The Microsoft Power Up final challenge : Briefing document

Contents

The Power Up Challenge

The intention of the Power Up challenge is to enable you to confirm to the program that you have learned the basics of working with Power Platform products. It is a celebration of all that you have discovered on your journey.

Whilst the challenge you'll be undertaking represents the end of your time with the Power Up program we hope that this will be just the beginning of your time building Power Platform solutions to meet all manner of business needs.

So we invite you to take on this challenge with great positivity in your heart. This is an opportunity for you to demonstrate to the program and yourself that you are ready to progress further with the platform and we wish you to succeed just as much as you do.

Project Brief

Contoso High School has been facing increasing challenges with parking availability, leading to complaints from both teachers and visitors. To address these concerns, the leadership team seeks a better understanding of parking usage and occupancy.

To achieve this, they have requested the development of a simple system that enables staff and visitors to request parking on a daily basis. Additionally, the system should facilitate the inspection of parking spaces by a member of staff to verify help understand who is using the parking. This will be used to help improve parking management, enhance accessibility and reduce unauthorized use.

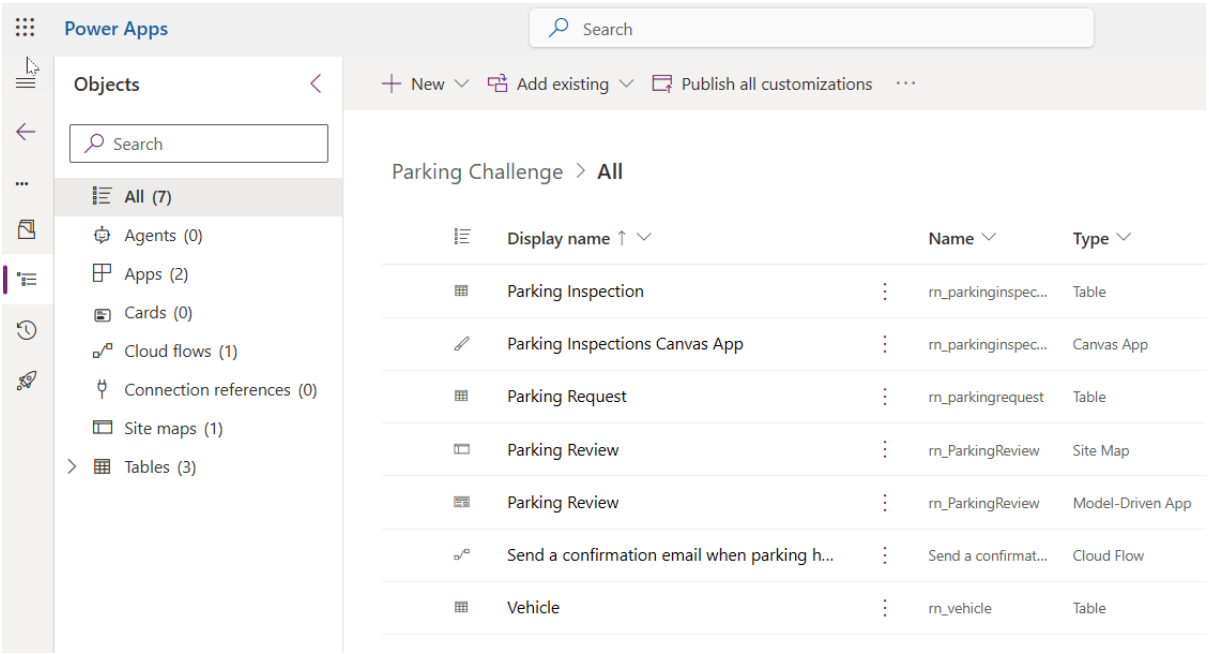
High level requirements

Using the resources with which you have been supplied you need to build the following

- A Power Platform solution called “Parking Challenge” with :
  - A dedicated publisher (see Appendix 4 for details)
  - A data model comprising 3 tables
    - Vehicle
    - Parking Request
    - Parking Inspection
  - A model-driven app to enter visitor and staff parking requests on a daily basis
  - A canvas app to be used by a parking inspector (a staff member) to log parking inspections
  - A Power Automate cloud flow to confirm parking requests
- A Power BI report is required to support the organization in managing their parking more effectively.

Additionally, you will need to explain how you would extend and improve the solution that you have built.

From a Power Platform solution perspective your solution will contain the following objects in addition to the creation of Power BI report.



Your submission will be in the form of a single narrated video explaining how you have met the requirements of the challenge in addition to discussing the ways in which you might choose to extend the solution.

Detailed requirements will be set out in the remainder of this document.

Assumptions and notes

Please note the following assumptions as far as this scenario is concerned.

- There is just 1 inspection of all parked vehicles each day at 5pm. During this inspection every vehicle parked at that time is logged.
- The inspection is performed by a member of staff using a canvas app on a tablet device
- Parking requests made after the inspection takes place are not considered valid
- We will be using specific datetime columns for the inspections and the requests to simplify the data import process
- There are only **15 car parking spaces** available at any one time (whilst this is artificially low it ensures that we are working with manageable amounts of data)
- Whilst we are considering parking in the context a school you may wish to consider the scenario in a business context, especially when thinking about how you might extend the solution

Challenge Resources

The resources you have been provided with include :-

- ParkingDataModel.xlsx to give you a strong indication of how to build the data model successfully in Dataverse
- 3 csv files to be used to populate the data tables
- A range of images that you may wish to use when building your apps, mainly to make them a bit more fun. You are most welcome to source your own image to put your own stamp on the products you create.

Dataverse requirements

You need to create 3 tables to support the rest of the solution

The 3 tables are

- Vehicles
- (Visitor and Staff) Parking requests
- Parking inspections

Details around the required table structure can be found in the **ParkingDataModel.xlsx** spreadsheet included in your challenge resources.

Forms

Forms are required for all 3 tables

Vehicles

- Include all columns as follows
    - Vehicle Name
    - Make
    - Model
    - Vehicle Image
    - VehicleOwnerEmail
- Create 2 subgrids in the **vehicles** form linked to **Parking Requests** and **Parking Inspections** tables.

Parking Requests

- Include all custom columns as follows
    - ParkingRequestName
    - Vehicle
    - ParkingRequestDateTime

Parking Inspections

- Include all custom columns as follows
    - ParkingInspectionName
    - InspectionDateTime
    - Vehicle
    - Parking Request

Views

Views are required for all 3 tables.

Vehicles

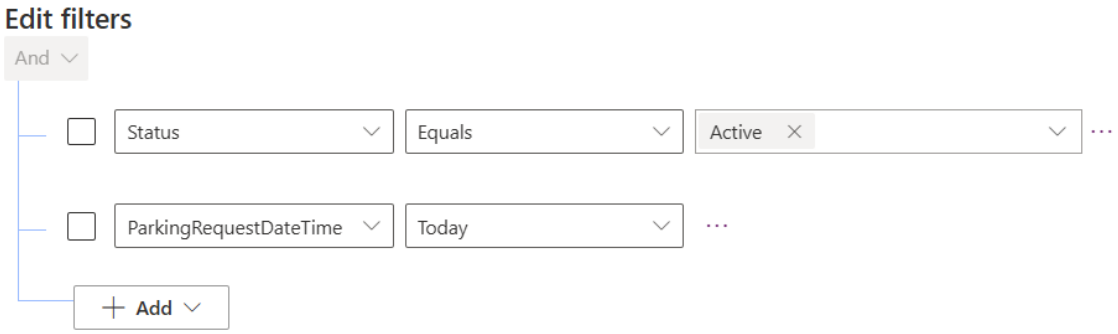
- Include the following custom columns in the **Active Vehicles** view
    - Vehiclename
    - Make
    - Model
    - VehicleOwnerEmail

Parking Requests

- Include the following columns in the **Active Parking Requests** view
    - ParkingRequestName

- ParkingRequestDateTime
- Vehicle

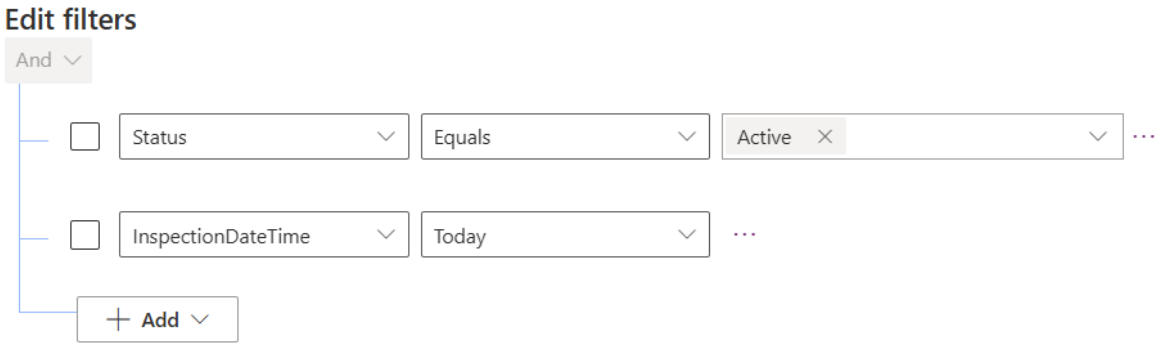
- Add an additional view called **Active Parking Requests Today**
  - Include the same columns as before (tip: save the view and then "save as" to create the new view)
  - Ensure that a filter is added so that it only shows records where the **ParkingRequestDateTime** column is Today



Parking Inspections

- Include the following columns in the **Active Parking Inspections** view
  - ParkingInspectionName
  - Vehicle
  - InspectionDateTime
  - ParkingRequest
  - ParkingRequestDateTime (note that this is a column from a related table)

- Add an additional view called **Active Parking Inspections Today**
  - Include the same columns as above
  - Ensure that the InspectionDateTime is filtered to Today

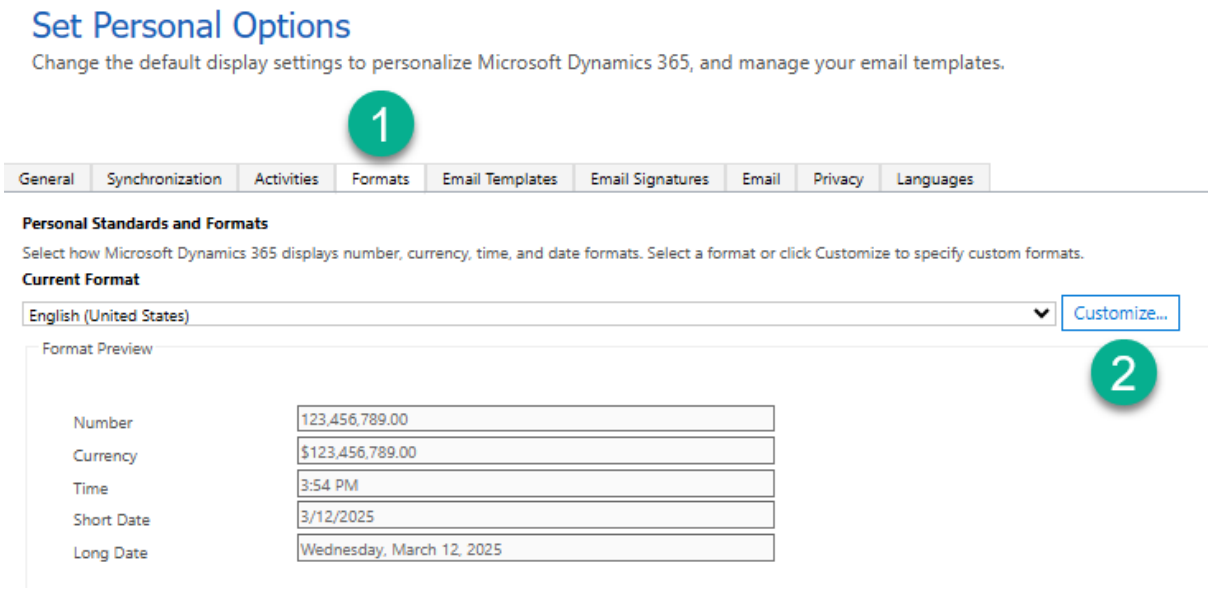


You will need to configure the forms and views for all tables and these should include all the custom columns you have created.

You will also need to ensure that you use the csv files provided in order to populate the 3 tables with data.

Tips:

- Use the **ParkingDataModel.xlsx** to identify the tables and columns required
- You may wish to review the data in the uploads csv file to understand the sort of data you might expect to find in the tables
- All columns created should be done **without spaces** in their names. This will simplify the upload process.
- Once you have created your data model, compare this to the **ParkingDataModel.xlsx** spreadsheet
- If you do need to delete a column and are prevented from doing so, please ensure that it is not included in any forms or views as you will need to delete them here before deleting the column from the table.
- When creating the tables the following order will prove the simplest – Vehicle, Parking Request, Parking Inspection
- There is no need to add the **VehicleImage** into the view for the vehicle table in your Dataverse view.
- When importing the data you it will need to be in the following order – Vehicle, Parking Request, Parking Inspection
- The csv files are in a **US datetime format**. To import the data from the model driven app you will need to ensure that your regional settings in your model-driven app are set to US English. To do this go to the settings cog when playing the app. Select personalization settings and then formats and select US English. Power Up environments by default have US format settings, however you may have changed these at some point during the course



Model-driven app requirements

Create a model-driven app with 3 tables to ensure an admin could review parking and create parking requests on behalf of staff and visitors and also review Parking Inspections.

Include the following tables

- Vehicles
- Parking Requests
- Parking Inspections

Canvas app requirements

A tablet style canvas app is needed to enable a member of staff to go out into the parking area and log the parking. This will give you plenty of time room to place the controls within your app.

The app will have 4 screens in total

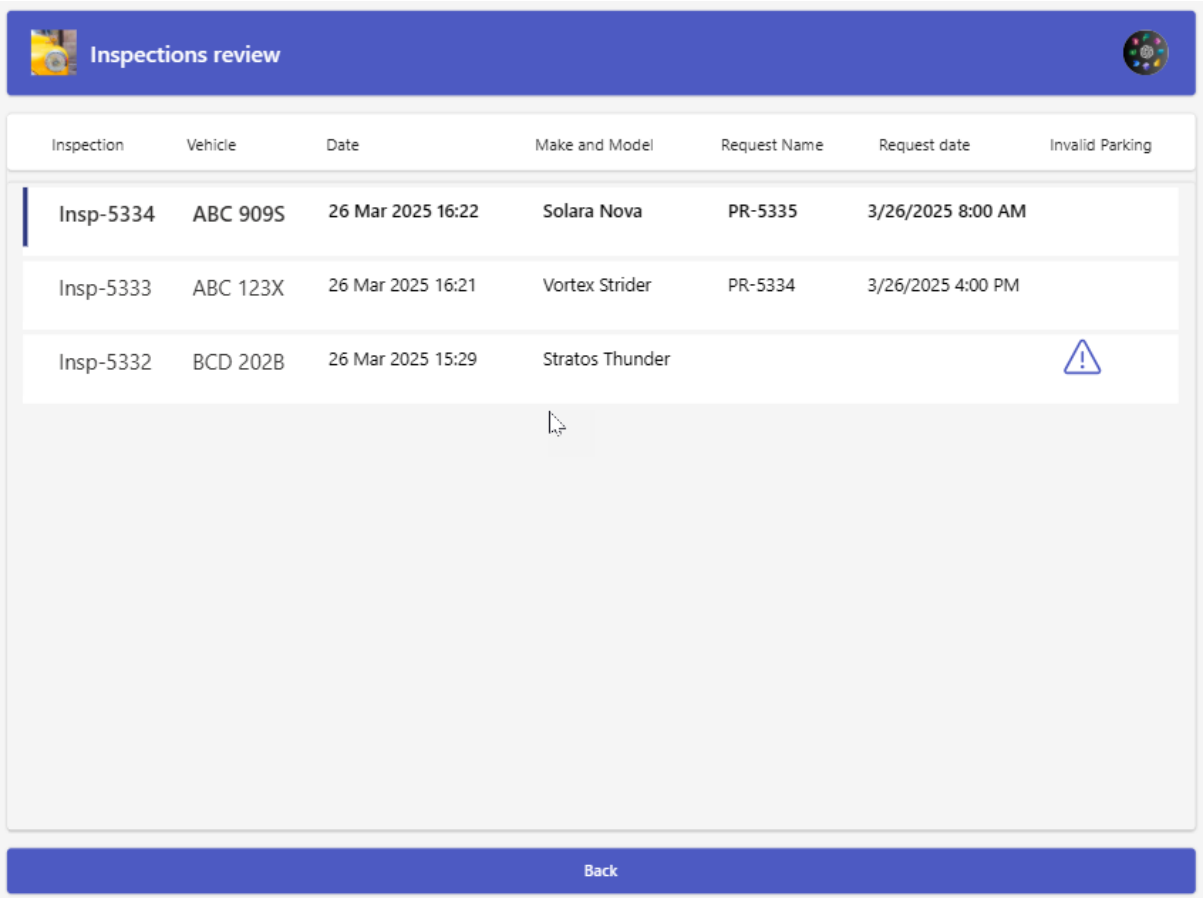
Home Screen

- 2 buttons enabling navigation to the **Review Screen** and the **New Inspection Screen**
- The button for the **Parking Inspection screen** will also ensure that the form is ready for use (hint: use the NewForm() function)

Review screen

- A gallery showing the parking inspections **Today** only.
- Columns required are as follows

- Inspection
- Vehicle
- InspectionDateTime
- Make
- Model (note that this has been combined with Make gallery below, but it is not a requirement)
- Parking Request (e.g. PR-5335)
- ParkingRequestDateTime
  - Tip ThisItem.ParkingRequest.ParkingRequestDateTime will reach into the ParkingRequest table and return the information
  - Note that this may well be blank if the Inspection has not been related to a request.
- Below is an example of a gallery for the parking inspections for the day, but please come up with your own designs. You do not need to include an icon to highlight instances of invalid parking.



In order to have a filtered view of **inspections today** the following formula would work

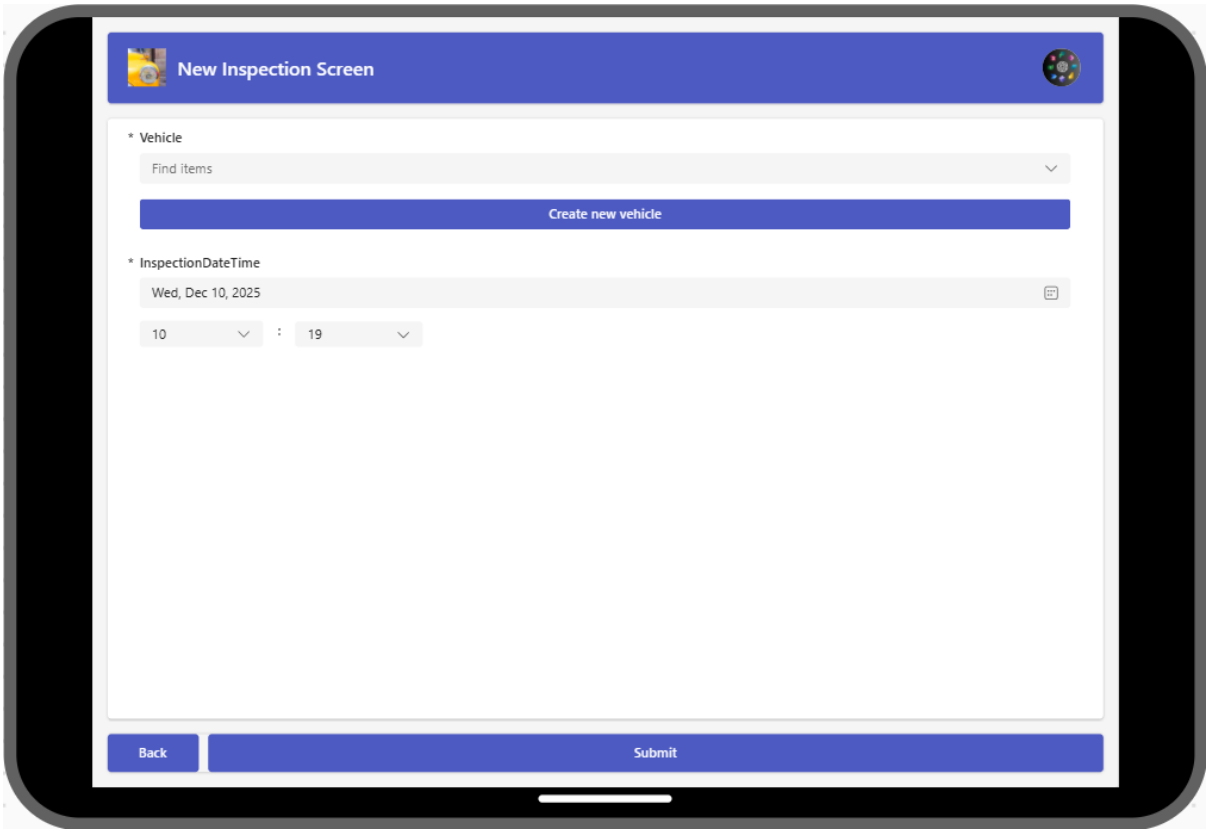
```
Sort(Filter('Parking Inspections','Parking Inspections (Views)'.'Active Parking Inspections Today'), InspectionDateTime,SortOrder.Descending)
```

For this to be effective you would have to have set up a view showing inspections today called “Active Parking Inspections Today” when building the model.

New inspection screen

- Contains a form to allow a new inspection to be created
    - Allow for the selection of a vehicle
    - The default inspection date is **today**
    - The default inspection hour is **now**.
        - Note the following formula is required in the defaultselecteditems property
          [If(FormMode.New,Text(Hour(Now()),"00"),Parent.Default)]
          (replace commas with ; if necessary)
    - The default inspection minute is **now**
        - Note the following formula is required in the defaultselecteditems property
          [If(FormMode.New, Text(Minute(Now()),"00"),Parent.Default)]
          (replace commas with ; if necessary)
- A button to submit the form
- Contains a navigation through to a **New Vehicle screen**

Below is a screenshot of what the **New Inspection Screen** might look like. Feel free to come up with your own designs.



Related Parking requests

Note that there is **NO REQUIREMENT** to add information regarding the **parking request** that relates to the Parking Inspection. This is primarily on the grounds of the complexity of the exercise as there is a level of precision required for it to function correctly within an app. However, Appendix 2 sets out the steps you would need to go through in order for your app to reflect this.

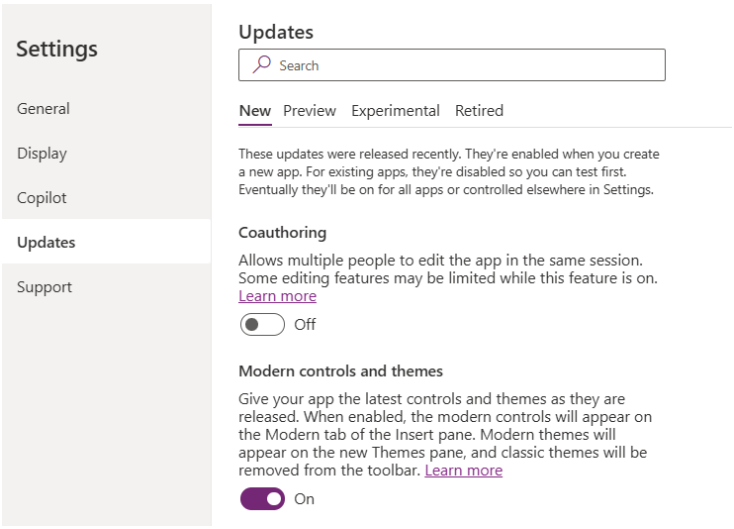
New Vehicle screen

- Contains a form to allow a new vehicle to be created
    - Note that this exists in particular to enable us to identify vehicles never seen before in the car park
    - Columns required are VehicleName, Make, Model and you may choose VehicleImage if you wish.
- A button to submit the form
- When the form is submitted return to the **Home screen** ensure that the parking inspections table is refreshed
- Navigation back to the **New Inspection screen**

Additional considerations

Use **MODERN controls** throughout your app where relevant. In order for these to be available you need to go to settings -> updates and ensure the slider is set to **ON**.

This will simplify the review process and ensure that you can be properly supported in the challenge.



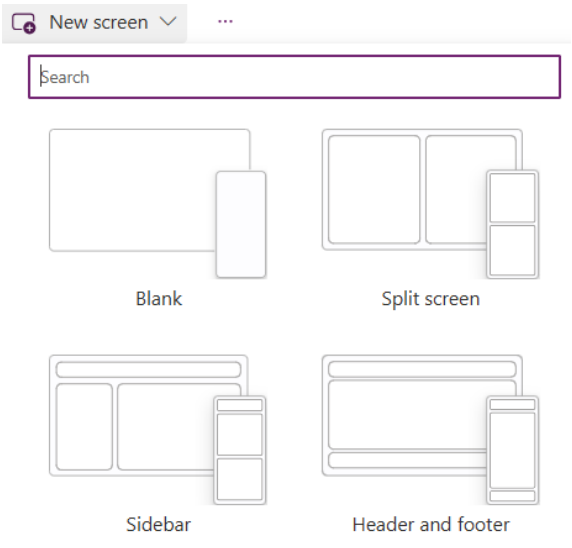
The app needs to work, inspections need to be created on the current date and vehicles must be capable of being created. There is no requirement to relate vehicle inspections to parking requests, but you may wish to try this using the instructions in appendix 2.

There need to be headers on all pages.

The navigation experience needs to be sensible and feel natural.

Tips

You may wish to play around with some of the out of the box screens available to you when you select new screen. They can be a little trickier to deal with, but they make the app pleasant to look at.



You may wish to make use of some of the vehicle images supplied within your data model and present them in your app.

Automation requirements

There is a requirement to send a confirmation email back to the owner of the vehicle to advise that their request to use the parking has been granted.

Flow name : Parking Request Notification

Email Subject : Parking Request Confirmation **<vehicle name to be inserted here>**

The text of the email is as follows:

> Please be advised that your parking request for vehicle **<vehicle name to be inserted here>** today has been granted.
>
> Request id : **<parking request name to be inserted here>**
> Many thanks,
>
> Contoso School Administration

Please add in the signature of the school into the email. (Note that when sending to gmail accounts the image will not be rendered.)

Tips

- If you are sending emails you would need to do so to a valid email address rather than the ones supplied in the vehicles dataset.
- You will want to turn **off** the flow if you upload multiple records into the parking requests table at a later date to avoid many emails from being triggered.
- There is no need to add an image of the vehicle into the email. Whilst this is technically possible it is not a requirement.

Power BI requirements

You will need to create a Power BI report that enables a user to understand the extent to which people are parking in the car park without the permission of the organization.

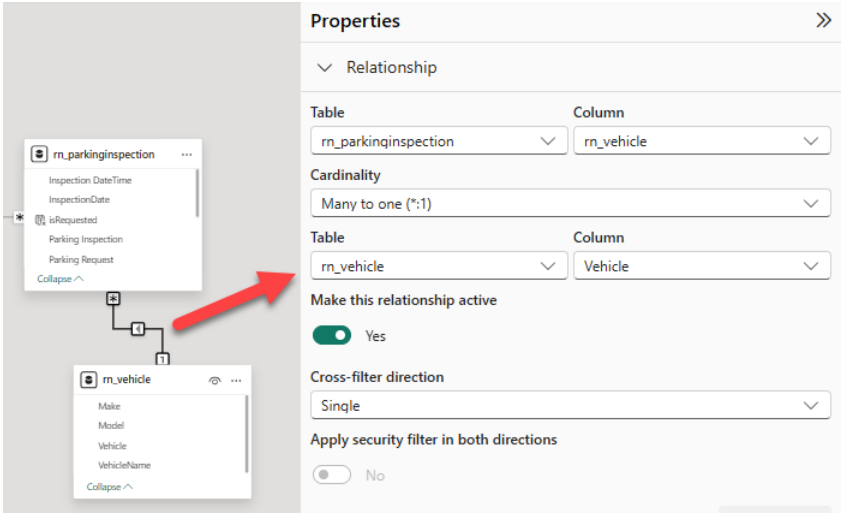
Data model

Parking Inspections table

Note you will need to add in calculations to enable you to understand if a vehicle has permission to park (tip: a valid parking inspection is one that is joined to the parking request table).

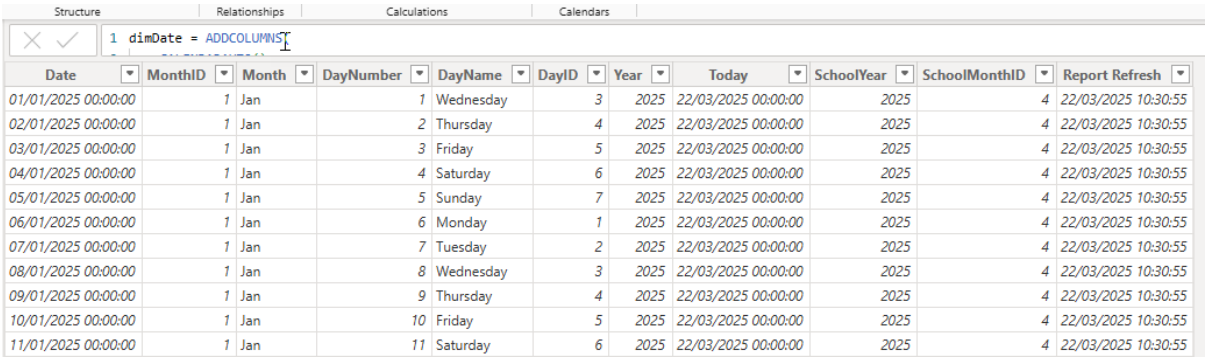
Vehicles table

Note that this is joined to the Parking Inspection table using the record unique identifier (a 36 character code). This is likely to be one of the more challenging parts of the exercise, but it is worthwhile as it comes up frequently when working with Dataverse and we make use of this in our filters.



Calculated Date Dimension table

This will look similar to the one shown below.



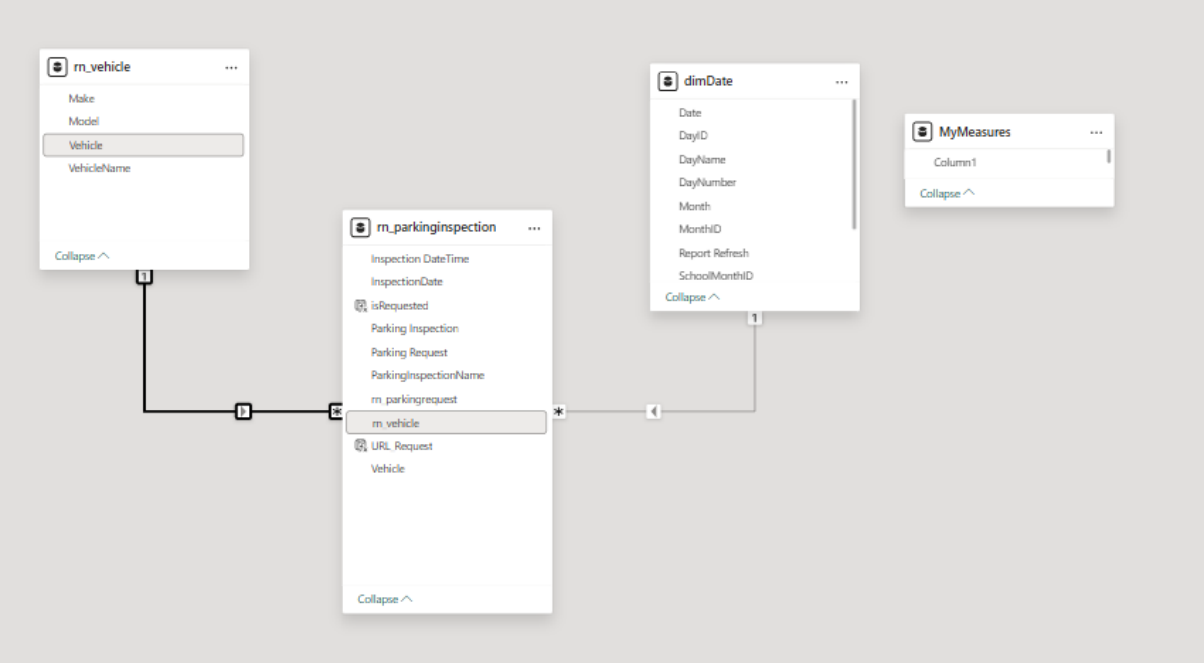
Date	MonthID	Month	DayNumber	DayName	DayID	Year	Today	SchoolYear	SchoolMonthID	Report Refresh
01/01/2025 00:00:00	1	Jan	1	Wednesday	3	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
02/01/2025 00:00:00	1	Jan	2	Thursday	4	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
03/01/2025 00:00:00	1	Jan	3	Friday	5	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
04/01/2025 00:00:00	1	Jan	4	Saturday	6	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
05/01/2025 00:00:00	1	Jan	5	Sunday	7	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
06/01/2025 00:00:00	1	Jan	6	Monday	1	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
07/01/2025 00:00:00	1	Jan	7	Tuesday	2	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
08/01/2025 00:00:00	1	Jan	8	Wednesday	3	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
09/01/2025 00:00:00	1	Jan	9	Thursday	4	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
10/01/2025 00:00:00	1	Jan	10	Friday	5	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55
11/01/2025 00:00:00	1	Jan	11	Saturday	6	2025	22/03/2025 00:00:00	2025	4	22/03/2025 10:30:55

The table can be created using the DAX code provided (See Appendix 1)

Note: You will need to ensure that the Month column (which contains the name of the month) is sorted by the MonthId or SchoolMonthID column.

MyMeasures table

To house any measures you create



Measures

Total Inspections – A count of the total inspections

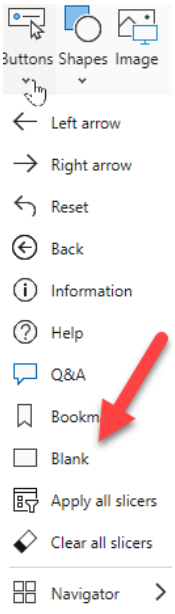
Total Requests – A count of the total inspections with valid requests

% Inspections with valid parking requests – Total Requests divided by total inspections

Data visualizations

3 Report pages in total. **Home Page**, **Filters** and **Parking Review**

Home Page – A simple set of navigation buttons to the filters page and parking review page. (Tip : Use blank buttons and configure them by altering the colour, title and action properties as covered within the course)



Filters – Contains filters on Vehicle Make and Model, Calendar filter, Day of Week filter. These must be set up to filter the Parking review page.

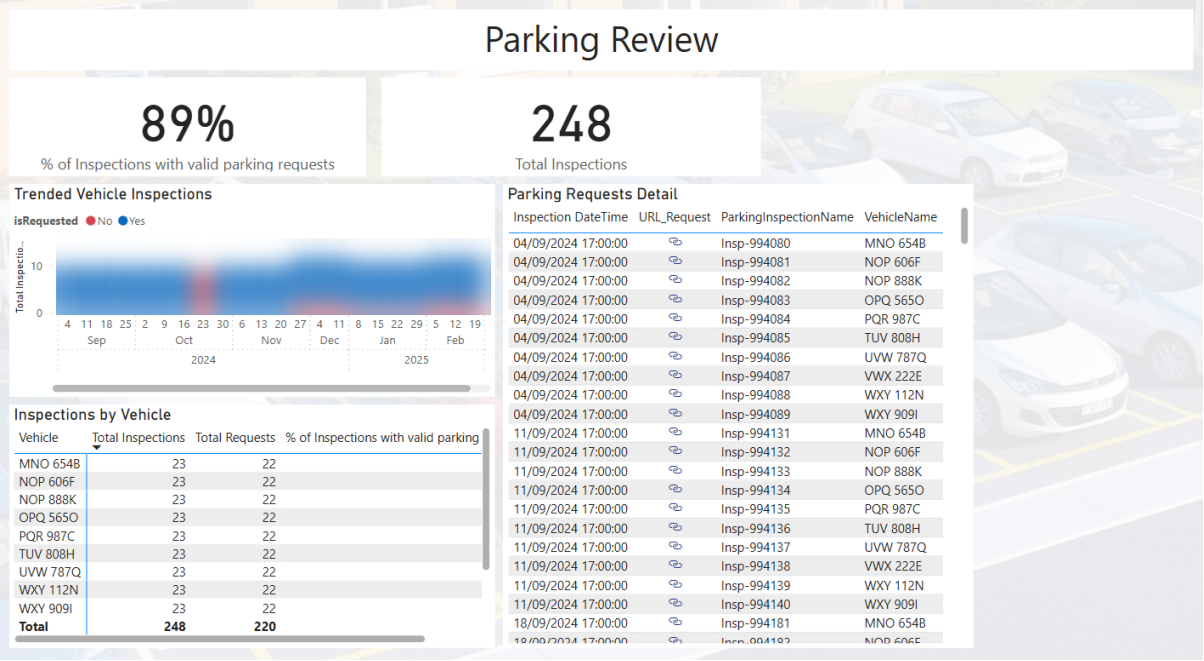
Parking review – Cards showing total inspections and Requests as a % of Inspections

Bar chart indicating parking inspections over time split between those where permission was granted and those where it was not

Matrix indicating total inspections by vehicles, total parking requests and the measure "% of Inspections with valid parking requests"

Detail table including a link back to the inspection record in your model driven app (note that you'll need a calculated column for this)

The screen might look similar to the one shown below, but you are welcome to lay it out as you see fit.



Notes

We are only looking at parking from the perspective of Inspections and not from that of Parking Requests. In fact for this exercise we do not need to bring in the Parking Requests table as the information we require is contained in the Parking Inspections table already.

It is possible that you only have relationships to between the requests and inspections for the data that you have imported. This is perfectly fine for the purposes of this exercise and it may be simpler to filter out the records created via the app building process.

Extending the solution

Whilst this comes at the end of the challenge this is arguably the most important part of it, as it is your chance to consider what YOU would do differently, based on the challenge and your knowledge of Power Platform.

It is rare for a Power Platform project to stay still or even be perfect at the outset. Now that you have familiarised yourself with the challenge you need to consider how you would change it in order to meet your own needs.

There is **no need to build out the changes** you would make and you need to come up with at least **one change** that you would make to the solution you have created – you don't need to hold yourself to the school scenario, it could be parking in the context of any organisation big or small.

Submission deliverables Checklist

- ☐ Solution created with new publisher with a prefix of the learner initials (See Appendix 4)
- ☐ 3 Tables created with forms and views configured for custom columns
    - Vehicles
    - Parking Requests
    - Parking Inspections
- ☐ Model-driven app created with 3 tables
- ☐ Canvas app with 4 screens created
    - Uses Modern controls
    - Home screen with 2 buttons navigating to the Review screen and the New Inspection screen
    - Review Screen showing inspections Today
    - New Inspection screen – form allowing the creation of an inspection of a vehicle
    - New Vehicle screen – form allowing the creation of a new vehicle
- ☐ Cloud flow created
    - Flow that sends an email to a vehicle owner when a Parking Request is made
    - Email needs to be in the format as described
- ☐ Power BI report
    - 3 Screens
        - Home Screen with buttons to navigate to the 2 other screens
        - Filters Screen – Filters on Vehicle Make and Model, Calendar filter, Day of Week filter

            Filters must be set up to work across pages (so that they affect the Parking Review screen)

        - Parking Review Screen
            - Cards, Matrix, Tables, Bar chart as requested
- At least 1 element of the original solution considered and proposals for changes explained.

Tips

- Have a clear idea in your mind as to the things you would like to say in advance. You don't need a script, but you may find notes helpful. Just be yourself.
- You may wish to set up all the browser tabs you need in advance.
- If you are able to within your recording software pause the recording, think about what you're going to say and then say it as this can reduce the number of retakes.

Submit these via the Power Up challenge portal within the deadlines set for your cohort

Appendix 1 – Date Dimension

Below is the DAX, as used in the Power BI module to create a date dimension

```
dimDate = ADDCOLUMNS(
    CALENDARAUTO(),
    "MonthID", MONTH([Date]),
    "Month", FORMAT([Date], "MMM"),
    "DayNumber", DAY([Date]),
    "DayName", FORMAT([Date], "dddd"),
    "DayID", WEEKDAY([Date], 2),
    "Year", YEAR([Date]),
    "Today", TODAY(),
    // Add a SchoolYear column that adjusts the year based on the month
    "SchoolYear", IF(
        MONTH([Date]) >= 9,
        YEAR([Date])+1,
        YEAR([Date])
    ),
    "SchoolMonthID", if (MONTH([Date]) >= 9,
        Month([Date])-9,
        Month([Date])+3
),
"Report Refresh",Now() //We'll use this to display the report refresh date
)
```

Note that if this does work then please refer to the text file included in the resources for the overall course.

Appendix 2 – Canvas Apps – Identifying Parking Requests

It is logical to consider that when creating the Inspections canvas app to want to automatically identify those **Parking Requests** that related to a **Parking Inspection**. This way an inspector would be able to understand if someone parking had indeed made a parking request that day.

This has been left out of the requirements on the basis that solving this requirement has some complexities.

Broadly speaking to solve this a number of actions would need to be taken.

- Introduce the **ParkingRequest** column into the form
- Update the vehicle combobox name to *cmbVehicle* – it will most likely be called DatacardValueX
- Bring the ParkingRequest column into the ParkingInspection form
    - Update the **items** property

```
Filter(
    'Parking Requests',
    'Parking Requests (Views)'.'Active Parking Requests
Today',
    Vehicle.Vehicle = cmbVehicle.Selected.Vehicle
)
```

    - Update the **defaultselecteditems** property (so that it is automatically selected)

```
If(
  FormMode.New,
  Table(
    First(
      Filter(
        'Parking Requests',
        'Parking Requests (Views)'.'Active Parking Requests Today',
        Vehicle.Vehicle = cmbVehicle.Selected.Vehicle
      )
    )
  ),
  [ Parent.Default ]
)
```

    - Update the combobox name to cmbParkingRequest

- At this point you might also wish to confirm to yourself that you have correctly identified the parking request. To do this I would suggest adding a 2 text columns and into the text properties add
  - `CmbParkingRequest.Selected.Vehicle.VehicleName`
  - `Text(cmbParkingRequest.Selected.ParkingRequestDateTime," dd mmm yyyy hh:mm")`

Once these actions have been performed the app will look closer to the following

Appendix 3 – Glossary

Vehicle – A vehicle that has at some point parked in the car park whether permitted or otherwise.

Vehicle Name – A column within the **Vehicle** table. The government recognised reference for the vehicle which might be known as the registration number, license plate number, tag number of the vehicle

ParkingRequest – A request made by a staff member or member of the public to park on a given day. A single request is valid for the entire day and these are entered by reception staff

ParkingInspection – A record indicating that a vehicle has been noted as having parked in the car park at a specific time on a specific day. These are created by a member of staff of the organization.
Inspections only take place once per day at 5.00pm, primarily to simplify the data model required and reporting building process.

Appendix 4 – Solution publisher

The unique publisher for your solution should be named as "**Your Name** Publisher" and uses **your initials** (e.g. rn) as a prefix for your objects as shown below