

Class & Module	IU 03: Get started with Power Apps
Name	I Wayan Wipram Sembah Klan Gamurci
IC No.	
Date & Time	Mon Apr 11 2022 21:40:38 GMT+0800 (Central Indonesia Time)

Lab: How to build a canvas app, Part 1

Scenario

Bellows College is an educational organization with multiple buildings on campus. Campus visits are currently recorded in paper journals. The information is not captured consistently, and there are no means to collect and analyze data about the visits across the entire campus.

Campus administration would like to modernize their visitor registration system where access to the buildings is controlled by security personnel and all visits are required to be pre-registered and recorded by their hosts.

Throughout this course, you will build applications and perform automation to enable the Bellows College administration and security personnel to manage and control access to the buildings on campus.

In part 1 this lab, you will design a Power Apps canvas app that college staff can use to manage visits for their guests.

High-level lab steps

We will follow the below outline to design the canvas app:

- Create the app from data using the phone form factor template
- Configure a detail page with visit info
- Configure an edit page to create to visits
- Configure a gallery control to show the visits
- Add filtering on the gallery data source to show only future visits

Prerequisites

- Completion of **Module 0 Lab 0 - Validate lab environment**
- Completion of **Module 2 Lab 1 - Introduction to Microsoft Dataverse**

Things to consider before you begin

- What is the most prevalent form factor for the target audience?
- Estimate the number of records in the system
- How to narrow the records selected to improve app performance and user adoption

Exercise #1: Create Staff Canvas App

Objective: In this exercise, you will create a canvas app from a template and then modify it to include required data.

Task #1: Create Canvas App

In this task, you will create a canvas app using the phone layout template based on Microsoft Dataverse. Using Visits as a selected table from Dataverse, the template will generate a Gallery - View - Edit app to manage campus visits.

1. Start creating an app from data
 - Sign in to <https://make.powerapps.com>
 - Select your **environment** at the top right if it is not already set to your Practice environment.
 - Select the **Dataverse** icon within **Start from data** on the Home screen.
2. Connect to your Visits table
 - Select **+ New connection**
 - Select **Microsoft Dataverse** and click **Create**
 - Locate and select your **Visits** table
 - Select **Connect**
3. The **Welcome to Power Apps Studio** window may appear. Click **Skip**.
4. Save application
 - Click **File > Save**.
 - Enter [Your Last Name] Campus Staff as the **App name**.
 - Press **Save**.

Task #2: Configure Visits Detail Form

In this task, you will configure the Detail form to view information about individual visit records.

1. Select the **Back** arrow at the top left to go back to the app definition.
2. Expand **DetailScreen1** under **Tree view**
3. Select **DetailForm1**
4. Select **Edit fields** next to **Fields** in the right-hand panel.
5. Click **Add field**

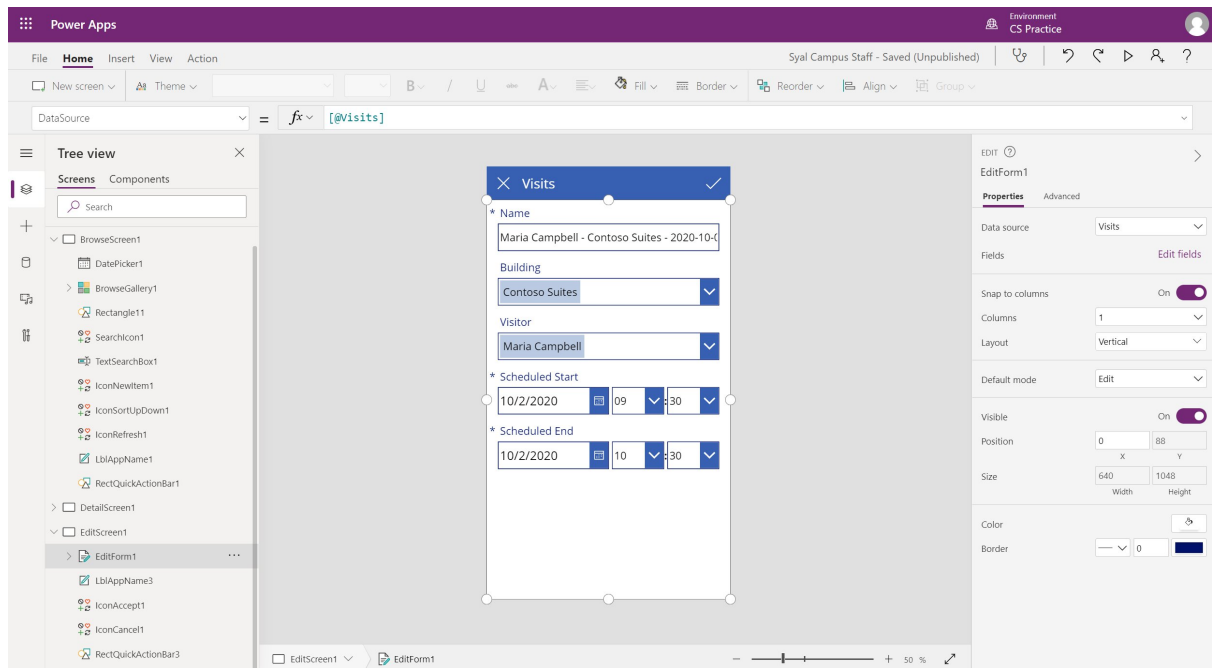
6. Select the following fields:
 - Actual End
 - Actual Start
 - Building
 - Code
 - Scheduled End
 - Scheduled Start
 - Visitor
 7. Click **Add**
 8. Rearrange fields in the **Fields** pane by dragging and dropping field names up or down. Recommended order is:
 - Code, Name, Building, Visitor, Scheduled Start, Scheduled End, Actual Start, Actual End
- Tip:** You can collapse each field by clicking the down arrow beside the field name.
9. Remove the **Created On** field by clicking the ellipses (...) beside the field name and selecting **Remove**.
 10. Close the **Fields** pane.
 11. To preserve work in progress, click **File** then click **Save**. Use the back arrow to return to the app.

Task #3: Configure Visits Edit Form

In this task, you will configure a form to edit information about individual visit rows.

1. Expand **EditScreen1** under **Tree view**
 2. Select **EditForm1**
 3. Select **Created On** field and press **Del** key to remove the field
 4. Select **Edit fields** in the properties panel
 5. Click **Add field**
 6. Select the following fields:
 - Building
 - Scheduled End
 - Scheduled Start
 - Visitor
 7. Click **Add**
 8. Rearrange fields in the **Fields** pane by dragging and dropping field names up or down. Recommended order is:
 - Name, Building, Visitor, Scheduled Start, Scheduled End
- Tip:** You can collapse each field by clicking the down arrow beside the field name.
9. Close the **Fields** pane.
 10. To preserve work in progress, click **File** then click **Save**. Use the back arrow to return to the app.

Your screen should look approximately like the following:



Task #4: Configure Visits gallery

In this task, you will configure the pre-generated gallery to display the title, start date and end date for the visit.

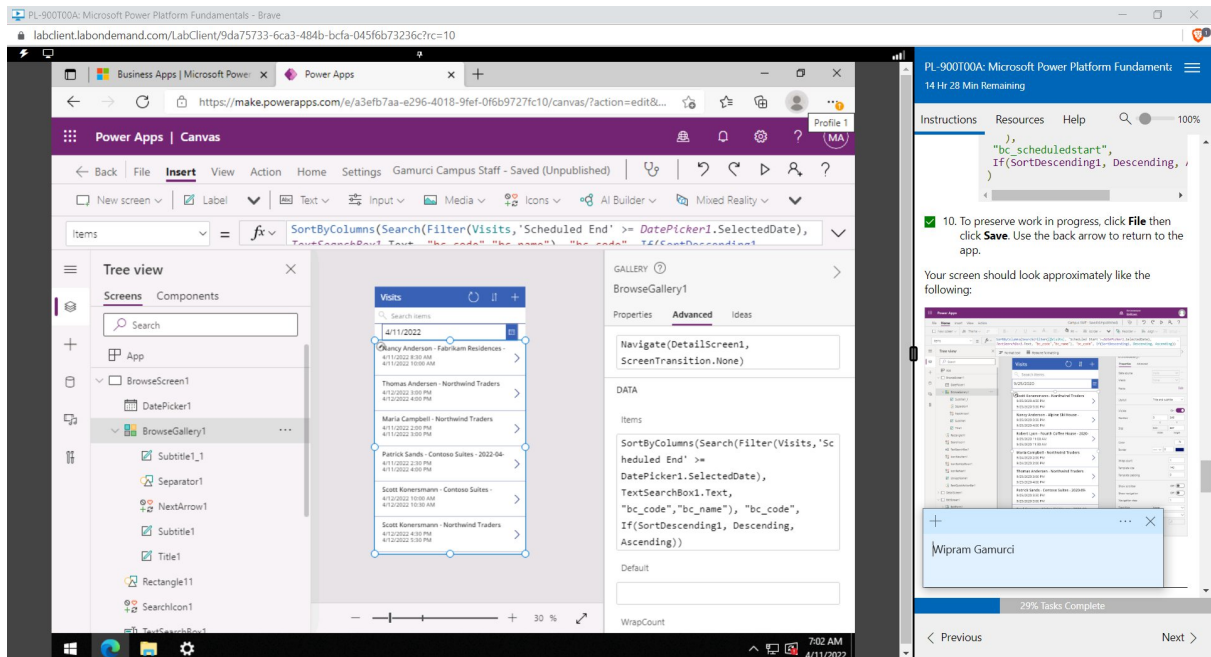
1. Expand **BrowseScreen1** under **Tree view**
2. Select **BrowseGallery1**
3. Select **TemplateSize** property from in the Advanced Properties panel on the right
4. Replace the expression with the following `Min(150, BrowseGallery1.Height - 60)`.
That will ensure sufficient space for additional information.
5. In the app preview, select the first Date Time field in the gallery.
6. In the formula bar at the top, change `**ThisItem.'Created On'*` to `ThisItem.'Scheduled Start'`
7. Select the field again
8. Press **CTRL-C** then **CTRL-V** to create a copy of the field.
9. Using either mouse or keyboard, move the copied control down and align it with the other controls in the gallery, beneath the other Date Time field.
10. In the formula bar at the top, change `ThisItem.'Scheduled Start'` to `ThisItem.'Scheduled End'`
11. To preserve work in progress, click **File** then click **Save**. Use the back arrow to return to the app.

Task #5: Add date filter

Because number of visits continuously grows, users need a feature to filter the visits gallery. For example, the user may want to see only the future visits. In this task, you will add ability to show visits only after a date selected by the user.

1. Select **BrowseScreen1**
2. Select **Insert** menu at the top.
3. Click **Input** and select **Date picker**.
4. Using either keyboard or mouse, position the control below the search box.
5. Select **BrowseGallery1**
6. Resize and move the gallery control so that it is located under the date picker and covers the screen. You can do this by clicking the resize icon at the top center of the gallery control and resizing the control to start after the date picker.
7. With **BrowseGallery1** selected, click the **Advanced** tab of the Properties pane.
8. Locate the **Items** property and click in the text box.
9. In the expression, locate **[@Visits]** and replace it with `Filter(Visits, 'Scheduled End' >= DatePicker1.SelectedDate)`. The full expression should look like the following:
 10. `SortByColumns(`
 11. `Search(`
 12. `Filter(`
 13. `Visits,`
 14. `'Scheduled End' >= DatePicker1.SelectedDate`
 15. `),`
 16. `TextSearchBox1.Text,`
 17. `"bc_code", "bc_name"`
 18. `),`
 19. `"bc_scheduledstart",`
 20. `If(SortDescending1, Descending, Ascending)`
 21. `)`
21. To preserve work in progress, click **File** then click **Save**. Use the back arrow to return to the app.

Create a screenshot showing the above results



Exercise #2: Complete the App

In this exercise you will test the application and, once successful, you will add it to your solution.

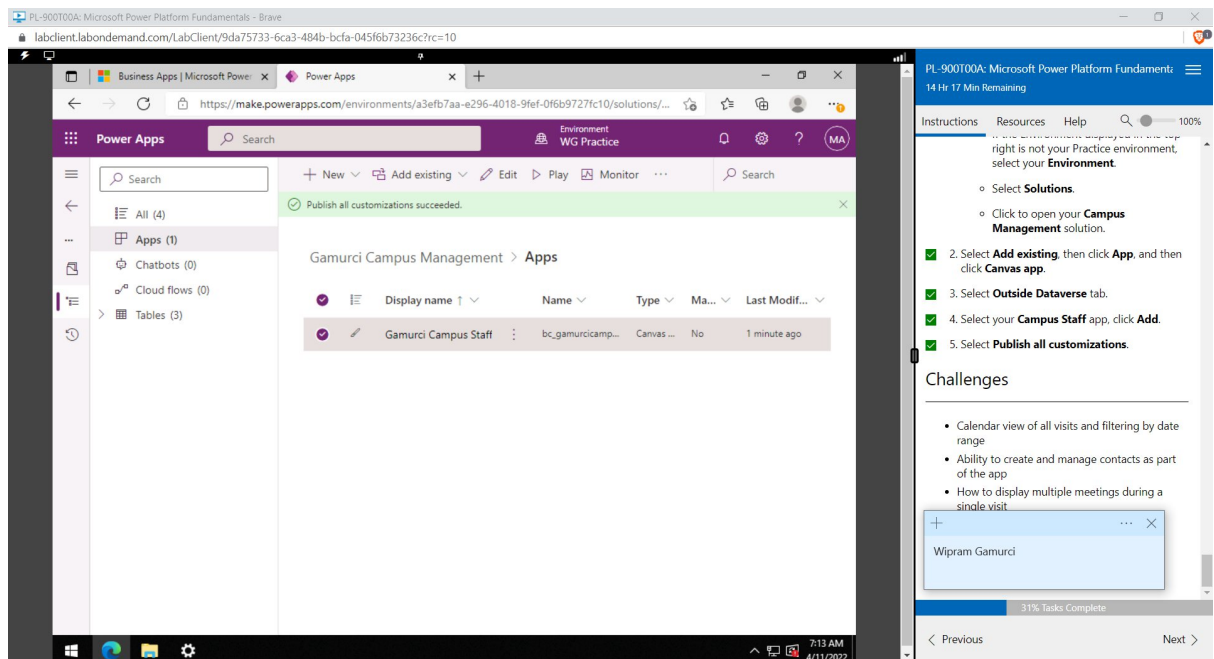
Task #1: Test App

1. Start the application
 - Select the **BrowseScreen1** and press Function **F5**, or click the **Play** icon at the upper-right corner to preview the app.
 - The application should load and show a list of visits.
 - Test the filter by selecting different dates in the date picker control
 - Select a visit and verify that display form is working properly
 - Return to the gallery and press **+** to create a new visit. Verify that edit form contains required columns including visitor, building, and scheduled start and end dates.
 - Fill in the information and submit. Verify that the new record appears in the gallery.
 - Create at least 2 more visits.
 - Press **ESC** key or click the **X** icon to close preview mode.
2. Save and publish the application
 - Click **File** and, if the Save button is displayed, click **Save**.
 - Click **Publish**.
 - Click **Publish this Version**.
 - Click the **Back** arrow to navigate back to the app.
 - Close the **Designer** browser window or tab.
 - Click **Leave** if prompted when tried to close the browser window.

Task #2: Add App to Solution and publish

1. Open the Campus Management solution.
 - o Sign in to <https://make.powerapps.com>
 - o If the Environment displayed in the top right is not your Practice environment, select your **Environment**.
 - o Select **Solutions**.
 - o Click to open your **Campus Management** solution.
2. Select **Add existing**, then click **App**, and then click **Canvas app**.
3. Select **Outside Dataverse** tab.
4. Select your **Campus Staff** app, click **Add**.
5. Select **Publish all customizations**.

Create a screenshot showing the above results



Lab 2: How to build a canvas app, part 2

Scenario

Bellows College is an educational organization with multiple buildings on campus. Campus visits are currently recorded in paper journals. The information is not captured consistently, and there are no means to collect and analyze data about the visits across the entire campus.

Campus administration would like to modernize their visitor registration system where access to the buildings is controlled by security personnel and all visits are required to be pre-registered and recorded by their hosts.

Throughout this course, you will build applications and perform automation to enable the Bellows College administration and security personnel to manage and control access to the buildings on campus.

In part 2 of this lab, you will create design and build a Power Apps canvas app that the security personnel will use at the building entrances to quickly confirm and register the visitors.

High-level lab steps

You will follow the below outline to design the canvas app:

- Create the app using the phone form factor
- Connect to Dataverse as a data source
- Capture the input (visitor code) and locate the visitor row
- Configure a form viewer control to show the visitor information
- Use a Dataverse view to populate the gallery
- Handle checking-in and checking-out process for a visitor

Prerequisites

- Completion of **Module 0 Lab 0 - Validate lab environment**
- Completion of **Module 2 Lab 1 - Introduction to Microsoft Dataverse**

Things to consider before you begin

- What information would a security officer need quick access to?
- What should happen if visitor code is invalid?
- What should happen if the visitor arrives outside of the scheduled hours?

Exercise #1: Create Security Canvas App

Objective: In this exercise, you will create a canvas app.

Task #1: Create Canvas App

1. Open your Campus Management solution.
 - Sign in to <https://make.powerapps.com>
 - If the Environment displayed in the top right is not your Practice environment, select your **Environment**.
 - Select **Solutions**.
 - Click to open your **Campus Management** solution.
2. Create new canvas application
 - Click **New** and select **App | Canvas App**.
 - In the Canvas app from blank window, enter **[Your Last Name] Campus Security** in the App name field.
 - Select **Phone** in the Format field.
 - Click **Create**. This will open the App Editor in a New window. Click **Skip** if presented with the Welcome to Power Apps Studio dialogue.
3. Save the canvas app
 - Click **File** and select **Save As**.
 - Check if **The cloud** is selected click **Save**.
 - Verify **[Your Last Name] Campus Security** for Name and click **Save**.
 - Click the **Back** arrow at the top left (below Power Apps) to return to the app.
4. Connect to data source (Visits)
 - Click **View | Data sources**
 - Click **+ Add Data**
 - Click **See all tables**
 - Select **Visits** and wait for the Visit table to display on the Data tab.
5. To preserve work in progress, click **File** then click **Save**. Use the back arrow to return to the app.

Task #2: Display Visitor information

1. Add search box
 - Select the **Tree View** tab on the left navigation bar.
 - Select **Screen1**.
 - Go to the **Insert** tab.
 - Click **Text** and select **Text input**.
2. Edit the text input object
 - While still selecting the Text input object, select the text in the **Default** property and clear the value.
 - Select **Hint Text** property and enter **"Enter visitor code"** as the value (including double quotes)
 - Click on ... next to the control name in tree view (TextInput1), select **Rename**, change the name to **textCode**
3. Add a form view

- On **Insert** tab click **Forms** then select **Display** (you may need to click the down arrow on the right side of the ribbon to see Forms)
 - Drag to position the form and align with the bottom of the screen
 - While still selecting the new form, select **DataSource** property and select **Visits**
 - In the properties pane select **Horizontal** as **Layout**
4. Edit form view
 - While still selecting the new form, click **Edit fields**
 - Remove both the **Name** and **Created On** fields
 - Click **Add field** and select the following fields: **Actual End, Actual Start, Building, Scheduled End, Scheduled Start, Visitor**
 - Press **Add**
 - Change the order of the selected fields by dragging the field cards in the list. Recommended order is: Visitor, Building, Scheduled Start, Scheduled End, Actual Start, Actual End (you can collapse the fields to make them easier to drag)
 - Click the **X** to close the Fields pane
 5. While still selecting the form view, select the Advanced tab on the Properties pane. Select **Item** property and enter `Lookup(Visits, Code = textCode.Text)`
 6. To preserve work in progress, click **File** then click **Save**. Use the back arrow to return to the app.
 7. Prepare to test the app
 - Switch to the browser tab containing the solution
 - Click **Done** in the pop-up window
 - Select **Visit** table
 - Select **Data** tab
 - Open the View Selector in the top right by clicking the current View name, **Active Visits**
 - Change the View to **All columns**
 - Locate a Visit row that does not have an Actual Start or Actual End value (i.e., both columns are blank). Select and copy the **Code** for this Visit.
 8. Test the app
 - Switch to the browser tab with the app, press **F5** or click the **Play** icon at the upper-right corner to preview the app.
 - Paste the copied value into the search textbox, verify that the record is displayed in the form
 9. Clear the search textbox contents.
 10. Press **ESC** to exit the running app.

Task #3: Add Check In and Check Out Buttons

In this task, we will create buttons for the user to check in and check out of their Visit.

1. Save search results in a variable to reuse across the control
 - Select **textCode** control
 - In the properties pane, select the **Advanced** tab and select **OnChange** property
 - Enter the following expression `Set(Visit, Lookup(Visits, Code = textCode.Text))`

This will save the visit in a global variable when a user searches in the textCode searchbox. That allows us to use the variable *Visit* throughout the app without the need to re-enter the entire lookup expression.

2. Add Check In Button
 - Select **Insert** tab
 - Click **Button**
 - In the properties pane, change the button **Text** property to "**Check In**" (you can type within the existing quotes)
 - Click on ... next to the button name in tree view (Button1), select **Rename**, change the name to **CheckInButton**
3. Add Check Out Button
 - Click **Button** on the Insert tab to insert another button
 - In the properties pane, change the button **Text** property to "**Check Out**" (you can type within the existing quotes)
 - Rename the button as **CheckOutButton**
 - Position the buttons below the search box, with **Check In** above **Check Out**

Task #4: Enable and disable buttons depending on visit data

Once users have looked up the visit, we would like them to use the Check In button to check in for that visit. We would like to enable **Check In** button when the visit record has been located (not blank), record status is active, and the visit has not started yet, i.e. the actual start value is blank.

1. Select the **Check In button** and click on the **Display Mode** property of the button in the Properties tab
2. Enter the expression below in the function bar:

```
3. If(!IsBlank(Visit)
4.  && Visit.Status = 'Status (Visits)'.Active
5.  && IsBlank(Visit.'Actual Start'),
6.    DisplayMode.Edit,
7.    DisplayMode.Disabled
)
```

The expression can be broken down as following:

- **!IsBlank(Visit)** - visit record was found
- **&&** - logical AND operator
- **Visit.Status = 'Status (Visits)'.Active** status of the record is *Active*
- **IsBlank(Visit.'Actual Start')** - Active Start field does not have any data in it
- **DisplayMode.Edit, DisplayMode.Disabled** - If the above conditions are met, the button will become editable. If not, the button will remain disabled.

We would like to enable **Check Out** button when the visit record has been located (not blank), record status is active, and the visit has already started, i.e. the actual start value is not blank.

1. Select the Check Out button and click on the **Display Mode** property of the button in the Properties tab
2. Enter the expression below in the function bar:

```
3. If(!IsBlank(Visit)
4.  && Visit.Status = 'Status (Visits)'.Active
5.  && !IsBlank(Visit.'Actual Start'),
6.    DisplayMode.Edit,
```

```
7.     DisplayMode.Disabled
8. )
```

8. To preserve work in progress, click **File** then click **Save**. Use the back arrow to return to the app.
9. Press **F5** to run the app.
10. Both buttons should be disabled. Enter the code value you copied previously and press **Tab** to move the focus away from the textbox (or click outside of the textbox). The **Check In** button should become enabled.
11. Clear the search box contents.
12. Press **ESC** to exit the running app.

Task #5: Complete Check In and Check Out Process

To perform the check in and check out process we need to update Dataverse visit data as following:

- When visitor checks in, set *Actual Start* field to the current date and time
- When visitor checks out, set *Actual End* field to the current date and time.
- After check out, set the record status to inactive, indicating that the visit has been completed

1. Select **Check In** button.
2. Set **OnSelect** property on the Advanced tab to the following expression.

```
3. Patch(
4.     Visits,
5.     Visit,
6.     {'Actual Start': Now()}
7. );
8. Refresh([@Visits]);
Set(Visit, LookUp(Visits, Code = textCode.Text));
```

This expression contains the following parts:

- **Patch(Visits, Visit, {'Actual Start': Now()});** *Patch* method updates **Visits** table, the row identified by **Visit** variable (which is the current visit). The expression sets the value of *Actual Start* column to the current date and time (*Now()* method).
- **Refresh([@Visits]);** This expression refreshes the visit rows, as the underlying values have changed
- **Set(Visit, LookUp(Visits, Code = textCode.Text));** This expression updates the *Visit* variable with fresh data from Dataverse.

When a user clicks this button, the Actual Start of the Visit will be set to the current date and time and the data will refresh.

1. Select **Check Out** button.
2. Set **OnSelect** property on the Advanced tab to the following expression:

```
3. Patch(
4.     [@Visits],
5.     Visit,
```

```

6.     {
7.         'Actual End': Now(),
8.         Status: 'Status (Visits)'.Inactive
9.     }
10.);
11.Refresh([@Visits]);
Set(Visit, LookUp(Visits, Code = textCode.Text));

```

When a user clicks this button, the Actual End will be set to the current date and time, the Status of the Visit will be set to Inactive, and the data will refresh.

12. To preserve work in progress, click **File** then click **Save**. Use the **Back** arrow to return to the app.
13. Press **F5** or click the Play button to run the app. Enter the code value you copied previously and press **Tab** to move the focus away from the textbox. The **Check In** button should become enabled.
14. Press **Check In** button. The following should happen:
 - **Actual Start** is set to the current date and time
 - **Check In** button is disabled
 - **Check Out** button is enabled
15. Press **Check Out** button.
 - **Actual End** is set to the current date and time
 - Both buttons are disabled
16. Clear the search box contents.
17. Press **ESC** to exit the running app.

Task #6: Add visual indicators

Usability of a mobile app significantly improves when visual indicators are provided. In this task, we will add an icon indicating if a visitor can be checked in or checked out.

1. Select **Insert** tab
2. Select **Icons | Add**. Select an Icon. At this point it does not matter which icon we select as we want the value to be dynamic.
3. Resize and place the icon to the left of the buttons
4. In the Advanced tab for the Icon, select **Icon** property (in the Design section) and enter the following expression

```

5. If(
6.     CheckInButton.DisplayMode = DisplayMode.Disabled
7. && CheckOutButton.DisplayMode = DisplayMode.Disabled,
8.     Icon.EmojiFrown,
9.     Icon.EmojiSmile
10.)

```

10. To preserve work in progress, click **File** then click **Save**. Use the **Back** arrow to return to the app.
11. Press **F5** to run the app. Enter the code value you copied previously and press **Tab** to move the focus away from the textbox. Verify the icon displays a frown emoji.
12. Find a different code value that has not been used before (it should not have an Actual Start or Actual End value).

You can navigate to the previous tab to copy another Code from one of the Visits you have created. You also have the option to run your **Campus Staff** app created previously to create new visit records. Verify the icon displays a smile emoji for this code.

Your running app should look approximately like the following:

2020-02-11-1182



Check In

Check Out

Visitor

Sidney Higa

Building

Fabrikam Residences

Scheduled Start 11/27/2020 4:00 PM

Scheduled End 11/27/2020 5:30 PM

Actual Start

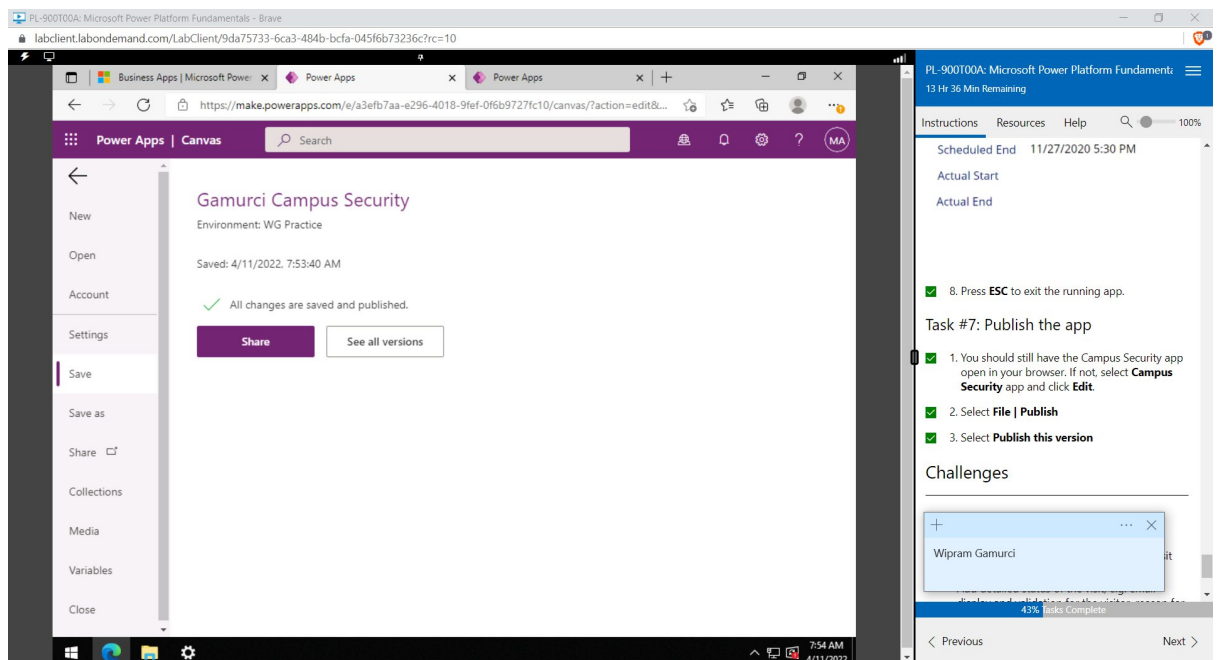
Actual End

1. Press **ESC** to exit the running app.

Task #7: Publish the app

1. You should still have the Campus Security app open in your browser. If not, select **Campus Security** app and click **Edit**.
2. Select **File | Publish**
3. Select **Publish this version**

Create a screenshot showing the above results



Lab 3: How to build a model-driven app

Scenario

Bellows College is an educational organization with multiple buildings on campus. Campus visitors are currently recorded in paper journals. The information is not captured consistently, and there are no means to collect and analyze data about the visits across the entire campus.

Campus administration would like to modernize their visitor registration system where access to the buildings is controlled by security personnel and all visits are required to be pre-registered and recorded by their hosts.

Throughout this course, you will build applications and perform automation to enable the Bellows College administration and security personnel to manage and control access to the buildings on campus.

In this lab, you will build a Power Apps model-driven app to allow the backoffice campus staff to manage visit records across the entire campus.

High-level lab steps

As part of creating the model-driven app, you will complete the following:

- Create a new model-driven app named Campus Management
- Edit the app navigation to reference the required tables
- Customize the forms and views of the required tables for the app

We will work with the following components:

- **Views:** Views allow the user to display the existing data in the form table.
- **Forms:** This is where the user creates/updates new rows in the tables.

Both will be integrated to the model-driven app for a better user-experience.

Prerequisites

- Completion of **Module 0 Lab 0 - Validate lab environment**
- Completion of **Module 2 Lab 1 - Introduction to Microsoft Dataverse**

Things to consider before you begin

- What changes should we make to improve the user experience?
- What should we include in a model-driven app based on the data model we have built?

- What customizations can be made on the sitemap of a model-driven app?

Exercise #1: Customize Views and Forms

Objective: In this exercise, you will customize views and forms of the custom created tables that will be used in the model-driven app.

Task #1: Edit Visit Form

1. Sign in to <https://make.powerapps.com> if you are not already signed in.
2. Select your **environment**.
3. Select **Solutions**.
4. Click to open your **Campus Management** solution.
5. Click to open the **Visit** entity.
6. Select the **Forms** tab and select the **Main** form type and click **Edit form**.

By default, the form has two fields: Name (Primary Field) and Owner.

7. Select **+ Form field** and add the following fields below the **Owner** field by dragging columns to the form or simply clicking column names:
 - **Building**
 - **Visitor**
 - **Scheduled Start**
 - **Scheduled End**
 - **Actual Start**
 - **Actual End**
8. Drag the **Code** column and drop it in the form header.

The header is the top right area of the form. You may need to minimize the Properties panel on the right side of the screen to see the field on the form.

9. With the **Code** field still selected, check the checkbox for **Read-only** in the Properties panel.
10. Select **Owner** field. In the Properties panel, change the **Field label** to **Host**
11. Click **Save** at the top right and wait for the save to complete.
12. Click **Publish** at the top right and wait for the publishing to complete.
13. If the edit view opened in a new tab, close the tab. Otherwise, click **Back** at the top left of the screen. You should now be back to the Visit entity Forms Tab.

Task #2: Edit Visit Views

In this task, we will modify the default Active Visits view and create a new view for today's visits.

1. Select the **Views** tab and click to open the **Active Visits** view.
2. Add the following fields to the view by either clicking or dragging and dropping the fields:
 - **Code**
 - **Visitor**
 - **Building**
 - **Scheduled Start**
 - **Scheduled End**
3. Click the **Created On** column and select **Remove**. Field **Created On** will now be removed from the view.
4. Click the **Name** column and select **Remove**. Field **Name** will now be removed from the view.
5. In the Properties panel on the right, click **Sort by ...** and select **Scheduled Start**. Click on **Scheduled Start** again to change the order to descending.
6. Resize the individual column widths to fit the data.
7. Click **Save** and wait until the changes are saved.
8. Click **Publish** and wait for the publishing to complete.

Now, we will clone the view to create a new view for today's visits.

1. Press **Edit filters** link in the Properties panel.
2. Click **Add**, select **Add row**.
3. Select **Scheduled Start** as a field, then select **Today** as the condition in the drop-down.
4. Click the **...** on the **Status** row and click **Delete**.
5. Press **Ok** to save the condition. The view is now filtered to show only records where the Scheduled Start date is today.
6. Add **Actual Start** and **Actual End** fields to the view.

Note: Since we no longer filter on the view status, we will get all today's visits including completed ones. These fields will help to differentiate completed visits and visits in progress.

7. Click on the **dropdown arrow** by the Save button (be careful not to press the button itself) and select **Save As**.
8. Change the name to **Today's Visits** and press **Save**.
9. Click **Publish** and wait for the publishing to complete.

Exercise #2: Create Model-Driven Application

Objective: In this exercise, you will create the model-driven app, customize the sitemap, and test the app.

You will see several fields not addressed as you build out your application, particularly on the sitemap steps. We have taken some short cuts in the interest of doing the labs. In a real implementation, you would give these items logical names.

Task #1: Create Application

1. Open your Campus Management solution if you are not already in it.
 - Sign in to <https://make.powerapps.com>
 - While in your environment, click to open your **Campus Management** solution.
2. Create the Model-Driven Application
 - Click **New** and select **App** and then **Model-driven app**.
 - In the Model-driven app from blanks screen, click **Create**.
 - Enter **[Your Last Name] Campus Management** for Name.
 - Select **Use existing solution to create the App** checkbox
 - Select **Next**
 - Select your **Campus Management** solution
 - Click **Done**
3. Click the pencil icon next to **Site Map**.
4. Edit the default titles
 - Select **New Area**.
 - Change the Title of the New Area to **Campus** in the properties pane on the right.
 - Select **New Group**.
 - Change the Title of the New Group to **Security** in the properties pane on the right.
5. Add the Contact table to the sitemap
 - Select **New Subarea**.
 - In the **Properties** pane, select **Entity** from the dropdown for **Type**.
 - Search for **Contact** table from the dropdown for **Entity**.
6. Add the Visit table to the sitemap
 - Select **Security** group and click **Add**.
 - Select **Subarea**.
 - Go to the **Properties** pane.
 - Select **Entity** from the dropdown for **Type** and search for **Visit** table from the dropdown for **Entity**.
7. Add the Building table to the sitemap
 - Select **Campus** area and click **Add**.
 - Select **Group**.
 - Enter **Settings** for **Title** in the **Properties** pane.
 - With the **Settings** group still selected, click **Add**.
 - Select **Subarea**.
 - Go to the **Properties** pane.
 - Select **Entity** from the dropdown for **Type** and search for **Building** table from the dropdown for **Entity**.

8. Click **Save**. This will show the loading screen while the changes are getting saved.
9. Click **Publish** to publish the sitemap and wait for the publishing to complete.
10. Click **Save and Close** to close the sitemap editor.

You will see the assets for the entities that were added to the sitemap are now in the application.

11. Click **Save** on the App Designer.
12. Click **Validate** to validate the changes done in the application.

This will show some warnings but we can ignore them, since we have not referenced a specific View and Form for the entities and the users will have access to all the Views and Forms for **Visit** and **Building** entities.

13. Click **Publish**
14. Click **Save and Close** to close the app designer.
15. Click **Done**.
16. Select **Solutions** and select **Publish all Customizations**.
17. Select **Apps** and your application should now be listed.

Task #2: Test Application

1. Start the application
 - Select **Apps** and click on your **Campus Management** app. (If you don't see your app at first, you may need to refresh your browser.)
 - The application should open in a new window.
2. Create new Contact
 - The app should open to the **Active Contacts** view
 - Click **New** from the top menu.
 - Provide **First Name** as **John** and **Last Name** as **Doe**.
 - Provide your personal email as **Email**. This will be used in a future lab.
 - Click **Save and Close**.
 - You should now see the created contact on the **Active Contacts** view.
3. Create new Building
 - Select **Buildings** from the sitemap.
 - Click **New**.
 - Enter the **Name** as **Microsoft Building**
 - Click **Save and Close**. This will show the newly created record on the Active Buildings View.
4. Create new Visit
 - Select **Visits** from the sitemap.
 - Click **New**.
 - Enter the fields as following
 - **Name:** **New test visit**

- **Building:** Alpine Ski House
 - **Visitor:** select John Doe
 - **Scheduled Start:** select today's date and 2:00 PM as start time
 - **Scheduled End:** select today's date and 3:30 PM as end time
- Click **Save and Close**. This will create the Visit and you should be able to see it on the Active Visits View.
 - Change view to **Today's Visits**. You should no longer see the new visit in the view, since it is scheduled for tomorrow.
5. You may add more test records.

Create a screenshot showing the above results

The screenshot displays a Microsoft Power Apps application titled 'Gamurci Campus Management'. The main view is 'Today's Visits', which shows a table of visit records. The table has columns for Code, Visitor, Building, Scheduled Start, Scheduled End, and Actual Start. The records are filtered for today's date (4/12/2022). A search bar is visible at the top right of the table.

Code	Visitor	Building	Scheduled Start	Scheduled End	Actual Start
2022-10-04-1...	Susanna Stubberod..	Contoso Suites	4/12/2022 6:00 ...	4/12/2022 7:00 ...	---
2022-10-04-1...	Thomas Andersen	Fourth Coffee House	4/12/2022 5:00 ...	4/12/2022 5:30 ...	---
2022-10-04-1...	Scott Konersmann	Northwind Traders Tower	4/12/2022 4:30 ...	4/12/2022 5:30 ...	---
2022-10-04-1...	Nancy Anderson	Alpine Ski House	4/12/2022 3:30 ...	4/12/2022 4:00 ...	---
2022-10-04-1...	Patrick Sands	Contoso Suites	4/12/2022 3:30 ...	4/12/2022 5:00 ...	---
2022-10-04-1...	Thomas Andersen	Northwind Traders Tower	4/12/2022 3:00 ...	4/12/2022 4:00 ...	---
2022-12-04-1...	John Doe	Alpine Ski House	4/12/2022 2:00 ...	4/12/2022 3:00 ...	---
2022-10-04-1...	Robert Lyon	Fourth Coffee House	4/12/2022 11:00...	4/12/2022 11:30...	---
2022-10-04-1...	Scott Konersmann	Contoso Suites	4/12/2022 10:00...	4/12/2022 10:30...	---
2022-12-04-1...	John Doe	Microsoft Building	4/12/2022 1:00 ...	4/12/2022 1:30 ...	---

The right side of the image shows a portion of a presentation slide titled 'PL-900T00A: Microsoft Power Platform Fundamentals'. It contains instructions for creating a visit record and a screenshot of the application interface. The slide also mentions '56% Tasks Complete' and has 'Previous' and 'Next' navigation buttons.

Lab 4: How to build a Power Apps portal

Scenario

Bellows College is an educational organization with multiple buildings on campus. Campus visits are currently recorded in paper journals. The information is not captured consistently, and there are no means to collect and analyze data about the visits across the entire campus.

Campus administration would like to provide the visitors with the information about the buildings on campus. The visitors will be able to view the buildings list on a website, which will be built using a Power Apps portal.

In this lab, you will provision a Power Apps portal and create a portals webpage that will show a listing of the buildings on campus.

High-level lab steps

You will follow the below outline to design the Power Apps portal:

- Provision a Power Apps portal in the Dataverse environment
- Create and configure a webpage to show a list of the buildings
- Create a new theme and apply it to the portal

Prerequisites

- Completion of **Module 0 Lab 0 - Validate lab environment**
- Completion of **Module 2 Lab 1 - Introduction to Microsoft Dataverse**

Things to consider before you begin

- Power Apps portals apps are always started from a template instead of a blank application. Your portal should have been created in Module 0 Lab 0. Once you provision a portal, it will already have pages, menus and a default theme.

Exercise #1: Create a Portal Webpage

Objective: In this exercise, you will create a new webpage that will display some static content as well as a list of buildings from Dataverse.

Task #1: Navigate to Portal

1. Navigate to <https://make.powerapps.com>.
2. Verify that you are in your Practice Environment. If you are not, change the environment at the top right.
3. Click on **Apps**
4. Locate the app that has the **Type** of **Portal**
5. Click on the app name to open the portal

You should be redirected to your portal website landing page with a welcome message. Navigate your portal to see what was created by default when you provisioned your portal.

Task #2: Create a Webpage

1. Open Power Apps portals Studio
 - Sign in to <https://make.powerapps.com> (you may still have this open in your tabs)
 - Select **Apps**
 - Locate the app that has the **Type** of **Portal**
 - Click on the ellipses (...) to the right of the portals app name and choose **Edit**

You are now in the Power Apps portals Studio. This is where you can modify and create portal content.

2. Create a new page
 - From the command bar, select **New page**
 - Select **Landing page**
3. In the properties pane, under **Display** change the **Name** from **New page (1)** to **Building Directory**
4. In the **Partial URL** change the value to **building-directory**, press the Tab key (to initiate auto-save)

The title of the page should now read **Building Directory**

Task #3: Add Static Content

1. Add a section to the webpage
 - On the canvas (area showing webpage), select a section of the page that is not a column.
 - On the toolbar (left side), select the **Components** icon
 - Choose **Two columns section** from the **Section layout** area
2. Add Static Text
 - On the canvas (area showing webpage), select the left column
 - On the toolbar (left side), select the **Components** icon
 - Choose **Text** from the **Portal components** area

- In the new text area, enter the following text: **The following is the building directory.**
 - Select the text box above the one you just edited, and click **Delete** on the command bar to remove the default text.
3. Add an Image
- On the canvas (area showing webpage), select the right column
 - On the toolbelt (left side), select the **Components** icon
 - Choose **Image** from the **Portal components** area
 - In the properties pane, click **Select an image**. Locate and select the **Pages.png**
 - In the properties pane, click the **Formatting** section drop-down and change the **Width** to 70% (be sure to type the %). You can play around with the sizing of the image until it is as desired.
4. Configure rights to display the the building list
- From the left menu, click on Settings (gear symbol) and choose **View more settings**. This will open additional settings in a new tab.
 - In the left menu, scroll down to **Security** and select **Table Permissions**.
 - Click **New** and add following values:
 - **Name:** Show Building List
 - **Table Name:** From the dropdown on the right side select Building (bc_building)
 - **Website:** Click on the Magnifier and select your website (Bellows College Visitors – [your naming])
 - **Access Type:** Global
 - **Privileges:** Read
 - From the top menu, select **Save**.
 - Scroll down to the **Web Roles** section and **Add Existing Web Role**.
 - Click on the magnifier, select **Anonymous Users** and click **Add**.
 - From the top menu, select **Save & Close**.
 - Go back to the previous tab.
5. Click **Browse website** to view the page so far. Notice that there is now the **Building Directory** option on the main menu.

You may need to configure your browser to allow pop-ups.

Task #4: Add a List Component

1. Navigate to the previous tab and continue to step #2. If not available, follow the below steps to return to this location.
 - Sign in to <https://make.powerapps.com> (you may still have this open in your tabs)
 - Locate the app that has the **Type** of **Portal**
 - Click on the ellipses (...) and choose **Edit**
 - On the toolbelt (left hand side), choose the **Pages** option

- Locate and select the **Building Directory** page you created earlier
- 2. Add a list component to the Building Directory page
 - Select the section with two columns.
 - On the toolbar (left side), select the **Components** icon
 - Choose **One column section** from the **Section layout** area (a section will appear below the image and text on the webpage)
 - Select the new column section on the canvas
 - On the toolbar (left side), select the **Components** icon
 - Choose **List** from the **Portal Components** area (a list component will appear in the new section)
- 3. Configure the list component
 - Select the list component on the canvas
 - In the properties pane (right side), enter in **Buildings List** in the **Name** field
 - In the **Table** field, choose **Building (bc_building)** from the drop-down list
 - In the **Views**, choose **Active Buildings**
 - Leave the remaining default settings
- 4. Click **Browse website** to view the page.

You should see the list of Buildings from your Dataverse database appear on the webpage.

Exercise #2: Change the Portal Theme

Objective: In this exercise, you will create a new theme that will alter the color scheme of your portal.


Task #1: Apply and Edit a Theme

1. Navigate to the previous tab and continue to step #2. If not available, follow the below steps to return to this location.
 - Sign in to <https://make.powerapps.com> (you may still have this open in your tabs)
 - Locate the app that has the **Type** of **Portal**
 - Click on the ellipses (...) and choose **Edit**
2. Apply and customize a basic theme
 - On the toolbar (left side), select the **Themes** icon
 - Ensure that the toggle for **Enable basic theme** is set to on.
 - On one of presets, click the ellipses (...) and choose **Customize**
 - A copy of the basic theme has been created.
 - On the properties pane, play around with changing the colors and exploring the impact of these changes to your portal.
 - Rename your theme
3. On the command bar, click **Sync configuration**

Create a screenshot showing the above results

PL-900T00A: Microsoft Power Platform Fundamentals - Brave
labclient.labondemand.com/LabClient/9da75733-6ca3-484b-bcfa-045f6b73236c?rc=10

The following is the building directory.



Name ↑	Created On
Alpine Ski House	4/10/2022 1:56 AM
Contoso Suites	4/10/2022 1:56 AM
Fabrikam Residences	4/10/2022 1:56 AM

PL-900T00A: Microsoft Power Platform Fundamentals
11 Hr 56 Min Remaining

InstructionsResourcesHelp

Building Directory

Name	Created On
Alpine Ski House	4/10/2022 1:56 AM
Contoso Suites	4/10/2022 1:56 AM
Fabrikam Residences	4/10/2022 1:56 AM
Microsoft Building	4/10/2022 1:56 AM
Microsoft Office	4/10/2022 1:56 AM

Challenges

- Create a different view of Buildings that just displays the Building Name. You will need to select **Browse website** from the Portal studio to see the changes.
- On the toolbar, click on the **Themes** icon and edit the CSS of your custom theme.
- Create a page with the **Form** component and modify a **List** component to add or edit

Wipram Gamurci

61% Tasks Complete

< Previous

Next >

Class & Module	IU 04: Get started with Power Automate
Name	I Wayan Wipram Sembah Klan Gamurci
IC No.	
Date & Time	Thu Apr 14 2022 10:32:09 GMT+0800 (Central Indonesia Time)

Lab: How to build an automated solution

Scenario

Bellows College is an educational organization with multiple buildings on campus. Campus visitors are currently recorded in paper journals. The information is not captured consistently, and there are no means to collect and analyze data about the visits across the entire campus.

Campus administration would like to modernize their visitor registration system where access to the buildings is controlled by security personnel and all visits are required to be pre-registered and recorded by their hosts.

Throughout this course, you will build applications and perform automation to enable the Bellows College administration and security personnel to manage and control access to the buildings on campus.

In this lab, you will create Power Automate flows to automate various parts of the campus management.

High-level lab steps

The following have been identified as requirements you must implement to complete the project:

- The unique code assigned to each visitor must be made available to them prior to their visit.
- Security personnel need to receive notifications of visitors overstaying their scheduled timeslots.

Prerequisites

- Completion of **Module 0 Lab 0 - Validate lab environment**
- Completion of **Module 2 Lab 1 - Introduction to Microsoft Dataverse**
- Campus Staff app created in **Module 3 Lab 2 – How to build a canvas app, part 2** (for testing)
- John Doe contact created with a personal email address in **Module 3 Lab 4 - How to build a model-driven app** (for testing)

Things to consider before you begin

- What is the most appropriate distribution mechanism for the visitor codes?
- How could overstay be measured and strict policies enforced?

Exercise #1: Create Visit Notification flow

Objective: In this exercise, you will create a Power Automate flow that implements the requirement. The visitor should be sent an email that includes the unique code assigned to the visit.

Task #1: Create flow

1. Open your Campus Management solution.
 - Sign in to <https://make.powerapps.com>
 - Select your **environment**.
 - Select **Solutions**.
 - Click to open your **Campus Management** solution.
2. Click **New** and select **Automation, Cloud flow** and then **Automated**. This will open the Power Automate flow editor in a new window.
3. In **Choose your flow's trigger**, search for **Microsoft Dataverse**.
4. Select the trigger **When a row is added, modified or deleted**, and then click **Create**.
 - Select **Added** for **Change type**
 - Select **Visits** for **Table name**
 - Select **Organization** for **Scope**
 - On the trigger step, click the ellipsis (...) and click **Rename**. Rename this trigger "**When a visit is added**". This is a good practice, so you and other flow editors can understand the purpose of the step without having to dive into the details.
5. Select **New Step**. This step is required to retrieve visitors information, including email address.
6. Search for **Microsoft Dataverse**.
7. Select **Get a row by ID** action.
 - Select **Contacts** as **Table name**
 - In the **Row ID** field, select **Visitor (Value)** from the Dynamic content list.
 - On this action, click the ellipsis (...) and click **Rename**. Rename this action "**Get the Visitor**". This is a good practice, so you and other flow editors can understand the purpose of the step without having to dive into the details.
8. Click **New Step**. This is the step that will create and send email to the visitor.
9. Search for *mail*, select **Office 365 Outlook** connector and **Send an email (V2)** action.
 - If asked to Accept terms and conditions for using this action, click **Accept**.
 - Select **To** field, select **Email** from the Dynamic content list. Notice that it is beneath the **Get the Visitor** header. This means you are selecting the Email that is related to the Visitor that you looked up in the previous step.
 - Enter **Your scheduled visit to Bellows College** in the **Subject** field.
 - Enter the following text in **Email Body**:

Dynamic content needs to be placed where fields are named in brackets. It is recommended to copy & paste all text first and then add dynamic content in the correct places.

Dear {First Name},

You are currently scheduled to visit Bellows Campus from {Scheduled Start} until {Scheduled End}.


Your security code is {Code}, please do not share it. You will be required to produce this code during your visit.

Best regards,

Campus Administration
Bellows College

10. Select the **Untitled** flow name at the top and rename it to **Visit** notification
11. Press **Save**

Leave this flow tab open for the next task. Your flow should look approximately like the following:

 When a visit is created ? ...

* Change type

Create ▼

* Table name

Visits ▼

* Scope

Organization ▼

Show advanced options ▼



 Get the Visitor ? ...

* Table name


Contacts ▼

* Row ID


 Visitor (Value) ×

Show advanced options ▼



 Visit notification ? ...










* To


 Email × ;



* Subject


Your scheduled visit to Bellows College

* Body

Font ▼ 12 ▼ **B** *I* U         

Dear  First Name × ,

You are currently scheduled to visit Bellows Campus from  Scheduled Start × until  Scheduled End × .

Your security code is  Code × , please do not share it. You will be required to produce this code during your visit.

Best regards,

Campus Administration
Bellows College

Show advanced options ▼

Task #2: Validate and test the flow

1. Open a new tab in your browser and navigate to <https://make.powerapps.com>
2. Click **Apps** and select the **Campus Staff** app you created
3. Leaving this tab open, navigate back to the previous tab with your flow.
4. On the command bar, click **Test**. Select **Manually** and then **Save & Test**.
5. Leaving the flow tab open, navigate back to the previous tab with the **Campus Staff** app.
6. Press **+** to add a new Visit record
7. Enter **John Doe** as **Name** and choose any **Building**
8. Choose **John Doe** as the **Visitor**
9. Choose the **Scheduled Start** and **Scheduled End Dates** to any dates in the future.
10. Press the **Checkmark** icon to save the new visit
11. Navigate back to the previous tab with the flow being tested. Watch as the flow is run. If there are any errors, go back and compare your flow to the example above. If the email is sent successfully, you will receive it in your inbox.
12. Click the back arrow on the command bar
13. In the **Details** section, notice that the **Status** is set to **On**. This means your flow will run whenever a new Visit is created, until you turn it off. Any time the flow runs, you will see it added to the **28-day run history** list.
14. Turn the flow off by clicking **Turn off** on the command bar. You may need to press the ellipses (...) to see this option.
15. Close this window.

Exercise #2: Create Security Sweep flow

Objective: In this exercise, you will create a Power Automate flow that implements the requirement. A security sweep needs to be performed every 15 minutes, and security should be notified if any of the visitors overstayed their scheduled time.

Task #1: Create flow to retrieve records

1. Open your Campus Management solution.
 - Sign in to <https://make.powerapps.com>
 - Select your **Environment**.
 - Select **Solutions**.
 - Click to open your **Campus Management** solution.
2. Click **New** and select **Automation, Cloud flow** and then **Scheduled**. This will open the Power Automate flow editor in a new window.
3. Set the flow to repeat every **15** minutes.
4. Click **Create**.
5. Click **New step**. Search for *Current* and select **Microsoft Dataverse** connector. Select **List rows** action.
 - Enter **Visits** as **Table name**
 - Click **Show advanced options**
 - Enter the following expression as **Filter rows**


```
statecode eq 0 and bc_actualstart ne null and bc_actualend eq null and  
Microsoft.Dynamics.CRM.OlderThanXMinutes(PropertyName='bc_scheduledend',Pr  
opertyValue=15)
```


- To break it down:
 - **statecode eq 0** filters active visits (where Status equal Active)
 - **bc_actualstart ne null** restricts search to visits where Actual Start has a value, i.e. there was a checkin
 - **bc_actualend eq null** restricts search to visits where there was no check out (Actual End has no value)
 - **Microsoft.Dynamics.CRM.OlderThanXMinutes(PropertyName='bc_scheduledend',PropertyValue=15)** restricts visits where visits meant to complete more than 15 minutes ago.
- On this action, click the ellipsis (...) and click **Rename**. Rename this action "**List active visits that ended more than 15 minutes ago**". This is a good practice, so you and other flow editors can understand the purpose of the step without having to dive into the details.
- 6. Click **New step**. Search for *Apply*, select **Apply to each** action
- 7. Select **value** from dynamics content in the **Select an output from previous steps** field. Notice that it is beneath the **List active visits that ended more than 15 minutes ago** gray header. This means you are selecting the list of visits that you looked up in the previous step.
- 8. Retrieve Building data for related record
 - Click **Add an action** inside the Apply to Each loop.
 - Select **Microsoft Dataverse**.
 - Select **Get a row by ID** action.
 - Select **Buildings** as **Table name**
 - Select **Building (Value)** as **Row ID** from the Dynamic content
 - Click ... beside **Get a record**, select **Rename**. Enter **Get building** as step name
- 9. Retrieve Visitor data for related record
 - Click **Add an action** inside the Apply to Each loop.
 - Select **Microsoft Dataverse**.
 - Select **Get a row by ID** action.
 - Select **Contacts** as **Table name**
 - Select **Visitor (Value)** as **Row ID** from the Dynamic content
 - Click ... beside **Get a record**, select **Rename**. Enter **Get visitor** as step name
- 10. Send email notification
 - Click **Add an action** inside the Apply to Each loop. Add **Send an email (V2)** action from **Office 365 Outlook** connection.
- 11. Enter your email address as **To**
- 12. Enter the following in the **Subject** field. **Full Name** is a dynamic content from the **Get visitor** step.


```
{Full Name} overstayed their welcome
```
- 13. Enter the following in the **Body** field. **Name** is a dynamic content from **Get building** step. You may need to scroll to the bottom of the list.

14. There is an overstay in building {Name}.
15.
16. Best,
17.
Campus Security

18. Select flow name **Untitled** in the upper left corner and rename it to **Security Sweep**
19. Press **Save**

Your flow should look approximately like the following:


 Recurrence ...

* Interval
15

* Frequency
Minute ▼

Show advanced options ▼



 List active visits that ended more than 15 minutes ago ? ...

* Table name
Visits ▼

Select columns
Enter a comma-separated list of column unique names to limit which columns :

Filter rows
statecode eq 0 and bc_actualstart ne null and bc_actualend eq null and Microsoft.Dynamics.CRM.OlderThanXMinutes(PropertyName='bc_scheduledend',PropertyValue=15)

Sort By
Enter an Odata style orderBy query to sort the rows

Expand Query
Enter an Odata style expand query to list related rows

Fetch Xml Query
Enter a Fetch XML query for advanced customization


Row count
Enter the number of rows to be listed (default = all)

Skip token
Enter the skip token obtained from a previous run to list rows from the next page


Partition ID
An option to specify the partitionId while retrieving data for NoSQL tables

Hide advanced options ^



 Apply to each ...


* Select an output from previous steps
value x

 Get building ? ...

* Table name
Buildings ▼







* Row ID
Building (Value) x

Show advanced options ▼


 Send an email notification (V3) ? ...

* To
chsimms@microsoft.com

* Subject
Full Name x overstayed their welcome

* Body
Font 12 **B** *I* U      
There is an overstay in building Name x
Best.
Campus Security

Show advanced options ▼

 Add an action

Task #2: Validate and test the flow

Your flow will begin sending you emails (to the email you specified when creating the John Doe contact previously) if there are visits that meet the requirements laid out in the flow.

1. Validate that you have visit records that:
 - a. Have active status
 - b. Scheduled End is in the past (by more than 15 minutes)
 - c. Actual Start has a value.

Note: To view this data, navigate to make.powerapps.com in a new tab. Click Solutions on the left pane to locate your solution. Select the Visit entity, then select the Data tab. Click Active Visits in the top right-hand corner to display the view selector, then select All fields.

1. Navigate to your **Security Sweep** flow, if not already there.
2. When your flow opens, click **Test**.
3. Select **Manually**.
4. Click **Save & Test** and **Run Flow**.
5. When flow completes, click **Done**.
6. Expand **Apply to each**, then expand the **Send an email notification** step. Check the **Subject**, **Email Body** values.
7. Select the back arrow to the Security Sweep flow details. Select **Turn off** on the command bar. This is to prevent flow from executing on a schedule on the test system.

Create a screenshot showing the above results

