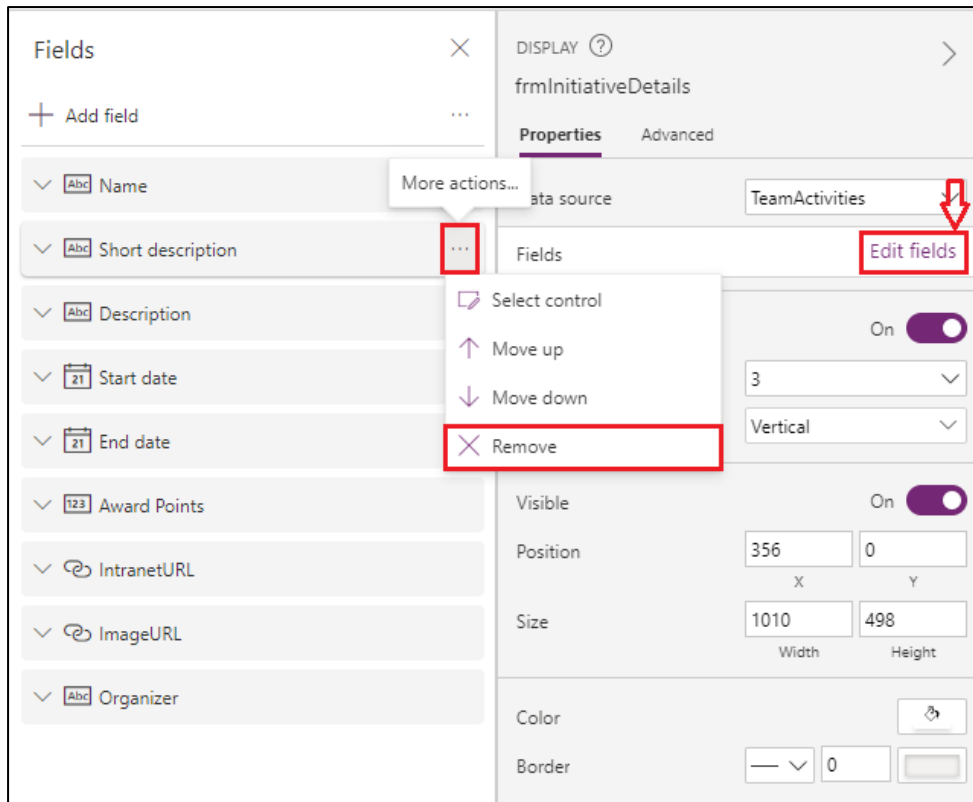
2. Let's hide some fields, that we don't want to display to users.

Click **Edit fields** in **Property** pane.

This interface can be used to add, reorder or remove form fields.

Remove **Short description**, **ImageURL** and **Organizer** from the form, if they are present.

If any fields are missing, then add them using **+Add field** button on top.



3. Change **Columns** property to **2** and resize form elements to make it look nicer.

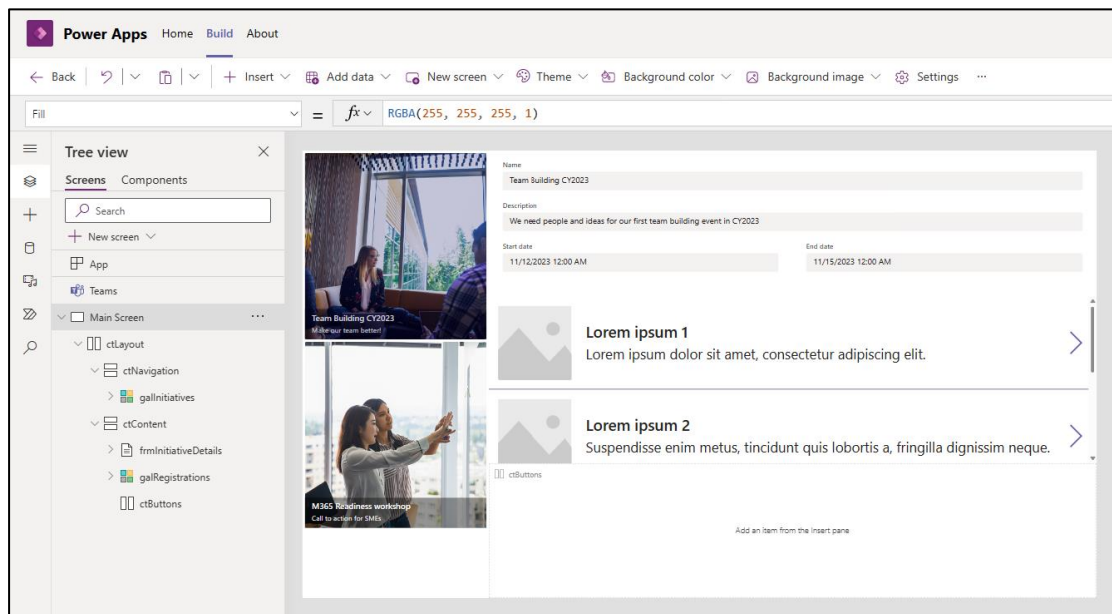


4. For **frmInitiativeDetails**, set **Item** property as



Form will display Initiative details of currently selected item in **galInitiatives**.

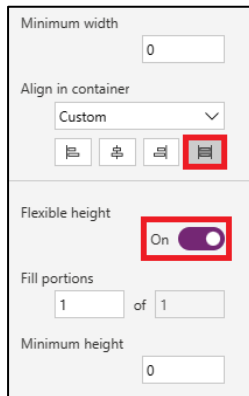
5. Your app may look like this:



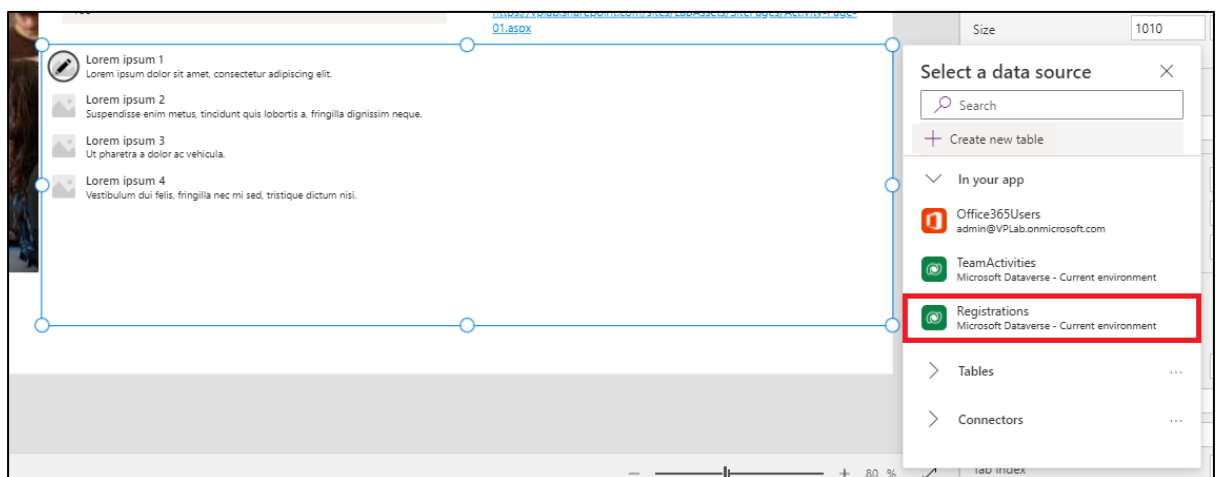
Don't forget to save your app. Task is completed.

Task 5: Setting up Registrations gallery

1. Select **galRegistrations** and set **Layout** property as **Image, title, and subtitle**
2. Make sure that **Flexible height** property is On and **Width** set as **Stretch**



3. Select **Registrations** table as a data source for it



4. Resulting gallery doesn't look useful.
Let's modify look and feel first and then we will filter data properly.

Expand **galRegistrations** in **Tree View**  and select title label (e.g., Title3).

Rename it as **lblUserName**

5. For **lblUserName**, check that **Text** property is



6. Select subtitle label in **galRegistrations** (e.g., Subtitle3).

Rename it as **lblRegisteredOn**.

7. For **lblRegisteredOn**, set **Text** property as

Text = **fx** "Registered on: " & **ThisItem**.'Created On'



Text concatenation

Ampersand & is a concatenation operator (glues two strings together), when used between text values in Power Apps formula language.

In this case we start with static text "Registered on: " in double-quotes and then we add a dynamic value **ThisItem**.'Created On'

(single quotes are used when column name contains spaces)

8. Select image in **galRegistrations** (e.g., Image3).

Rename it as **imgUserPhoto**.

9. For **imgUserPhoto**, set **Image** property as [\(go to localized formula\)](#)

Image = **fx** If(
 !IsBlank(**ThisItem**.UserEmail),
 Office365Users.UserPhotoV2(**ThisItem**.UserEmail)
)



Using connector actions in Power Apps

Office365Users represents data connector, that we added to this app in the beginning.

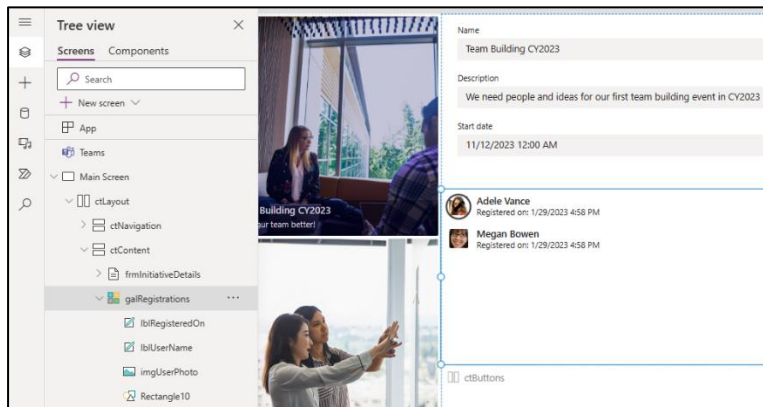
It allows to execute actions in the source system, just like we do in Power Automate flows.


Office365Users.UserPhotoV2(<User Id or UPN>) returns user photo.

We also need to check if **UserEmail** is blank before getting photo to avoid an error.

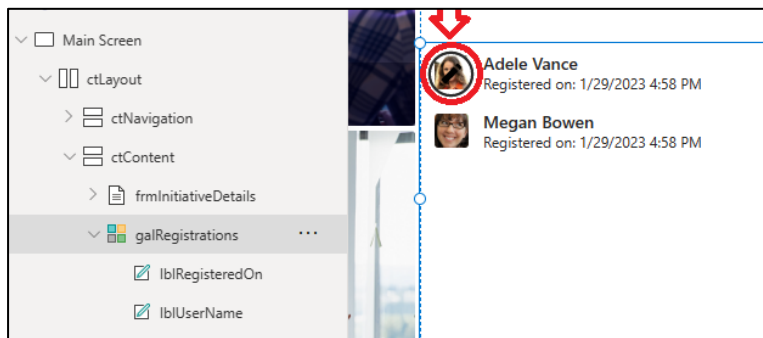
You may find many other uses to this connector, when you need to search for people, get profile details or traverse organizational structure.

10. **galRegistrations** should look like this now:

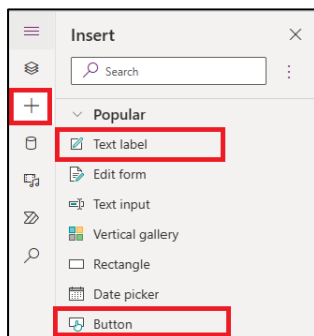


11. Select **galRegistrations** and then click **Edit gallery** icon .


Note: May be not clearly visible, because it's right on top of Adele's photo

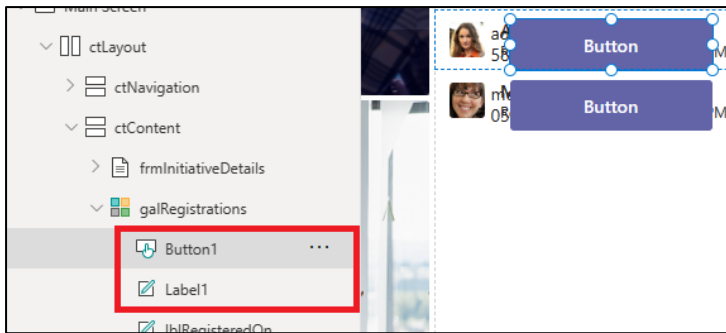


Switch to **Insert**  pane and add a **Text Label** and a **Button** to the template.



Add the button in the same way.

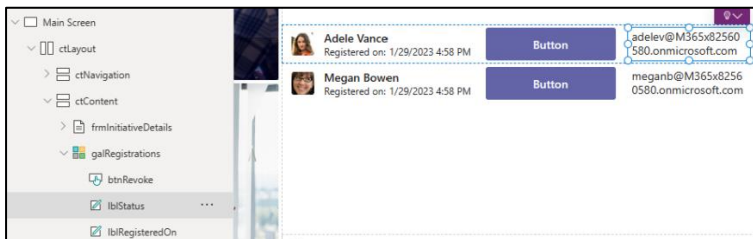
12. Open **Tree View**  and make sure that label and button are added **WITHIN galRegistrations**, not outside of it.



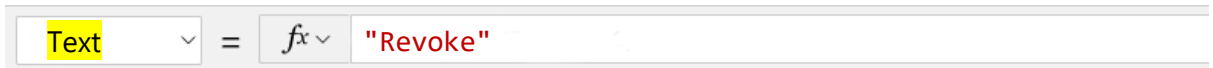
If the controls were created outside of gallery, you won't see them replicated for each gallery item – just recreate them **WITHIN** the gallery.

13. Rename controls and move them within the template

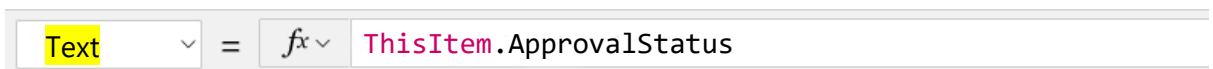
Button1 -> btnRevoke and **Label1 -> lblStatus**



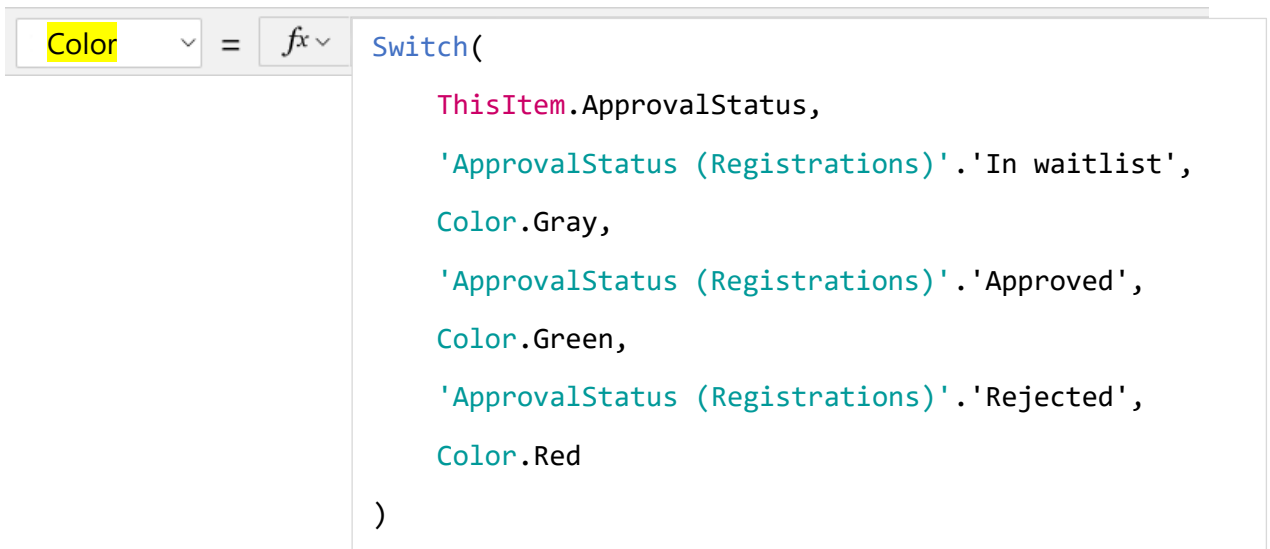
14. For **btnRevoke**, set **Text** property as



15. For **lblStatus**, set **Text** property as



16. For **lblStatus**, set **Color** property as [\(go to localized formula\)](#)



Let's use **Switch**, instead of, **If** this time.

This formula will enable conditional formatting for **ApprovalStatus** choices.



Enumerations and Dataverse Choices data type

You can notice, that instead of **RGBA()** function, this time colours defined differently.

Color.Green – we use built-in **Color** enumeration, that can be easier sometimes, then defining **RGBA()** parameters or HEX-code in **ColorValue()** function.

Power Apps has many built-in enumerations, here are some examples:

Color – pre-defined colours (e.g., **Color.Red**);

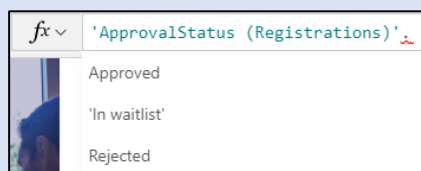
DisplayMode – control modes (e.g., **DisplayMode.Disabled** or **DisplayMode.Edit**)

ScreenTransition – screen navigation animations (e.g., **ScreenTransition.Cover**)

Dataverse **Choices** data types are represented as local or global enumerations that you defined, when you created a column of this data type.

Condition **ThisItem.ApprovalStatus = "In waitlist"** will not work, because of type mismatch.

Use **'ApprovalStatus (Registrations)'** local enumeration instead. It became automatically available, when you added **Registrations** table as a data source:



Condition

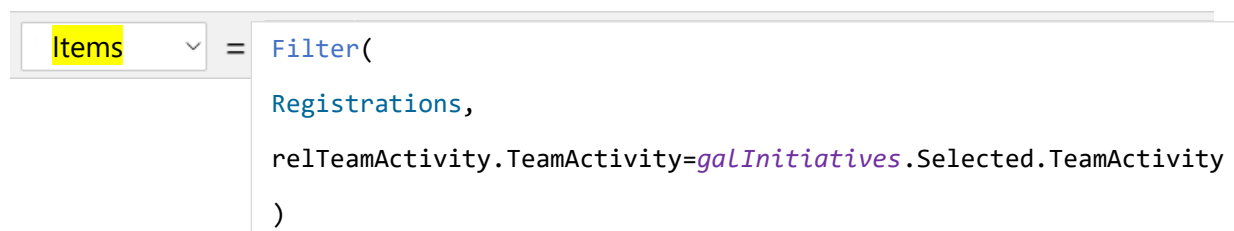
ThisItem.ApprovalStatus = 'ApprovalStatus (Registrations)'. 'In waitlist' will work correctly.

For global choices, you can just use **ApprovalStatus** enumeration instead without the table name in brackets.

17. There is a **relationship** between **Registrations** and **TeamActivities** tables, that was created in the previous exercise.

We can utilize it to easily **filter** registrations for the selected Initiative only.

For **galRegistrations**, set **Items** property as [\(go to localized formula\)](#)



Filtering query will be **delegated** to Dataverse for Teams database – it allows to achieve good performance, even on large data sets.

18. Try to navigate through Initiatives by Alt+Clicking on **galInitiatives**.

You will see error messages and some controls displayed, even if the gallery is empty.



Handling gallery display errors

Generally, there are two ways to resolve gallery display errors, that we're facing here:

1) Perform error handling and value validation for each control in gallery template, hide if empty or display a placeholder (more robust way)

It will help to handle any data in gallery no matter of data quality.

2) Hide gallery when it's empty (quick)

Doesn't resolve any data display issues – just hides empty gallery.

Let's imagine that we trust our data source quality and will just hide an empty gallery.

For **galRegistrations**, set **Visible** property as [\(go to localized formula\)](#)


Visible	▼	=	fx ▼	<code>If(!IsEmpty(galRegistrations.AllItems), true)</code>
---------	---	---	---------------	--

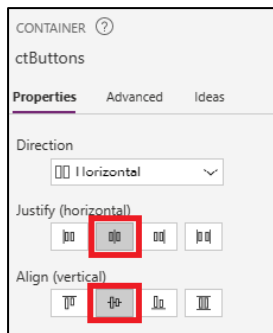
Once again, try to navigate through Initiatives by ALT+Clicking on **galInitiatives** – nothing will be displayed for Initiatives without Registrations.

Don't forget to save your app.

Task is completed.

Task 6: Add and delete Registrations

1. Select **ctButtons** container and from **Insert**  pane add a **Button** into it
2. Select **ctButtons** container
Set **Justify (horizontal)** property as **Center**
Set **Align (vertical)** property as **Center**



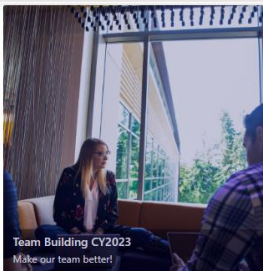

3. Select new button and rename it as **btnParticipate** and change button text to *"Participate"*
4. To add a new registration for the currently selected event, set **OnSelect** property of **btnParticipate** as [\(go to localized formula\)](#)



5. **Save** your app. Start **Preview** mode and try to click **btnParticipate**.

Give it a moment. Your user should appear in Registrations list for the currently selected event. Close **Preview** mode.

1 Your registration has been added.







Name
Team Building CY2023

Description
We need people and ideas for our first team building event in CY2023

Start date
11/12/2023 12:00 AM

End date
11/15/2023 12:00 AM

 Adele Vance Registered on: 1/29/2023 4:58 PM	Revoke	In waitlist
 Megan Bowen Registered on: 1/29/2023 4:58 PM	Revoke	Approved
 System Administrator Registered on: 1/29/2023 6:47 PM	Revoke	In waitlist

Participate

6. **btnRevoke** must be shown only for current user's registration, let's add a condition to its visibility. *Invisible buttons are not clickable.*
For **btnRevoke**, set **Visible** property as

Visible = $\text{Lower}(\text{ThisItem.UserEmail}) = \text{Lower}(\text{User}().\text{Email})$

Button should be visible only for current user now:

Tree view

Screens Components

Search

New screen

App

Teams

Main Screen

ctLayout

ctNavigation

ctContent

frmInitiativeDetails

galRegistrations

btnRevoke

lblStatus

lblRegisteredOn

lblUserName

imgUserPhoto

Rectangle10




ctButtons

btnParticipate

Name
Team Building CY2023

Description
We need people and ideas for our first team building event in CY2023

Start date
11/12/2023 12:00 AM

 Adele Vance Registered on: 1/29/2023 4:58 PM	In waitlist
 Megan Bowen Registered on: 1/29/2023 4:58 PM	Approved
 System Administrator Registered on: 1/29/2023 6:47 PM	Revoke In waitlist

7. To remove a registration for the currently selected event, set **OnSelect** property of **btnRevoke** as [\(go to localized formula\)](#)

```
OnSelect = Select(Parent);  
Remove(Registrations, ThisItem);  
Refresh(Registrations);  
Notify("Your registration has been revoked");
```

8. **Save** your app. Start **Preview** mode and try to click **btnRevoke** near your registration entry.

Give it a moment. Your user should disappear from Registrations gallery for the currently selected event. Close **Preview** mode.

9. We need to avoid adding the same user more than once.
For **btnParticipate**, set **DisplayMode** property as [\(go to localized formula\)](#)

```
DisplayMode = If(  
    Lower(User().Email) in  
    galRegistrations.AllItems.UserEmail,  
    DisplayMode.Disabled,  
    DisplayMode.Edit  
)
```

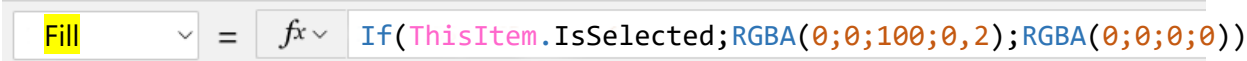
btnParticipate should be disabled now, if the current user is present in Registrations for the selected Initiative:



Save and **Publish** your app. **Task is completed.**

Appendix: Localized formulas

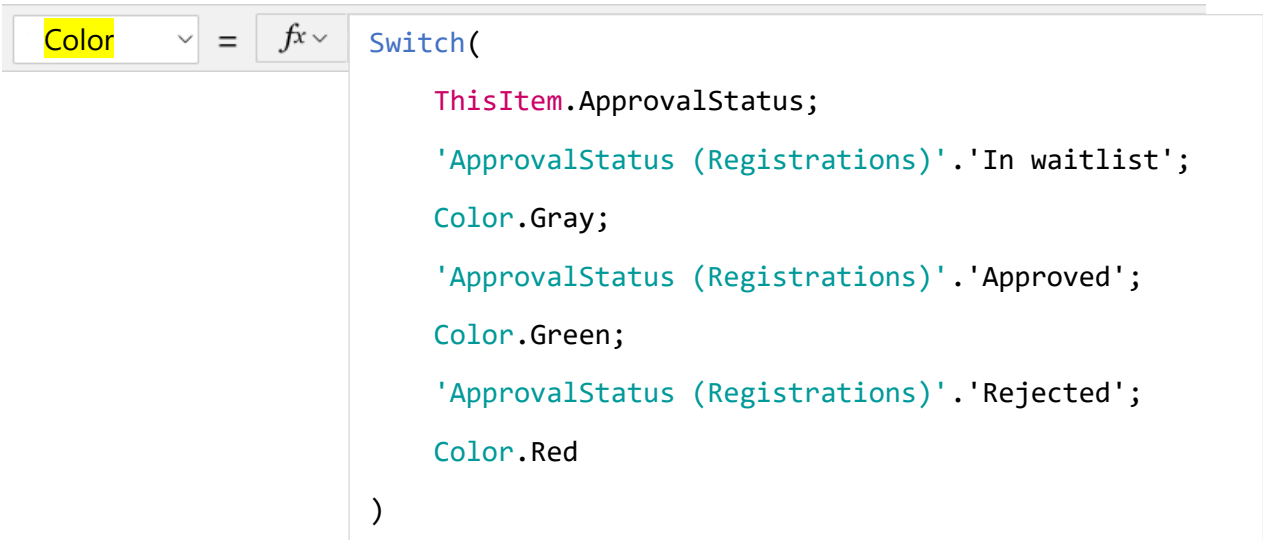
1. Exercise 2 – Task 3. Overlay colouring formula [\(go back\)](#)

The screenshot shows a formula bar with a dropdown menu set to 'Fill'. The formula entered is `= If(ThisItem.IsSelected; RGB(0;0;100;0,2); RGB(0;0;0;0))`.


2. Exercise 2 – Task 5. Getting user's photo [\(go back\)](#)

The screenshot shows a formula bar with a dropdown menu set to 'Image'. The formula entered is `= If(!IsBlank(ThisItem.UserEmail); Office365Users.UserPhotoV2(ThisItem.UserEmail))`.


3. Exercise 2 – Task 5. Approval status conditional colouring [\(go back\)](#)

The screenshot shows a formula bar with a dropdown menu set to 'Color'. The formula entered is `= Switch(ThisItem.ApprovalStatus; 'ApprovalStatus (Registrations)'. 'In waitlist'; Color.Gray; 'ApprovalStatus (Registrations)'. 'Approved'; Color.Green; 'ApprovalStatus (Registrations)'. 'Rejected'; Color.Red)`.

4. Exercise 2 – Task 5. Filtering Registrations [\(go back\)](#)

The screenshot shows a formula bar with a dropdown menu set to 'Items'. The formula entered is `= Filter(Registrations; relTeamActivity.TeamActivity=galInitiatives.Selected.TeamActivity)`.

5. Exercise 2 – Task 5. Hiding empty gallery [\(go back\)](#)

The screenshot shows a formula bar with a dropdown menu set to 'Visible'. The formula entered is `= If(!IsEmpty(galRegistrations.AllItems); true)`.

6. Exercise 2 – Task 6. Patching data to Dataverse table [\(go back\)](#)

```
OnSelect = Patch(
    Registrations;
    Defaults(Registrations);
    {
        Name: User().FullName;
        userEmail: Lower(User().Email);
        relTeamActivity: galInitiatives.Selected;
        ApprovalStatus:
        'ApprovalStatus (Registrations)'.In waitlist'
    }
);;
Refresh(Registrations);;
Notify("Your registration has been added.");;
```

7. Exercise 2 – Task 6. Removing data from Dataverse table [\(go back\)](#)

```
OnSelect = Select(Parent);;
Remove(Registrations; ThisItem);;
Refresh(Registrations);;
Notify("Your registration has been revoked");;
```

8. Exercise 2 – Task 6. Conditional DisplayMode for button [\(go back\)](#)

```
DisplayMode = If(
    Lower(User().Email) in
    galRegistrations.AllItems.UserEmail;
    DisplayMode.Disabled;
    DisplayMode.Edit
)
```