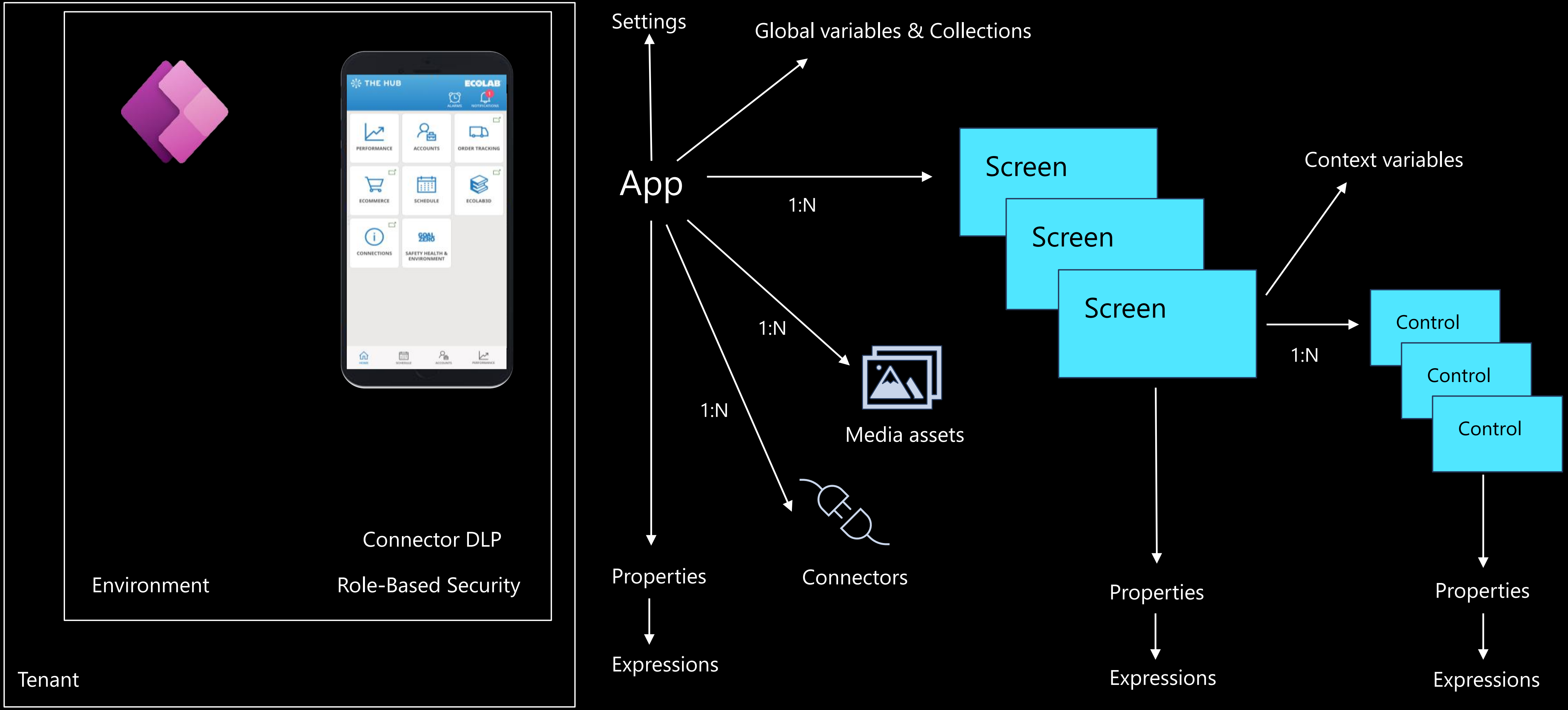
Microsoft
Power Platform

Canvas Apps Performance Best Practices

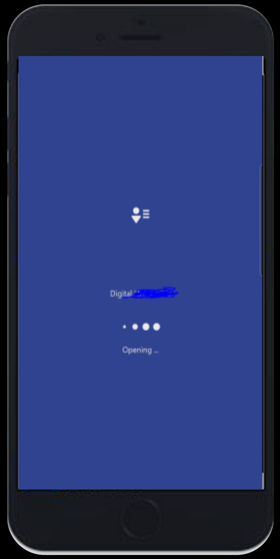
Content

1. Canvas App design/Architecture
2. Common Performance issues & Best practices
3. Tools to monitor/troubleshoot app performance

Power Apps Canvas App



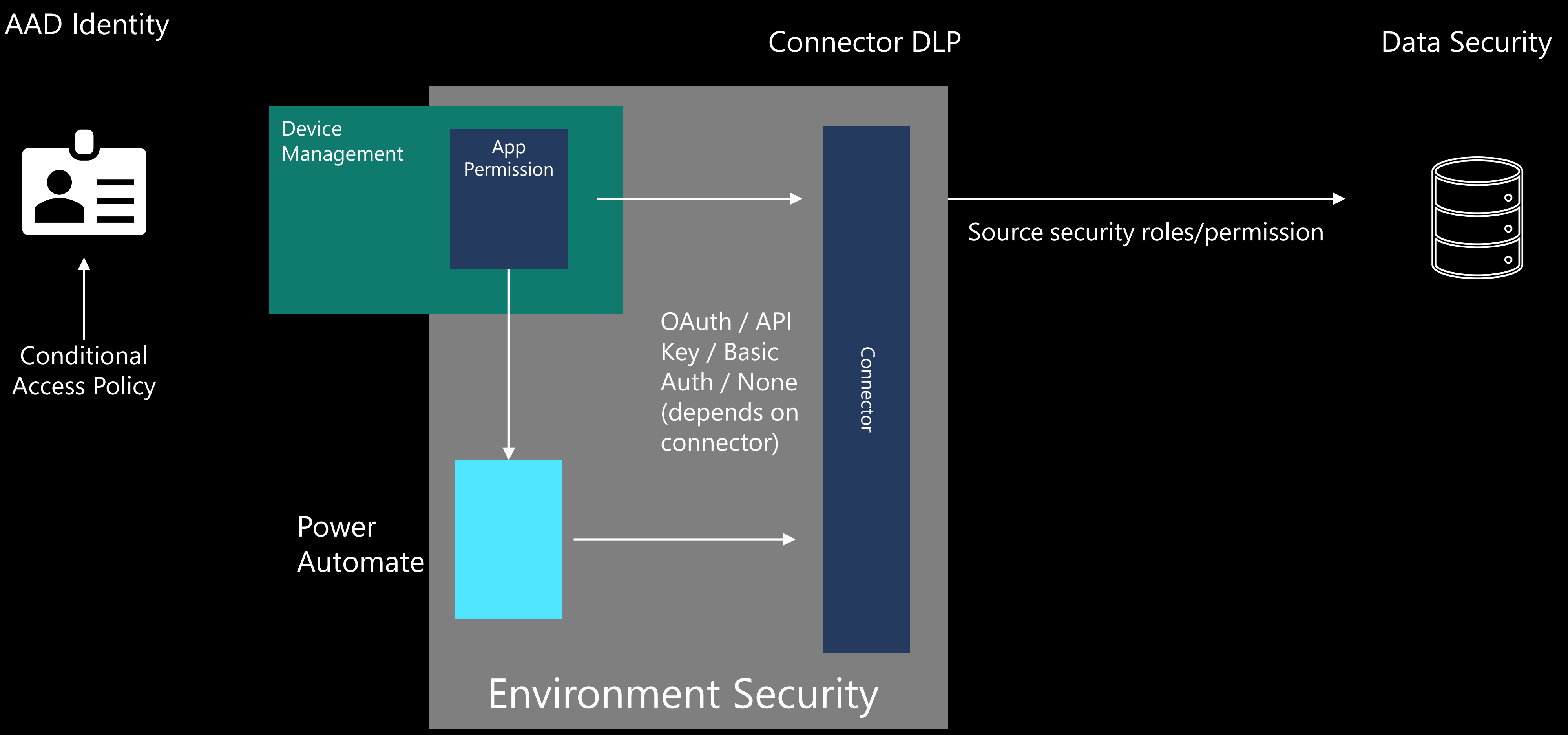
Run-time Execution Phases



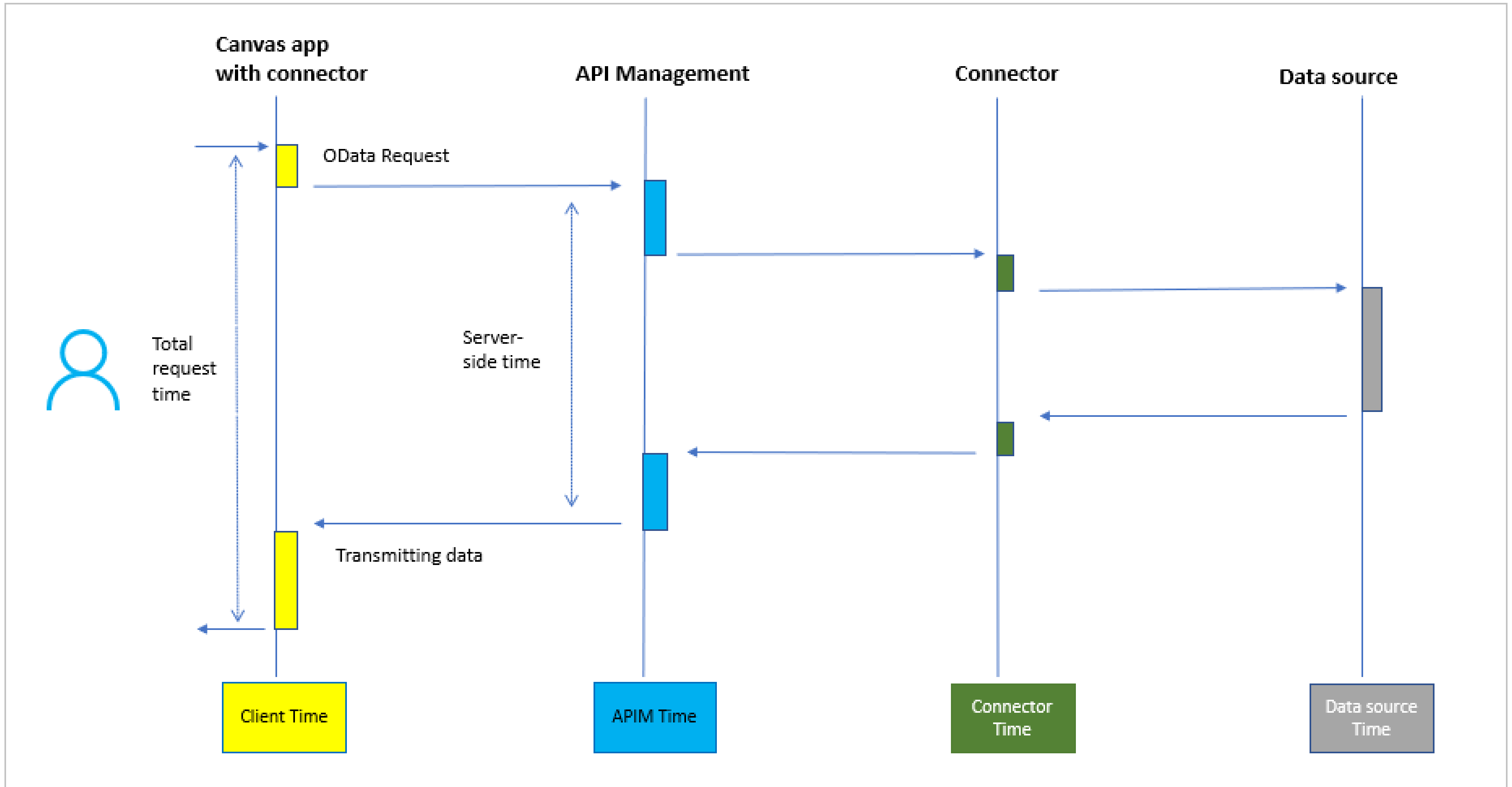
1. Authenticate the user – prompts user to sign in with credentials for connections the app needs.
2. Get metadata – retrieves metadata, such as version of the Power Apps platform
3. Initialize the app – performs any tasks specified in the **OnStart** property
4. Render the screens – renders the first screen with controls

[Understand canvas app execution phases and data call flow - Power Apps | Microsoft Docs]

Control gates

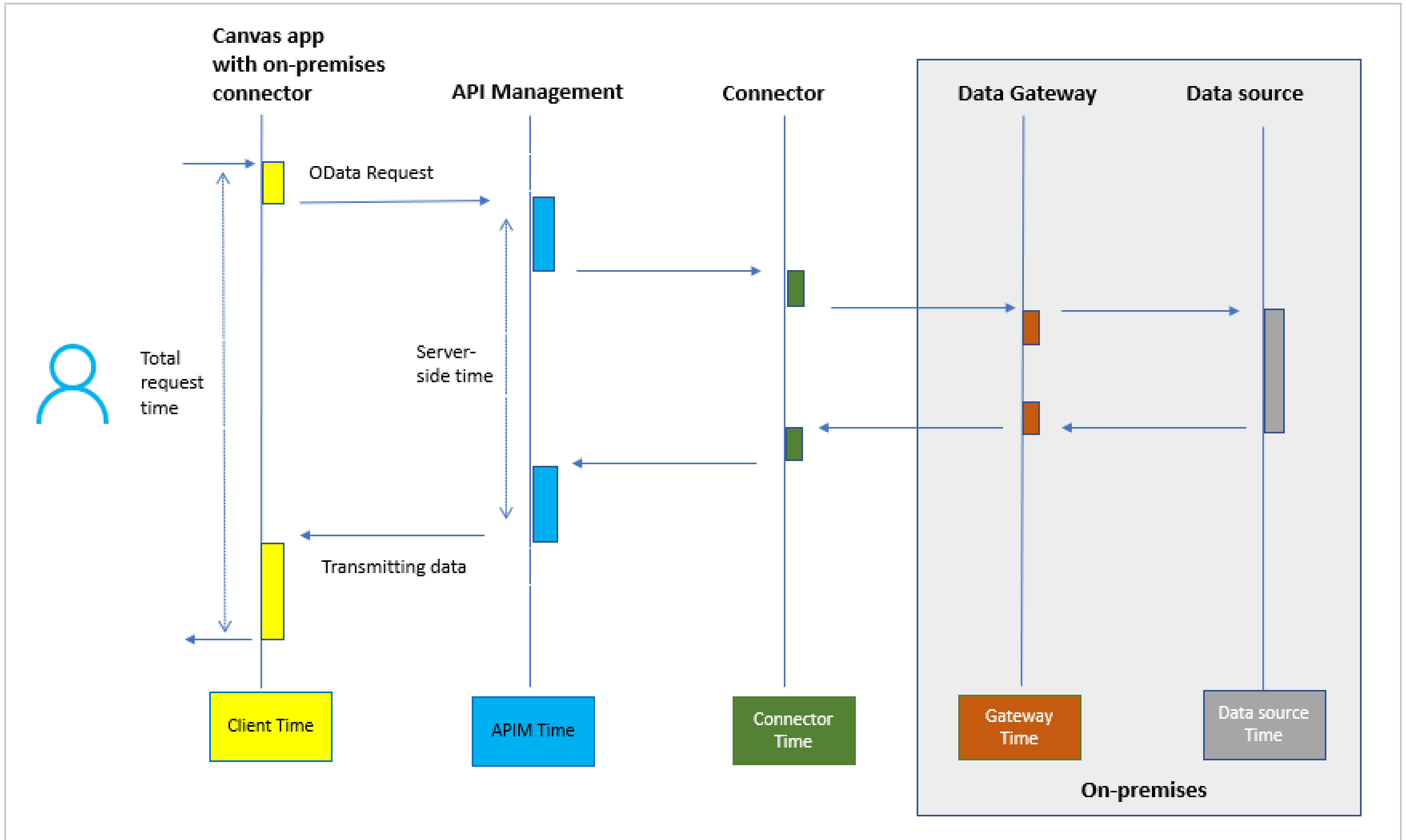

AAD Identity
Conditional Access Policy
Device Management
App Permission
Power Automate
OAuth / API Key / Basic Auth / None (depends on connector)
Connector
Connector DLP
Environment Security
Source security roles/permission
Data Security

Data call flow with online data sources



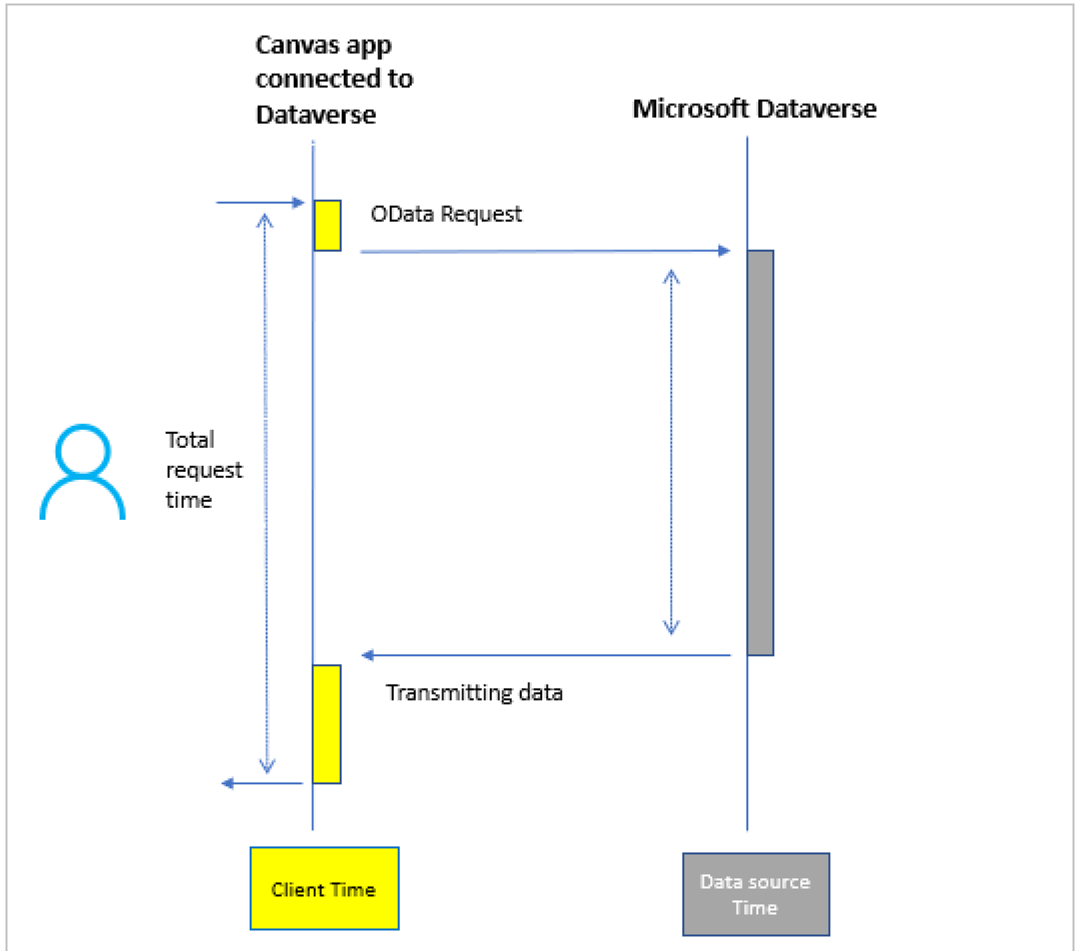
$$UX\ timetaken = \sum_{i=1}^{n}(Layer\ Time) + Transmission\ Time$$

Data call flow with on premise data sources



$$UX\ timetaken = \sum_{i=1}^{n}(Layer\ Time) + Transmission\ Time$$

Data call flow with Dataverse “Native”



$$UX\ timetaken = \sum_{i=1}^{n}(Layer\ Time) + Transmission\ Time$$

Common performance issues & best practices

1. Geographical latency

- Issue: User is too far away from the geo location of the Environment. On Premise Data Gateway is too far away from the underlying source.

- Best practice: Publish the app in an Environment nearest to the users. On Premise Data Gateway should be as close as possible to the underlying source.

2. Avoid client heavy app

- Issue: App retrieves large amount of data and performs heavy data operations at the client-side such as JOIN, Sort, AddColumns, GroupBy, etc.

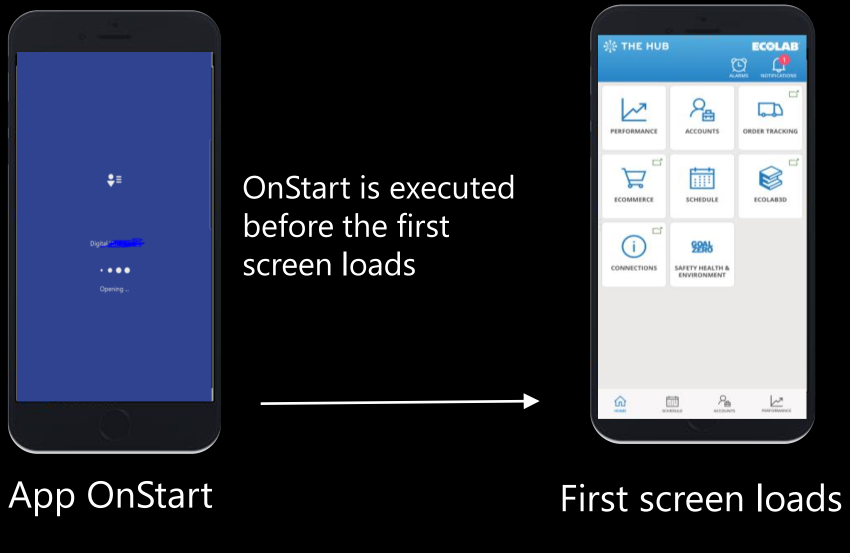
- Best practice: Perform data shaping/operations in the back-end where possible. For Dataverse sources, use **Views** and calculated columns.

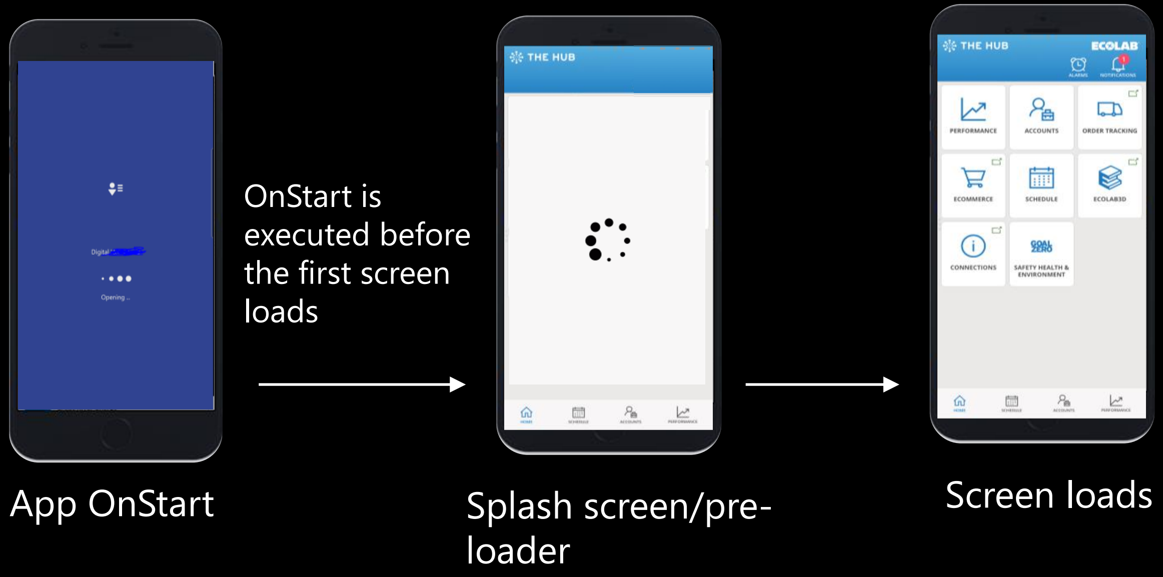
Common performance issues & best practices

3. Avoid long running OnStart event

- Issue: OnStart contains long running executions – eg gathering and loading data from various sources.

- Best practice: Move long running initialization to a "splash screen" / pre-loader screen. Distribute initialization across multiple screens where possible. Use "Concurrent" to parallelize execution.

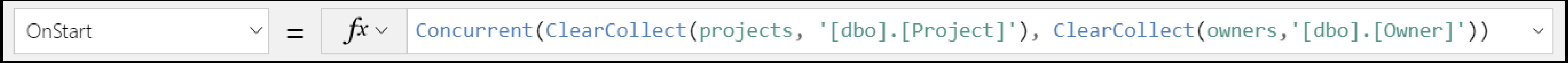




Common performance issues & best practices

4. Use Concurrent calls where possible

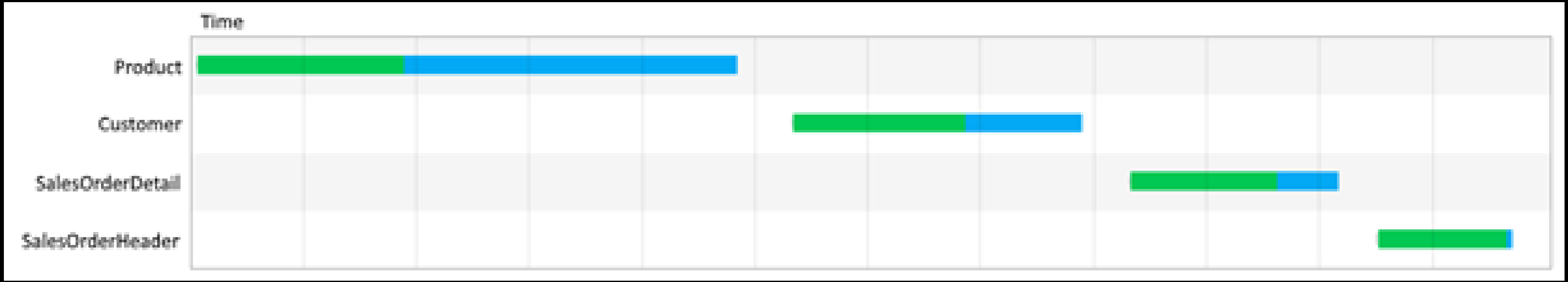
- Issue: App collects data from connectors sequentially from multiple sources with no inter-dependencies.
- Best practice: Use "Concurrent" function to run parallel executions.



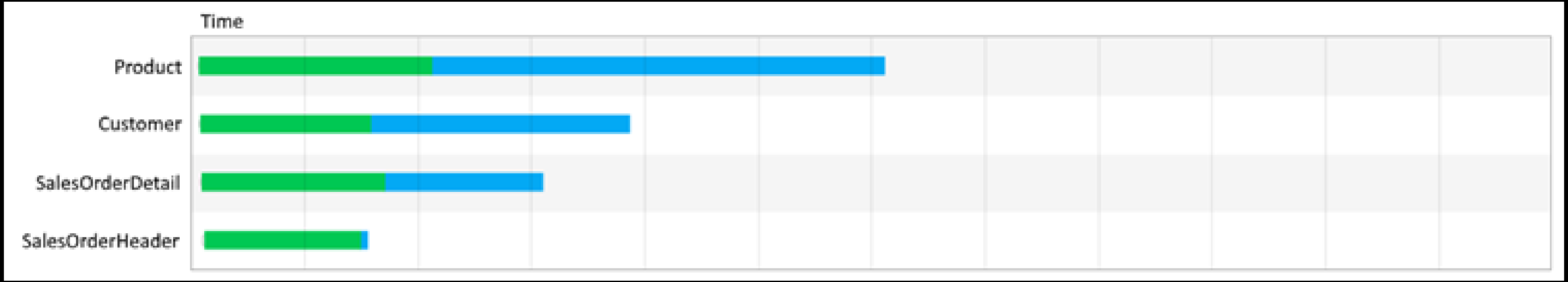
OnStart = *fx*

```
Concurrent(ClearCollect(projects, '[dbo].[Project]'), ClearCollect(owners,'[dbo].[Owner]'))
```

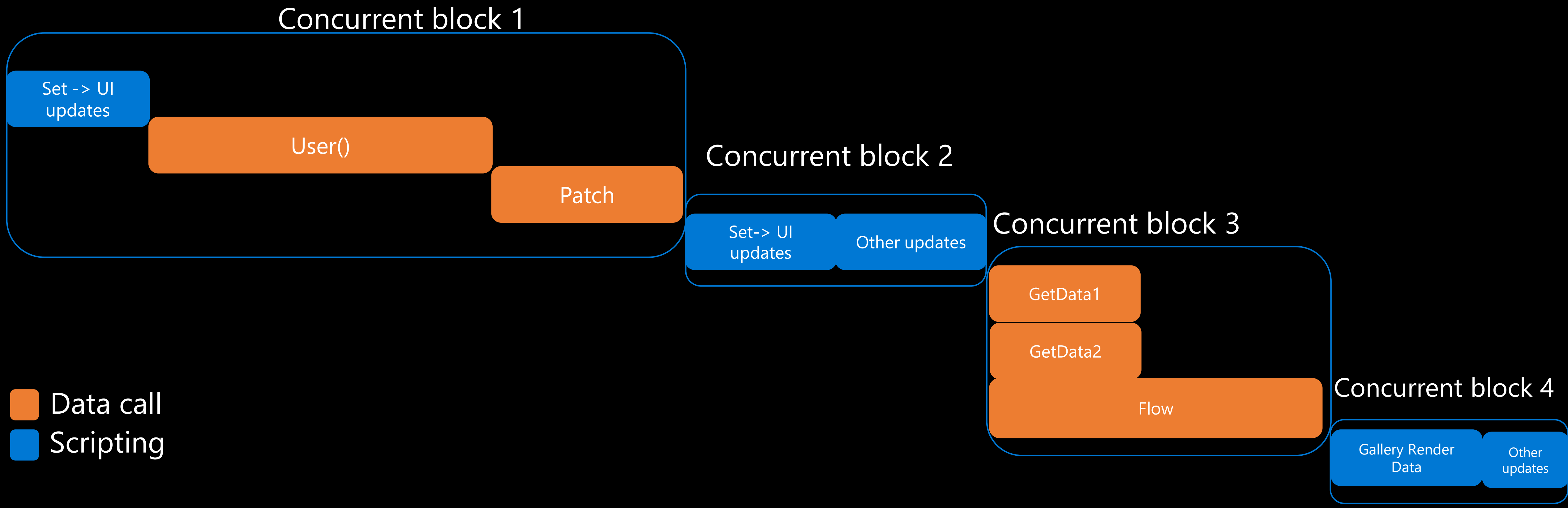
Sequential calls:



Concurrent calls:



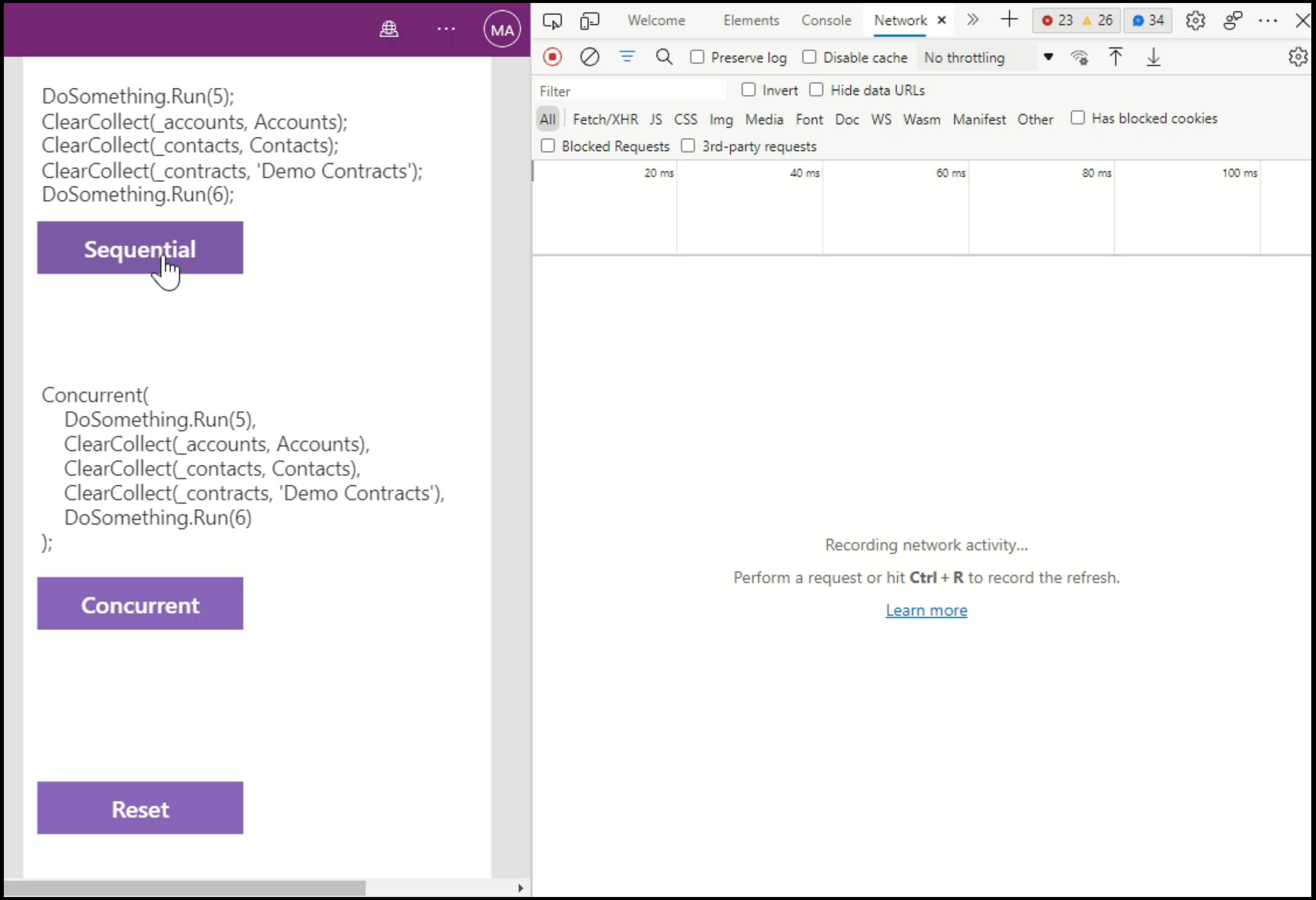
Concurrent blocks



🟧 Data call
🟦 Scripting

Common performance issues & best practices

4. Use Concurrent calls where possible



```
DoSomething.Run(5);
ClearCollect(_accounts, Accounts);
ClearCollect(_contacts, Contacts);
ClearCollect(_contracts, 'Demo Contracts');
DoSomething.Run(6);
```

```
Concurrent(
   DoSomething.Run(5),
   ClearCollect(_accounts, Accounts),
   ClearCollect(_contacts, Contacts),
   ClearCollect(_contracts, 'Demo Contracts'),
   DoSomething.Run(6)
);
```

Recording network activity...

Perform a request or hit **Ctrl** + **R** to record the refresh.

Common performance issues & best practices

5. Cache frequently used data

- Issue: App is too chatty and retrieves frequently used data many times.
- Best practice: Cache frequently used data – use variables for in-memory cache or save to local storage to avoid network trips.

Example: App retrieves current user's department name in multiple places in the app using the Office365 Users connector. Instead of invoking the connector in multiple places, cache the value using a global variable.

```
Set(_userDepartment, Office365Users.MyProfileV2().department)
```

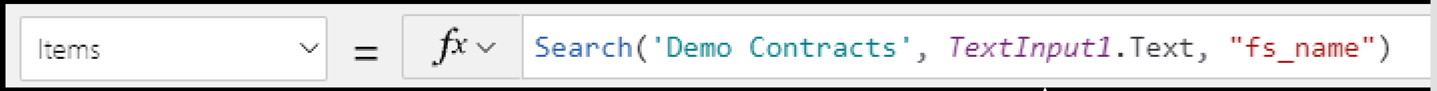
In subsequent places in the app, use the variable _userDepartment instead of using the connector.

Common performance issues & best practices

6. Set DelayOutput to true for input controls with dependent formula

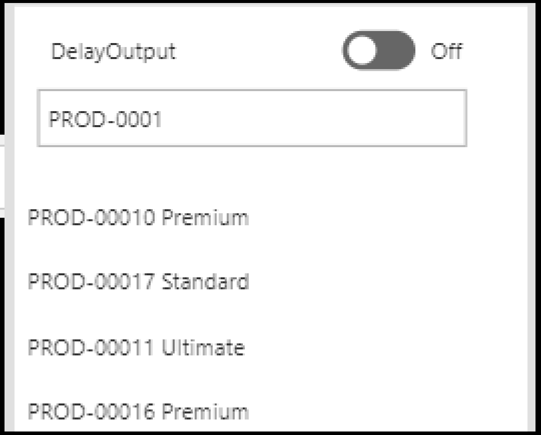
- Issue: App is too chatty when there are connections that depends on TextInput controls. The formula is evaluated each time a change is detected in the text input.
- Best practice: Set "DelayOutput" to **true** to delay execution of formula until the input has completed in the input control.

Example: When implementing a search experience with a text input.



```
Items = fx Search('Demo Contracts', TextInput1.Text, "fs_name")
```

Gallery has formula dependent on TextInput1



DelayOutput Off

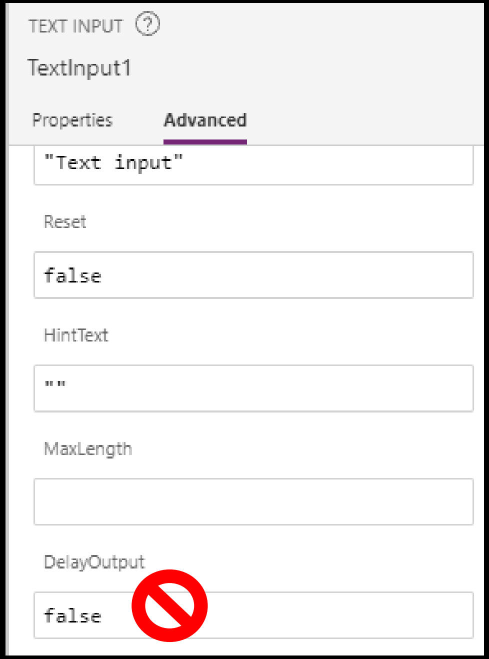
PROD-0001

PROD-00010 Premium

PROD-00017 Standard

PROD-00011 Ultimate

PROD-00016 Premium



"Text input"

Reset: false

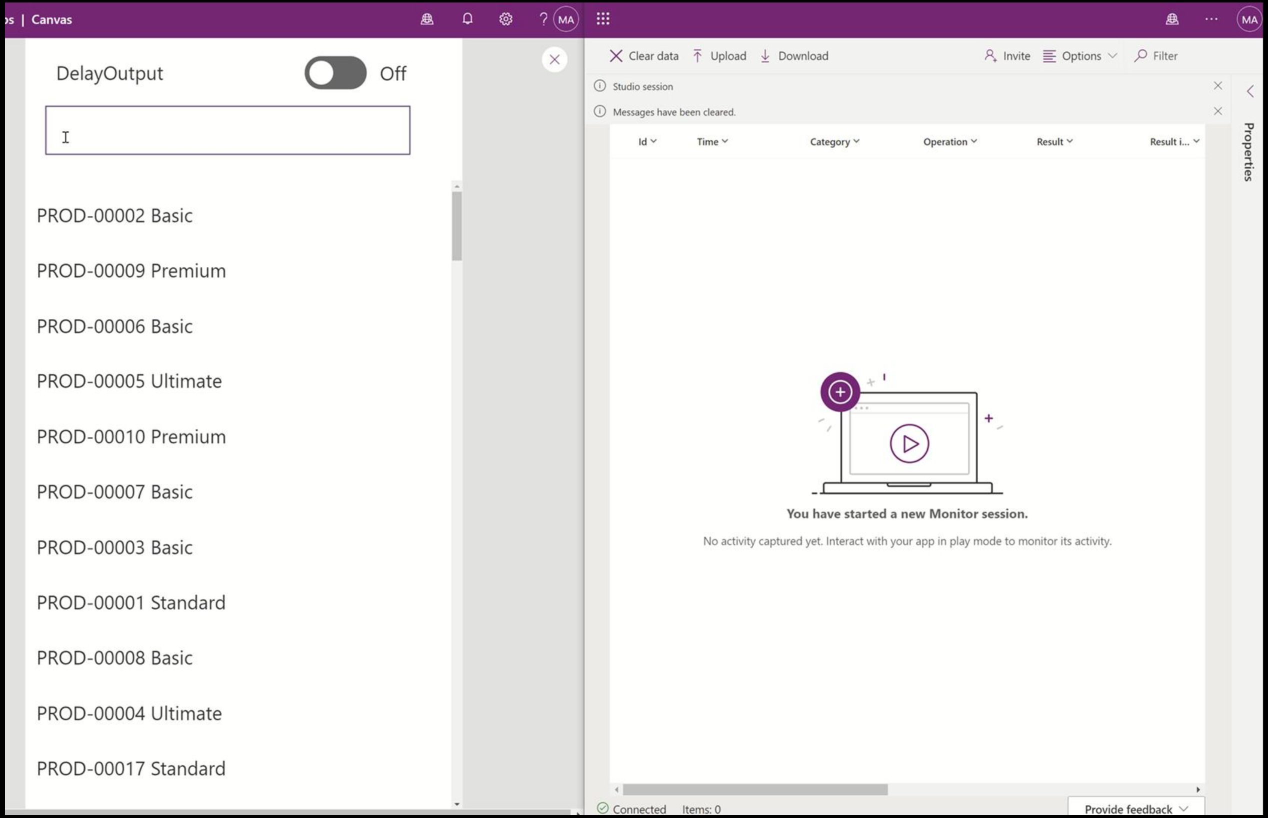
HintText: ""

MaxLength:

DelayOutput: false

Common performance issues & best practices

6. Set DelayOutput to true for input controls with dependent formula



DelayOutput Off

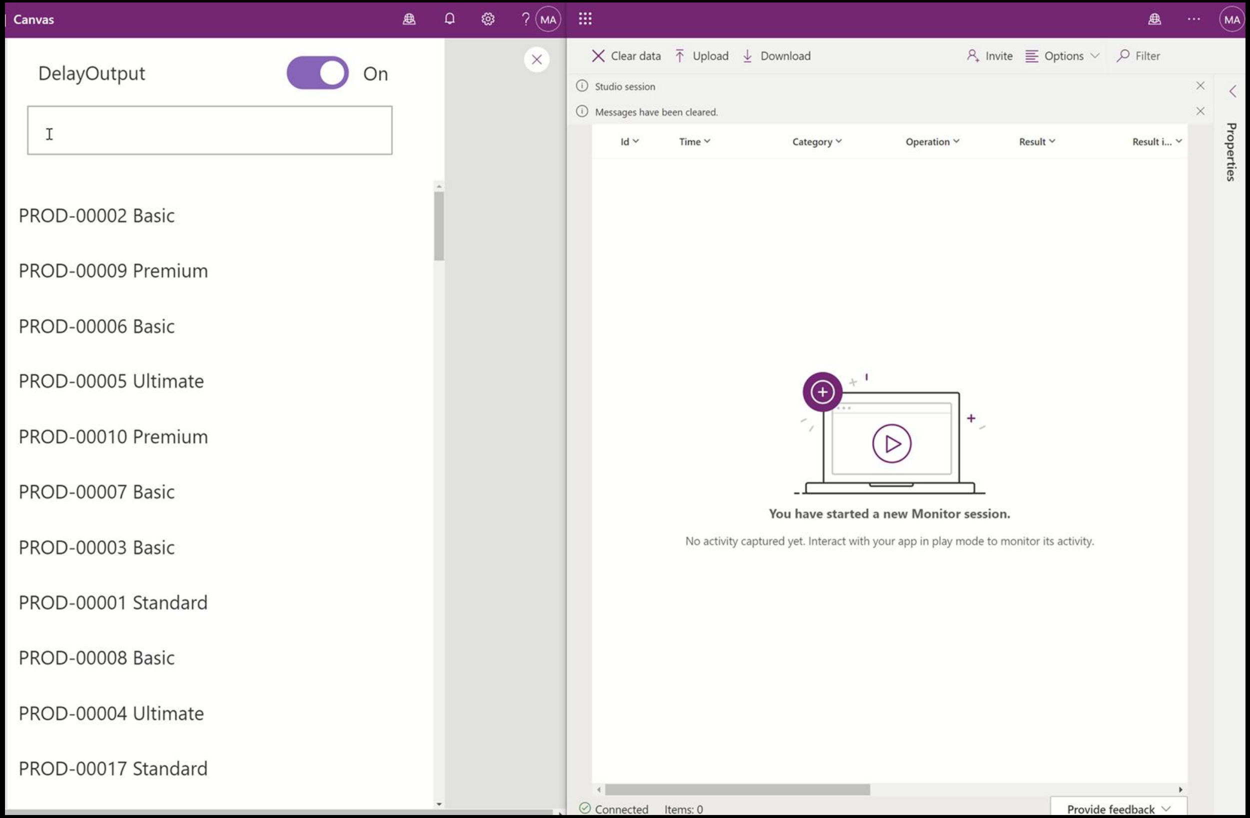
PROD-00002 Basic
PROD-00009 Premium
PROD-00006 Basic
PROD-00005 Ultimate
PROD-00010 Premium
PROD-00007 Basic
PROD-00003 Basic
PROD-00001 Standard
PROD-00008 Basic
PROD-00004 Ultimate
PROD-00017 Standard

Studio session
Messages have been cleared.

Id	Time	Category	Operation	Result	Result i...

You have started a new Monitor session.
No activity captured yet. Interact with your app in play mode to monitor its activity.

Connected Items: 0



DelayOutput On

PROD-00002 Basic
PROD-00009 Premium
PROD-00006 Basic
PROD-00005 Ultimate
PROD-00010 Premium
PROD-00007 Basic
PROD-00003 Basic
PROD-00001 Standard
PROD-00008 Basic
PROD-00004 Ultimate
PROD-00017 Standard

Studio session
Messages have been cleared.

Id	Time	Category	Operation	Result	Result i...

You have started a new Monitor session.
No activity captured yet. Interact with your app in play mode to monitor its activity.

Connected Items: 0

Common performance issues & best practices

7. Avoid N+1 queries

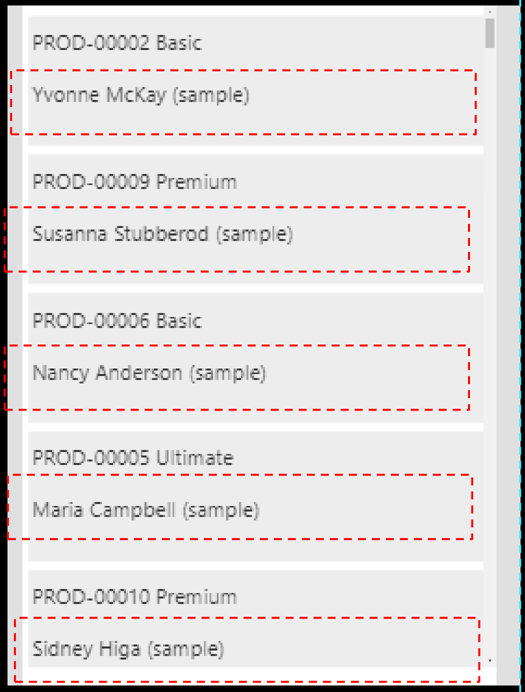
- Issue: Nested queries (lookups, filters, etc) in Gallery

Gallery: 1 network call
Control inside gallery: *n* network calls

Example: Gallery lists all contracts. Each contract record looks up Contact table for contact name.

LookUp(Contacts, Contact = ThisItem.Contact.Contact).'Full Name'

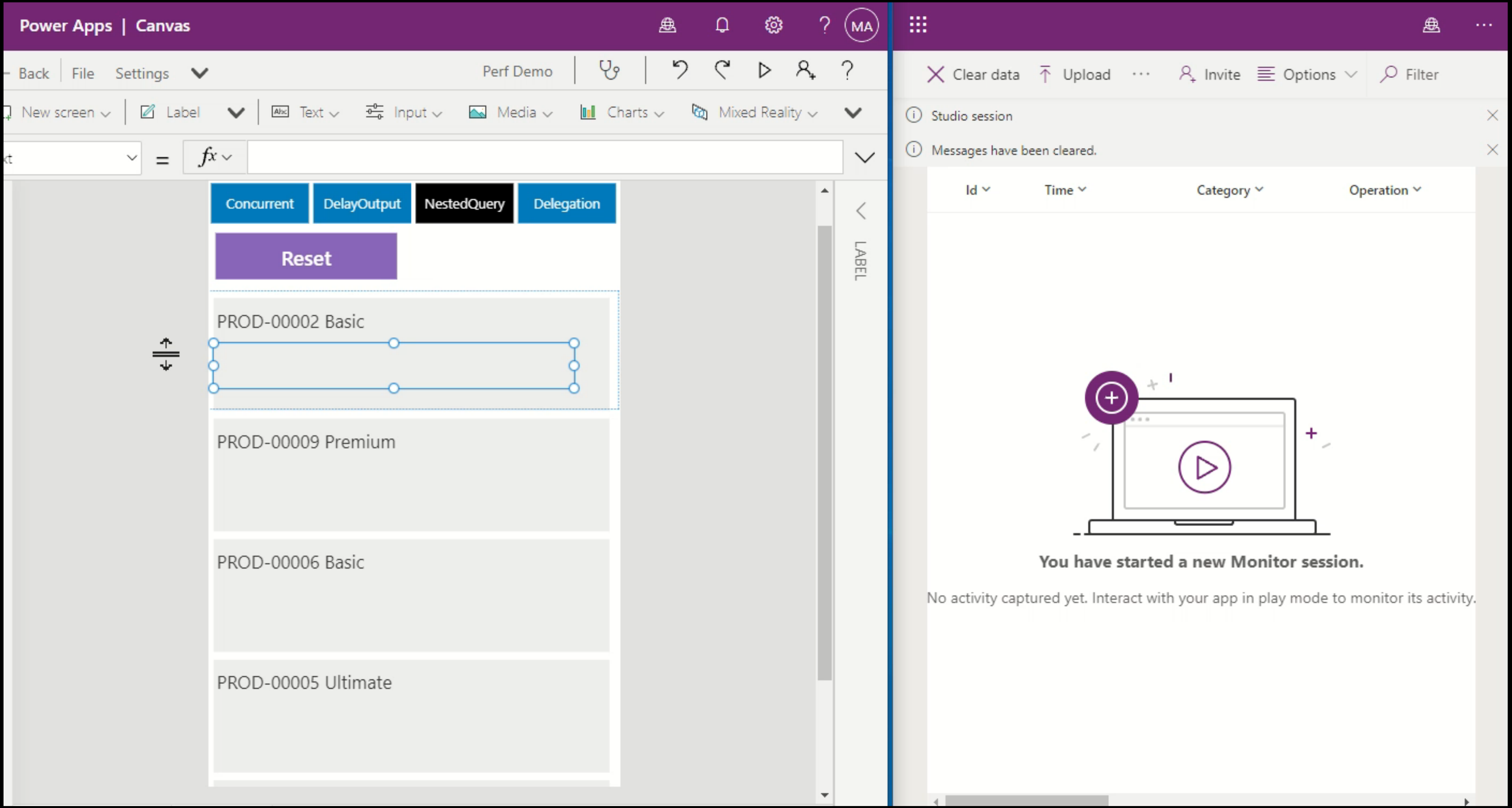
Items: "Contracts"


PROD-00002 Basic
Yvonne McKay (sample)
PROD-00009 Premium
Susanna Stubberod (sample)
PROD-00006 Basic
Nancy Anderson (sample)
PROD-00005 Ultimate
Maria Campbell (sample)
PROD-00010 Premium
Sidney Higa (sample)

- Best practice: Move nested query into a separate screen (eg a detail view). If using Dataverse, leverage relationship.

Common performance issues & best practices

7. Avoid N+1 queries



PROD-00002 Basic

PROD-00009 Premium

PROD-00006 Basic

PROD-00005 Ultimate

Studio session

Messages have been cleared.

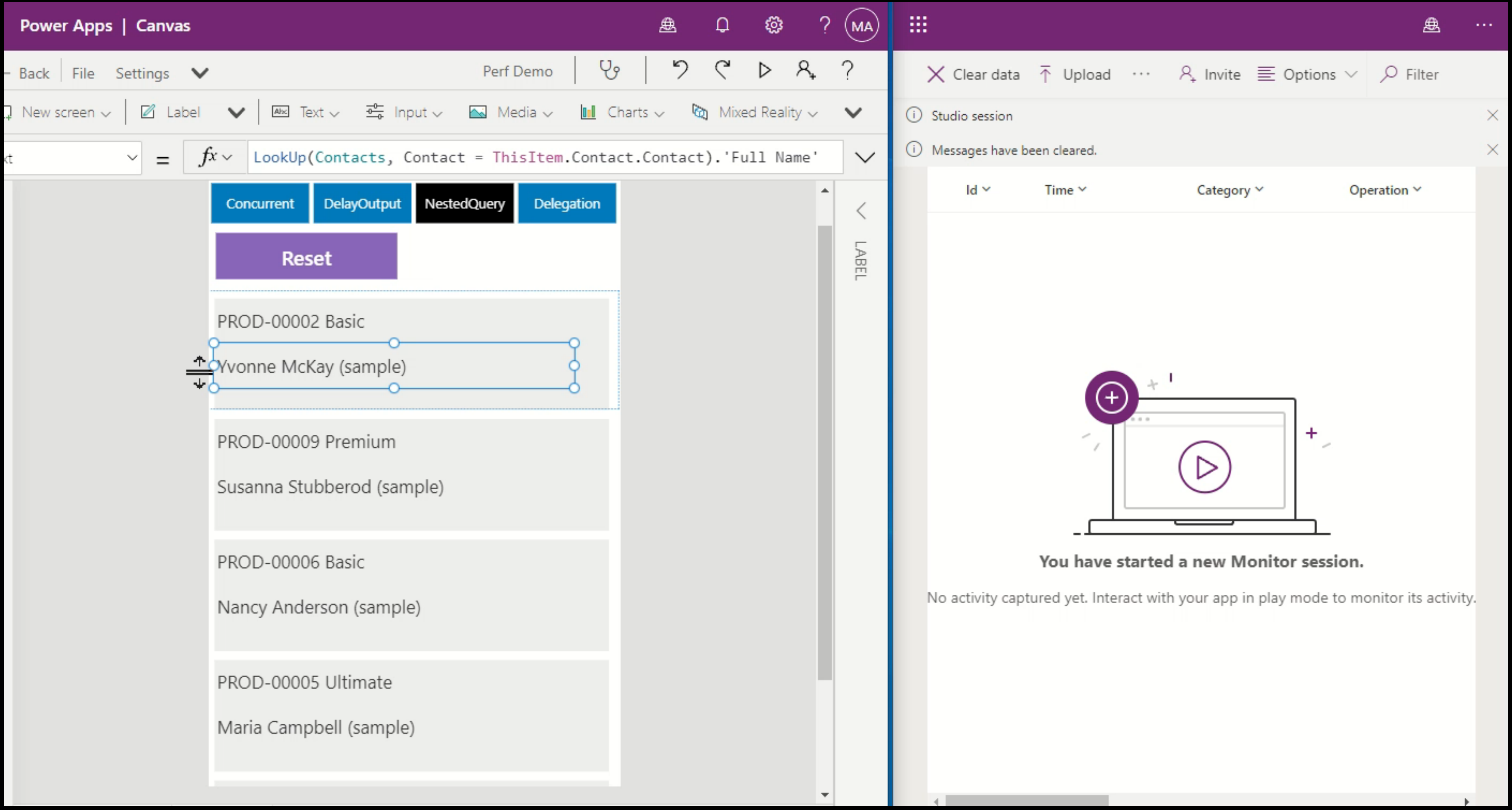
You have started a new Monitor session.

No activity captured yet. Interact with your app in play mode to monitor its activity.

Common performance issues & best practices

7. Avoid N+1 queries

Using relationship in Dataverse



```
LookUp(Contacts, Contact = ThisItem.Contact.Contact).'Full Name'
```

PROD-00002 Basic
Yvonne McKay (sample)
PROD-00009 Premium
Susanna Stubberod (sample)
PROD-00006 Basic
Nancy Anderson (sample)
PROD-00005 Ultimate
Maria Campbell (sample)

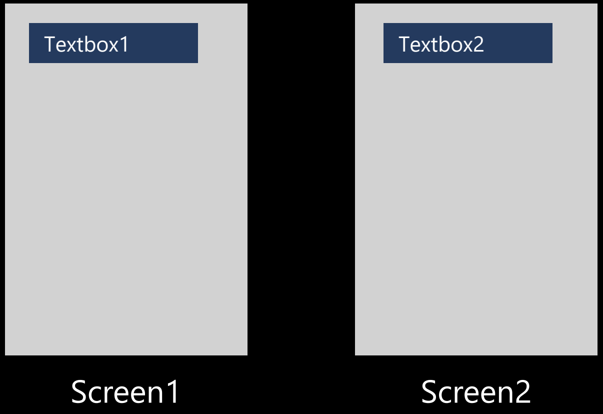
Studio session
Messages have been cleared.

You have started a new Monitor session.
No activity captured yet. Interact with your app in play mode to monitor its activity.

Common performance issues & best practices

8. Avoid control dependencies between screens

- Issue: Dependencies between screens creates more memory usage. Screens of an app are loaded into memory only as needed.
- Best practice: Use variables instead / Horizontal & Vertical container



```
Textbox2 DisplayMode = Textbox1.DisplayMode;
//creates dependencies between Screen1 and Screen2, causing both screens to be loaded in memory
```

```
Textbox2 DisplayMode = _displayModeVariable;
```

Common performance issues & best practices

9. Limit number of data connections and number of controls

- Issue: App have too many connections and/or controls, resulting in slow performance.
- Best practice:
  1. Don’t connect to more than 30 data sources in the same app.
  2. Don’t add more than 500 controls on the same app.
  3. Reuse component, container, use gallery instead of 2+related items

Avoid creating a mega-app. Break into smaller apps or consider Model Driven Apps for super-sophisticated/complex apps.

Common performance issues & best practices

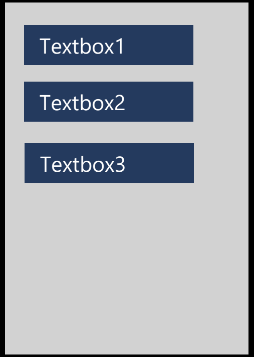
10. Efficient use of formulas

- Issue: Inefficient use of formulas causing unnecessary memory pressure or network calls.
- Best practice:
  - Consider formulas/functions that can achieve the same result in a single call vs multiple calls.
  - Example:

```
First(Filter(MyTasks, Category = “Important”))   //First + Filter results in the same operation as LookUp
LookUp(MyTask, Category = “Important”)

Filter(‘DataSource’, account.Name = “Hello World” && Status = “Current Status” && Value = ‘Rules’.Active)
//Use complex web API call instead of multiple simple call. Reduce network call numbers.
```

  - Reference pre-calculated results instead of re-calculating for multiple controls.
  - Example:



All Textbox visible property needs to be set based on a calculated result.
Don’t do this:

```
Textbox1 Visible = If(CountRows(Filter(_myData, Status = true))> 0, true, false);
Textbox2 Visible = If(CountRows(Filter(_myData, Status = true))> 0, true, false);
Textbox3 Visible = If(CountRows(Filter(_myData, Status = true))> 0, true, false);
```

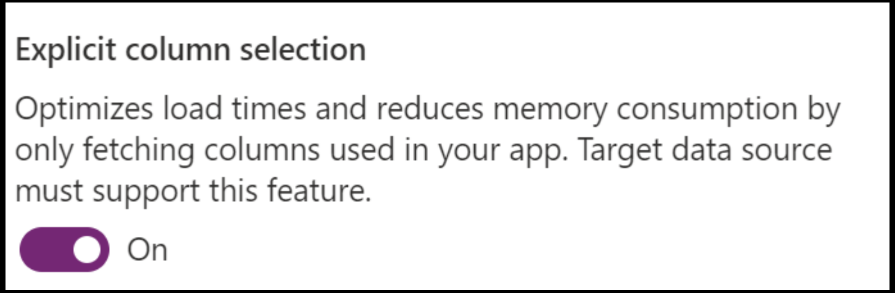
Instead, do this:

```
Textbox1 Visible = CountRows(Filter(_myData, Status = true))> 0;
Textbox2 Visible = Textbox1.Visible;
Textbox3 Visible = Textbox1.Visible;
```

Common performance issues & best practices

11. Reduce amount of data from connections

- Issue: App retrieves all columns from data connection, even if not used at all.
- Best practice: Turn on Explicit Column Selection. Power Apps will automatically reduce the number of columns fetched based on usage in the app. This is turned on by default – shouldn't have any good reason to turn this off. Older apps may not have this feature turned on – turn on in the settings and re-publish the app to take effect.
- Load data only when needed – use pagination to split into smaller queries. For example, when rendering a list of items by day and the app only shows items on a single day with a calendar navigation – there is no reason to load the entire table of items. Fetch only items for the current day view .



Explicit column selection

Optimizes load times and reduces memory consumption by only fetching columns used in your app. Target data source must support this feature.

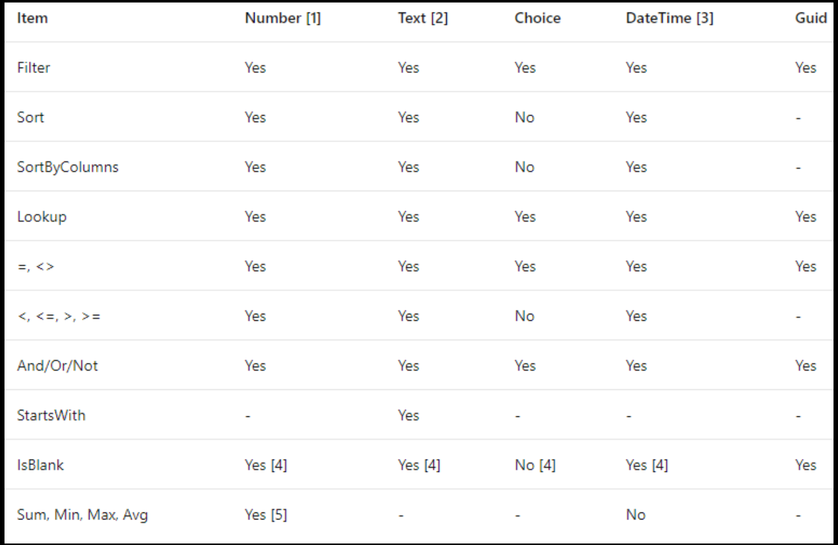
On

Common performance issues & best practices

12. Understand delegation limits

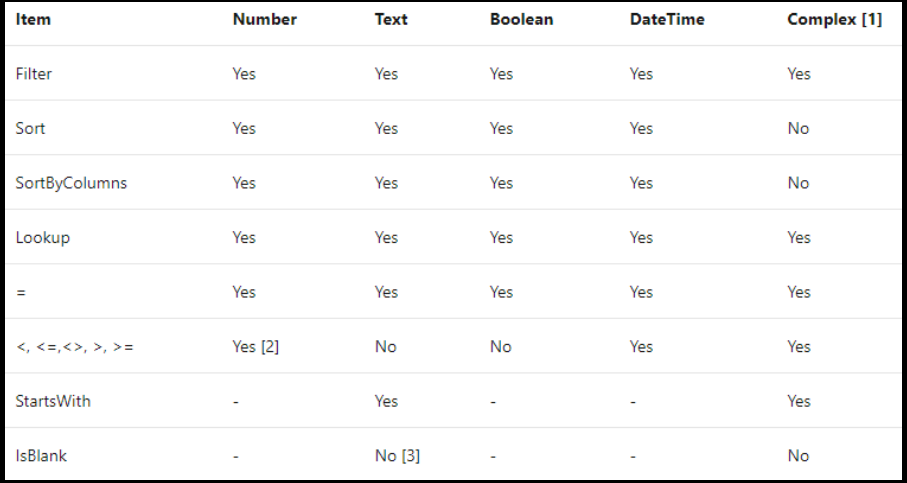
- Issue: Some operations cannot be delegated to the back-end for processing. Power Apps will fetch a limited set of records to be processed at the client-side. Default limit is 500 records and can be changed up to 2,000.
- Best practice: Avoid non-delegable queries. Use the correct data source when architecting solutions. SQL and Dataverse can utilize Views to perform filtering & sorting at the server-side.

Dataverse



Item	Number [1]	Text [2]	Choice	DateTime [3]	Guid
Filter	Yes	Yes	Yes	Yes	Yes
Sort	Yes	Yes	No	Yes	-
SortByColumns	Yes	Yes	No	Yes	-
Lookup	Yes	Yes	Yes	Yes	Yes
=, <>	Yes	Yes	Yes	Yes	Yes
<, <=, >, >=	Yes	Yes	No	Yes	-
And/Or/Not	Yes	Yes	Yes	Yes	Yes
StartsWith	-	Yes	-	-	-
IsBlank	Yes [4]	Yes [4]	No [4]	Yes [4]	Yes
Sum, Min, Max, Avg	Yes [5]	-	-	No	-

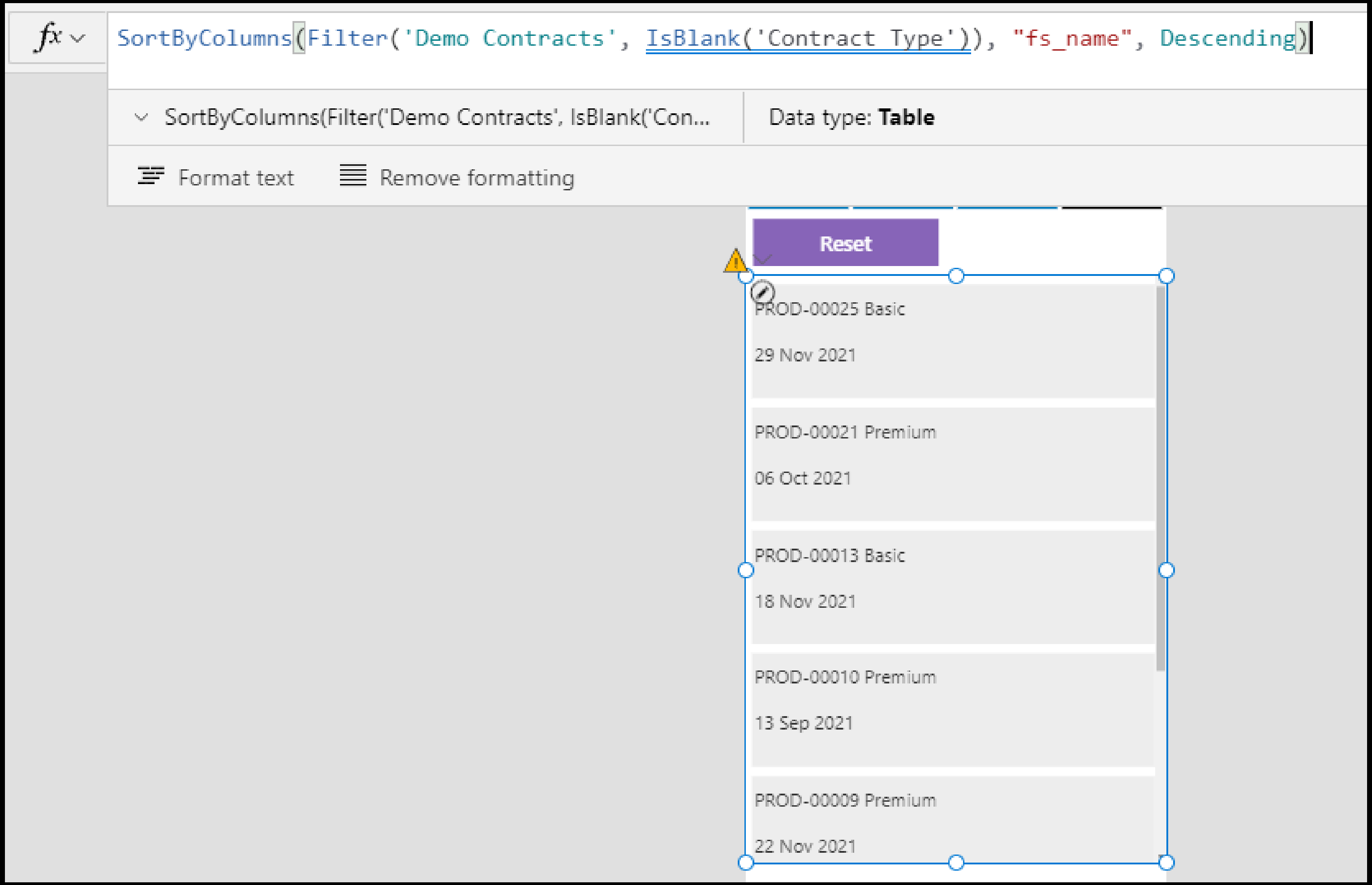
SharePoint



Item	Number	Text	Boolean	DateTime	Complex [1]
Filter	Yes	Yes	Yes	Yes	Yes
Sort	Yes	Yes	Yes	Yes	No
SortByColumns	Yes	Yes	Yes	Yes	No
Lookup	Yes	Yes	Yes	Yes	Yes
=	Yes	Yes	Yes	Yes	Yes
<, <=,<>, >, >=	Yes [2]	No	No	Yes	Yes
StartsWith	-	Yes	-	-	Yes
IsBlank	-	No [3]	-	-	No

Common performance issues & best practices

12. Understand delegation limits



```
SortByColumns(Filter('Demo Contracts', IsBlank('Contract Type')), "fs_name", Descending)
```

SortByColumns(Filter('Demo Contracts', IsBlank('Con...

Data type: **Table**

PROD-00025 Basic
29 Nov 2021

PROD-00021 Premium
06 Oct 2021

PROD-00013 Basic
18 Nov 2021

PROD-00010 Premium
13 Sep 2021

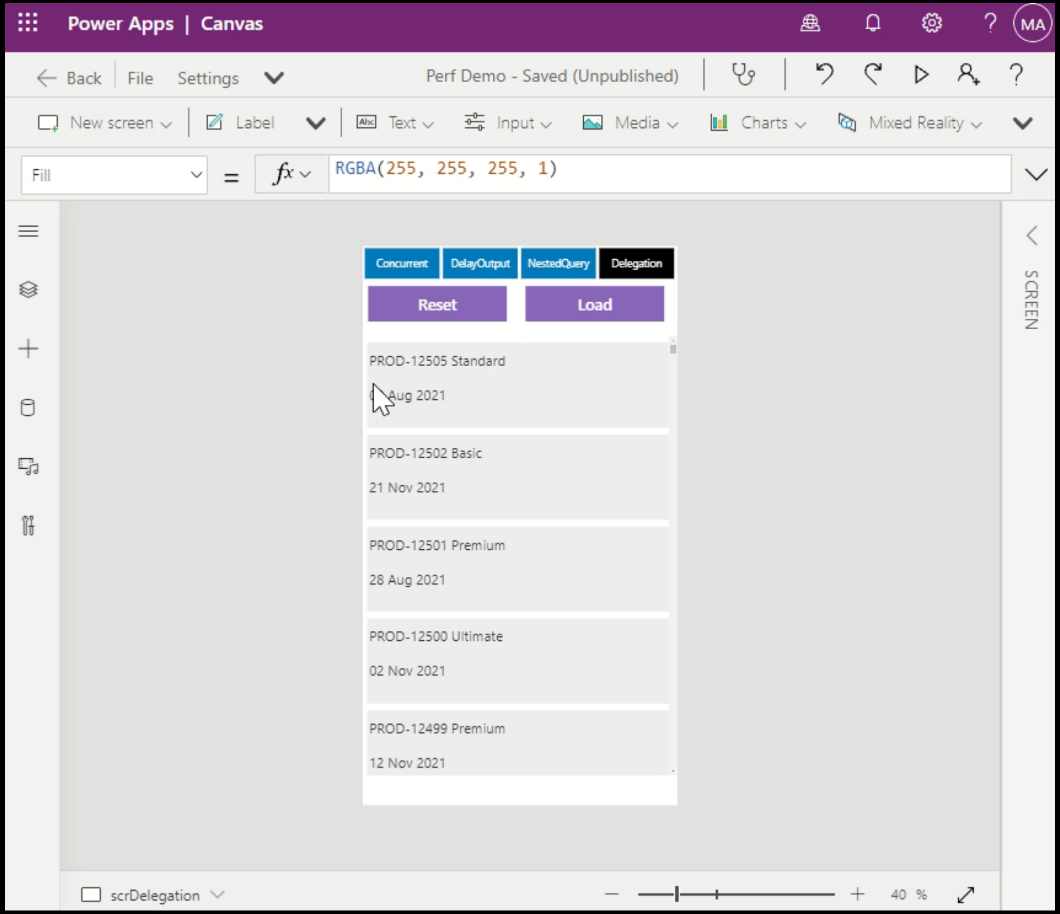
PROD-00009 Premium
22 Nov 2021

IsBlank() is not delegable

Common performance issues & best practices

12. Understand delegation limits

- Collect()/ClearCollect(), With{ … } only gets the first 500 (uses delegation limits)
- AsType(..) does not give delegation warnings, but it is non-delegable.



Fill

```
RGBA(255, 255, 255, 1)
```

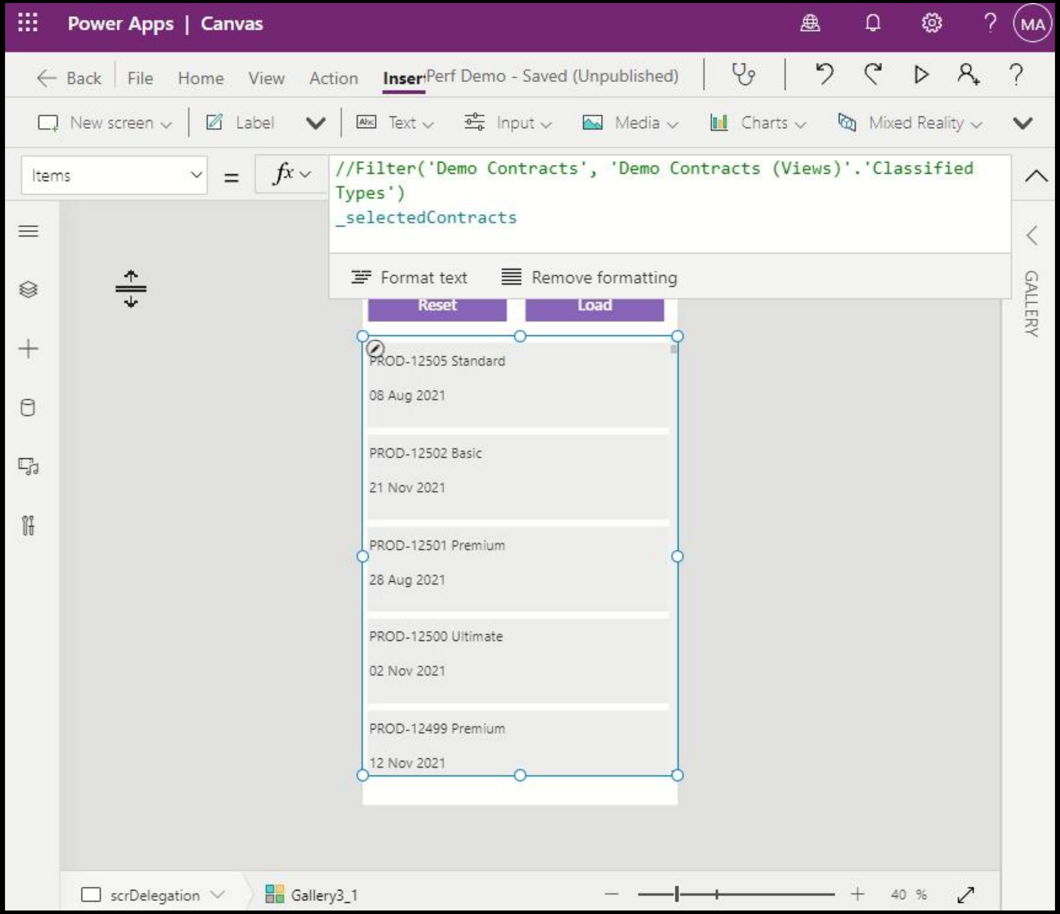
PROD-12505 Standard
Aug 2021

PROD-12502 Basic
21 Nov 2021

PROD-12501 Premium
28 Aug 2021

PROD-12500 Ultimate
02 Nov 2021

PROD-12499 Premium
12 Nov 2021



Items

```
//Filter('Demo Contracts', 'Demo Contracts (Views)'.'Classified
Types')
_selectedContracts
```

PROD-12505 Standard
08 Aug 2021

PROD-12502 Basic
21 Nov 2021

PROD-12501 Premium
28 Aug 2021

PROD-12500 Ultimate
02 Nov 2021

PROD-12499 Premium
12 Nov 2021

Common performance issues & best practices

13. Compress app assets

- Issue: App uses very large images – eg for screen backgrounds, load screens, etc.
- Best practice: Compress media assets. Background images do not need to be super-highres. Use “Fill” option for image controls.

  - https://compressor.io/
  - https://tinypng.com/
  - http://compressjpeg.com/

Common performance issues & best practices

14. Offload long-running processes from the front-end

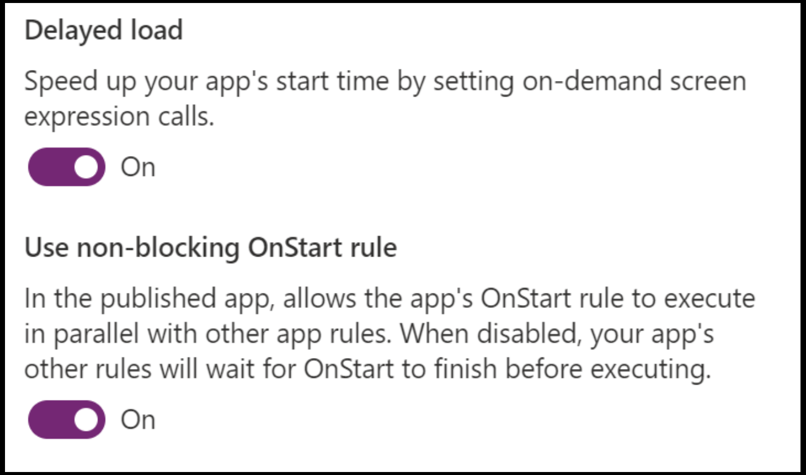
- Issue: App uses long running processes – eg a sequence of operations that loops through multiple records and update with conditions
- Best practice: Consider off-loading to Power Automate to reduce network calls on the client and reduce client-side operations.
- Several techniques:
  - Power Automate can return acknowledgement/initial results before completing longer running processes.
  - Option for "Fire-and-forget" pattern – Power Apps sends data to Power Automate without waiting for completion.
  - If using Dataverse – options to use FetchXML for complex queries (joins, nested look-ups, etc).

  - Next Session will cover options with Power Automate.

Common performance issues & best practices

15. App optimization settings + Republish app regularly

Preview: (Turned on by default)



Delayed load

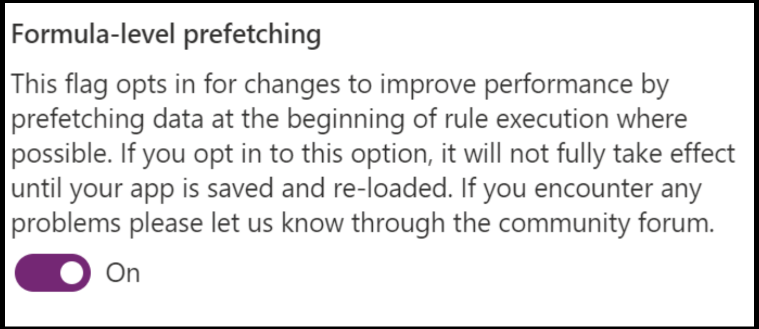
Speed up your app's start time by setting on-demand screen expression calls.

On

Use non-blocking OnStart rule

In the published app, allows the app's OnStart rule to execute in parallel with other app rules. When disabled, your app's other rules will wait for OnStart to finish before executing.

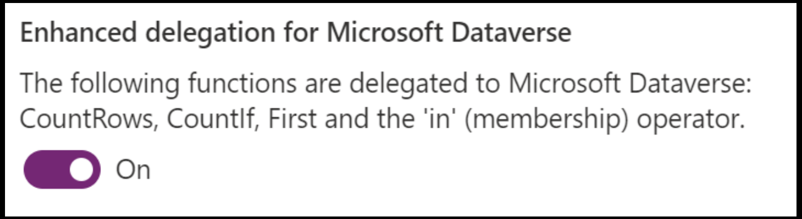
On



Formula-level prefetching

This flag opts in for changes to improve performance by prefetching data at the beginning of rule execution where possible. If you opt in to this option, it will not fully take effect until your app is saved and re-loaded. If you encounter any problems please let us know through the community forum.

On

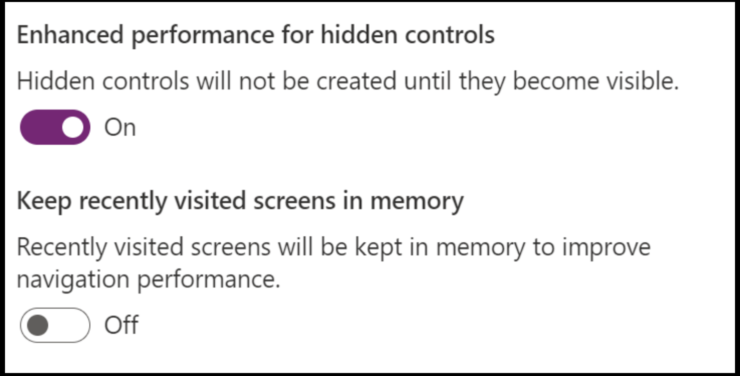


Enhanced delegation for Microsoft Dataverse

The following functions are delegated to Microsoft Dataverse: CountRows, CountIf, First and the 'in' (membership) operator.

On

Experimental: (Turned off by default)



Enhanced performance for hidden controls

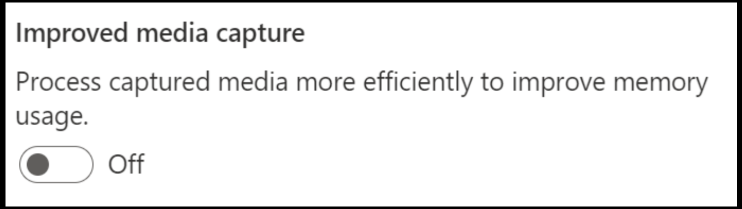
Hidden controls will not be created until they become visible.

On

Keep recently visited screens in memory

Recently visited screens will be kept in memory to improve navigation performance.

Off



Improved media capture

Process captured media more efficiently to improve memory usage.

Off

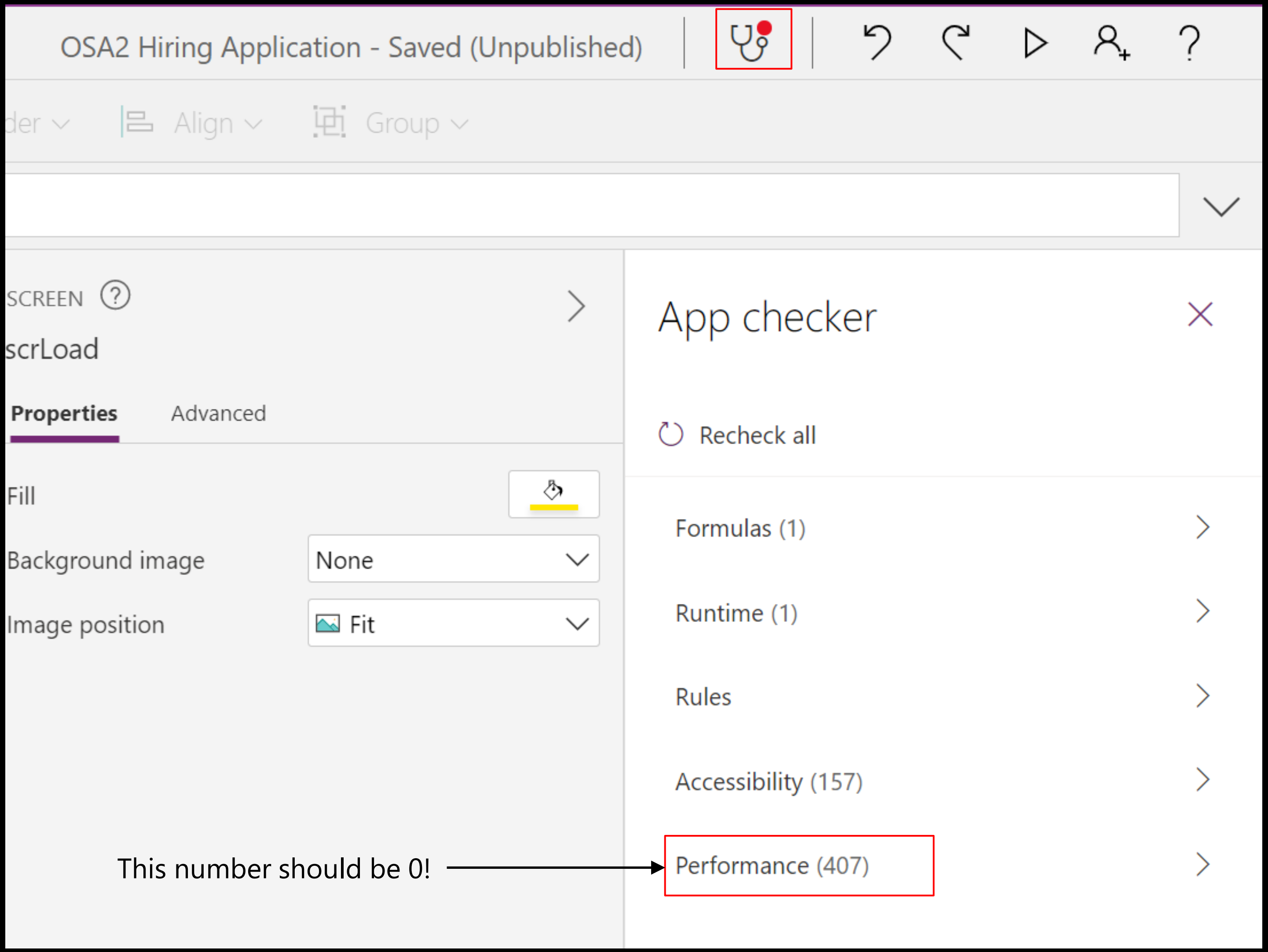
Common performance issues & best practices

16. Scope creep is real.

App that does 5 things very well > App that does 50 things badly

Tools to Monitor/Troubleshoot

1. Use App Checker



OSA2 Hiring Application - Saved (Unpublished)

der Align Group

SCREEN

scrLoad

Properties Advanced

Fill

Background image None

Image position Fit

App checker

Recheck all

Formulas (1)

Runtime (1)

Rules

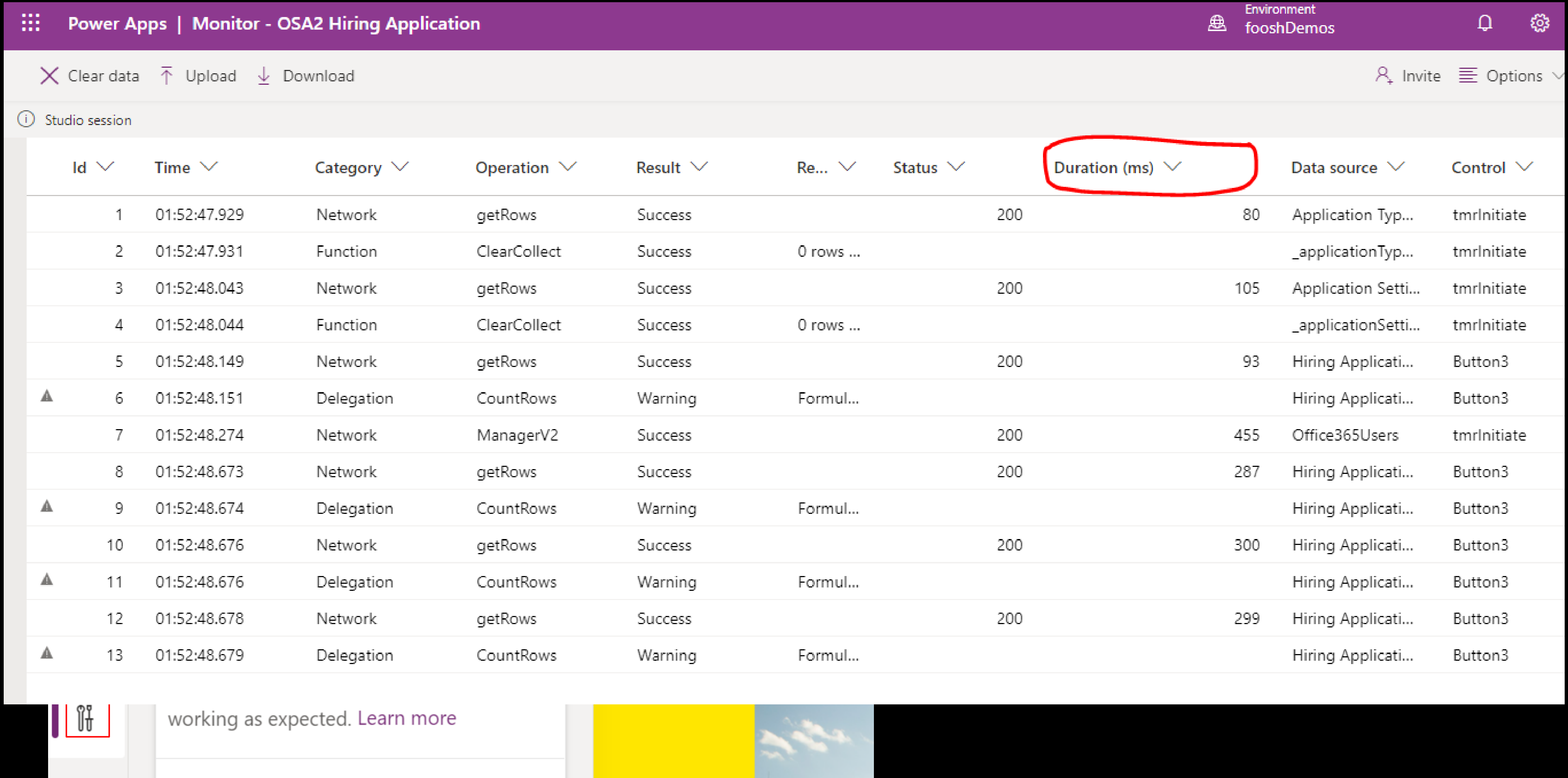
Accessibility (157)

Performance (407)

This number should be 0!

Tools to Monitor/Troubleshoot

2. Use Monitor

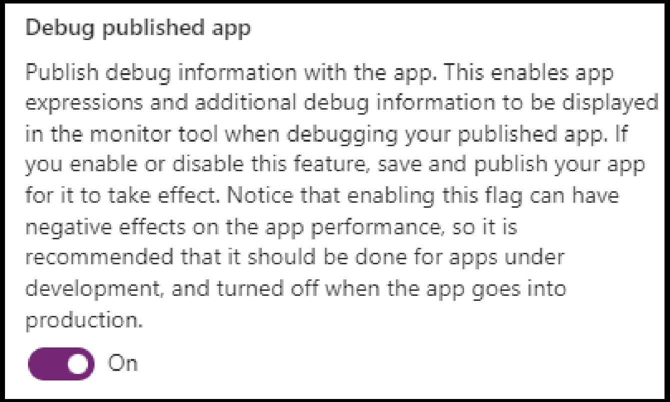


	Id	Time	Category	Operation	Result	Re...	Status	Duration (ms)	Data source	Control
	1	01:52:47.929	Network	getRows	Success		200	80	Application Typ...	tmrInitiate
	2	01:52:47.931	Function	ClearCollect	Success	0 rows ...			_applicationTyp...	tmrInitiate
	3	01:52:48.043	Network	getRows	Success		200	105	Application Setti...	tmrInitiate
	4	01:52:48.044	Function	ClearCollect	Success	0 rows ...			_applicationSetti...	tmrInitiate
	5	01:52:48.149	Network	getRows	Success		200	93	Hiring Applicati...	Button3
▲	6	01:52:48.151	Delegation	CountRows	Warning	Formul...			Hiring Applicati...	Button3
	7	01:52:48.274	Network	ManagerV2	Success		200	455	Office365Users	tmrInitiate
	8	01:52:48.673	Network	getRows	Success		200	287	Hiring Applicati...	Button3
▲	9	01:52:48.674	Delegation	CountRows	Warning	Formul...			Hiring Applicati...	Button3
	10	01:52:48.676	Network	getRows	Success		200	300	Hiring Applicati...	Button3
▲	11	01:52:48.676	Delegation	CountRows	Warning	Formul...			Hiring Applicati...	Button3
	12	01:52:48.678	Network	getRows	Success		200	299	Hiring Applicati...	Button3
▲	13	01:52:48.679	Delegation	CountRows	Warning	Formul...			Hiring Applicati...	Button3

working as expected. Learn more

Tools to Monitor/Troubleshoot

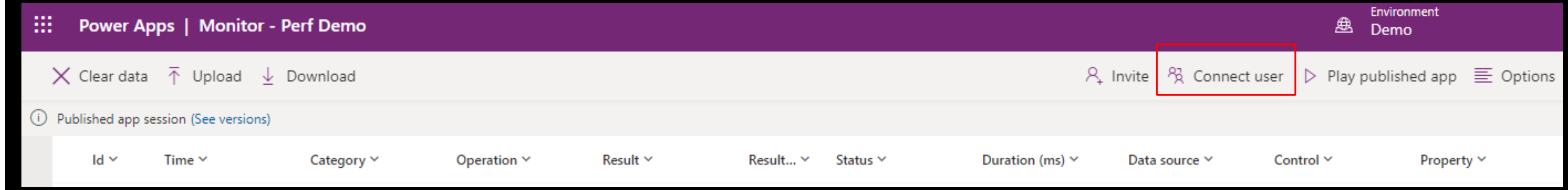
2. Use Monitor – Enable for Published apps

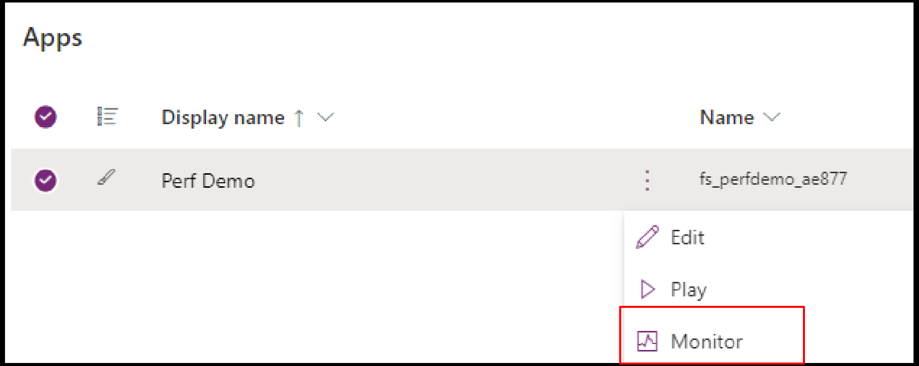


Debug published app

Publish debug information with the app. This enables app expressions and additional debug information to be displayed in the monitor tool when debugging your published app. If you enable or disable this feature, save and publish your app for it to take effect. Notice that enabling this flag can have negative effects on the app performance, so it is recommended that it should be done for apps under development, and turned off when the app goes into production.

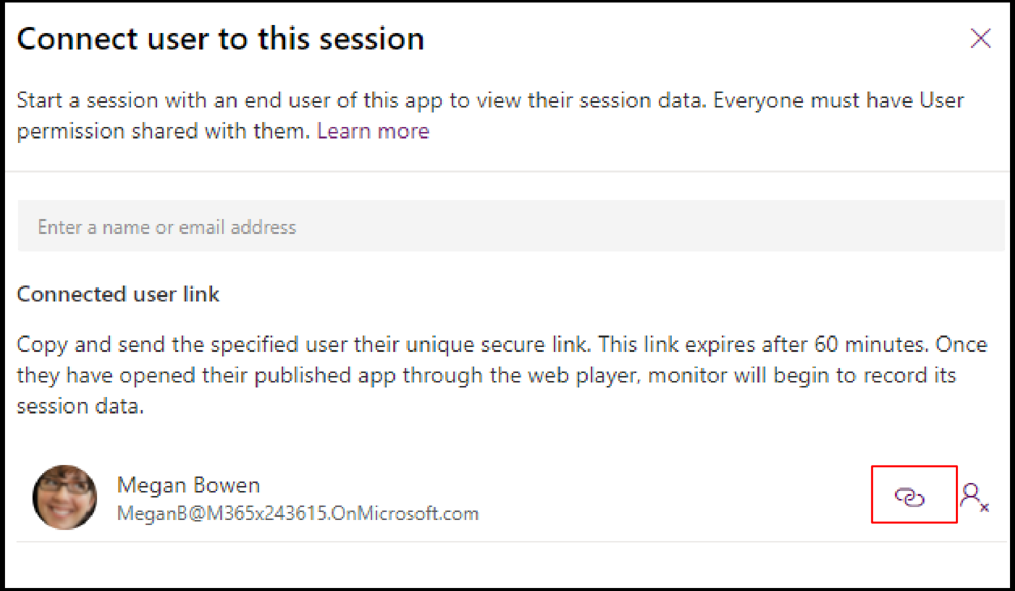
On





Apps

		Display name ↑	Name
✓		Perf Demo	fs_perfdemo_ae877



Connect user to this session

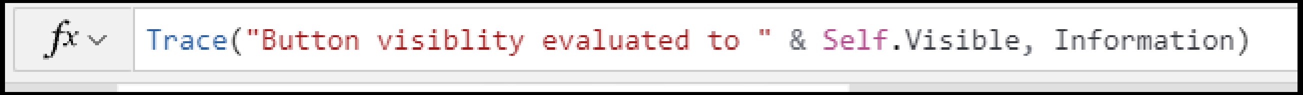
Start a session with an end user of this app to view their session data. Everyone must have User permission shared with them. Learn more

Enter a name or email address

Connected user link

Copy and send the specified user their unique secure link. This link expires after 60 minutes. Once they have opened their published app through the web player, monitor will begin to record its session data.

Megan Bowen
MeganB@M365x243615.OnMicrosoft.com



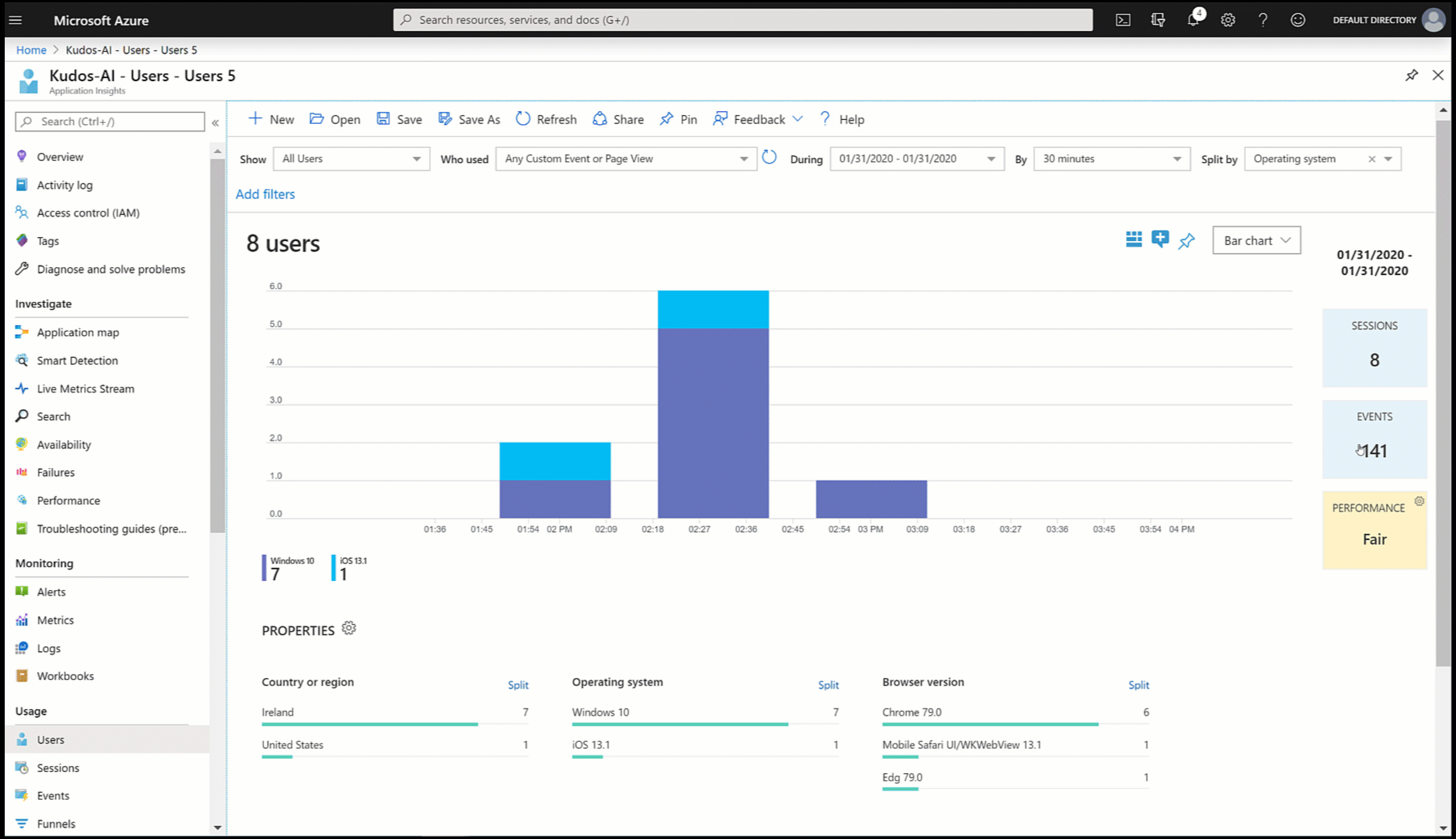
```
fx  Trace("Button visiblity evaluated to " & Self.Visible, Information)
```

Use Trace function to emit information into Monitor

Tools to Monitor/Troubleshoot

3. Use Azure App Insights (for Pro Devs)

[Analyze telemetry of a canvas app using Application Insights - Power Apps | Microsoft Docs]



Show: All Users | Who used: Any Custom Event or Page View | During: 01/31/2020 - 01/31/2020 | By: 30 minutes | Split by: Operating system

8 users

01/31/2020 - 01/31/2020

SESSIONS 8

EVENTS 141

PERFORMANCE Fair

Windows 10: 7 | iOS 13.1: 1

PROPERTIES

Country or region
Ireland	7
United States	1

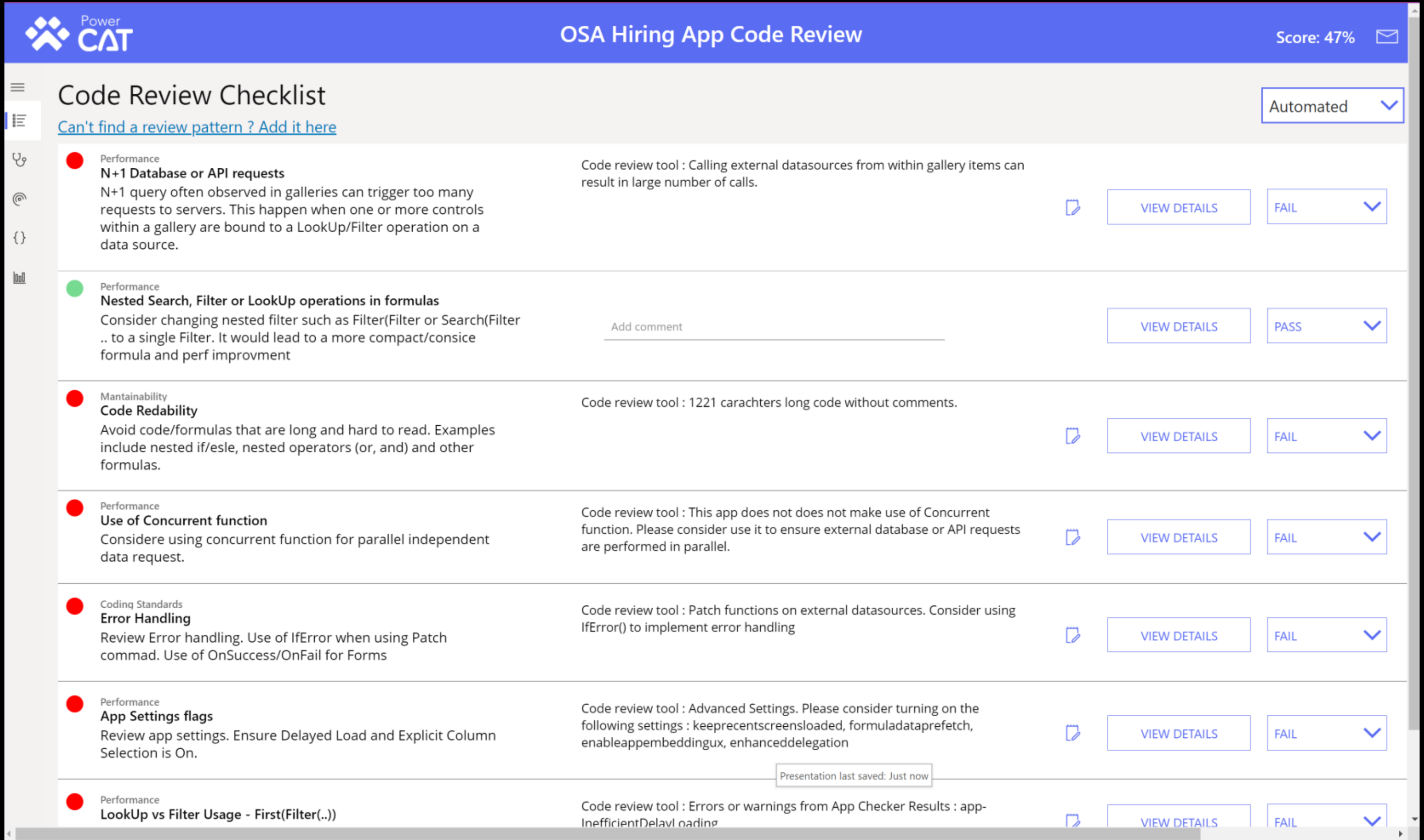
Operating system	
Windows 10	7
iOS 13.1	1

Browser version	
Chrome 79.0	6
Mobile Safari UI/WKWebView 13.1	1
Edg 79.0	1

Tools to Monitor/Troubleshoot

4. Download and use the Power Apps Code Review Tool!

[Power Apps Code Review Tool | Microsoft Power Apps]()



Power CAT — OSA Hiring App Code Review — Score: 47%

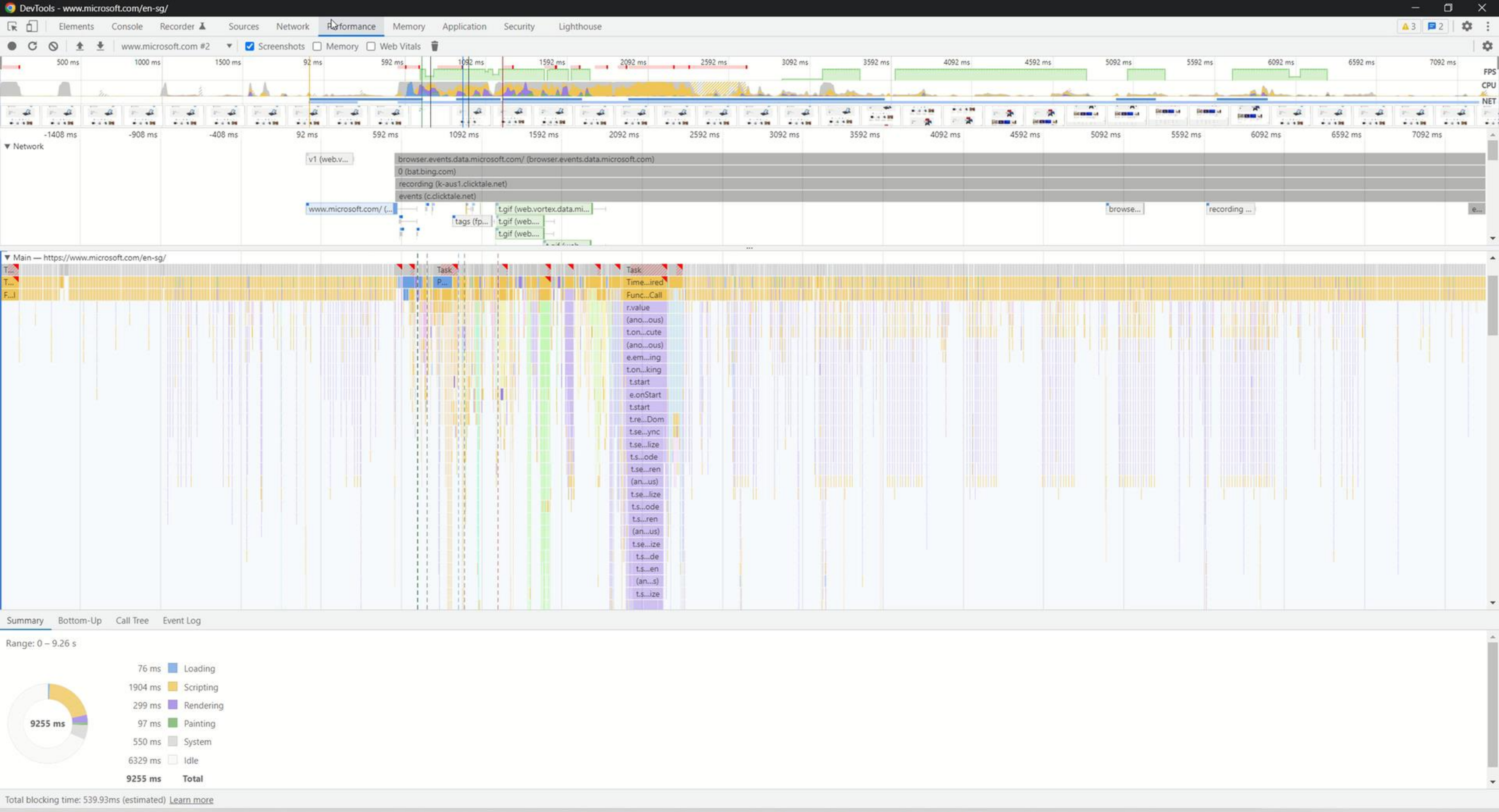
Code Review Checklist

Can't find a review pattern ? Add it here

Automated

🔴	Performance **N+1 Database or API requests** N+1 query often observed in galleries can trigger too many requests to servers. This happen when one or more controls within a gallery are bound to a LookUp/Filter operation on a data source.	Code review tool : Calling external datasources from within gallery items can result in large number of calls.	VIEW DETAILS	FAIL
🟢	Performance **Nested Search, Filter or LookUp operations in formulas** Consider changing nested filter such as Filter(Filter or Search(Filter .. to a single Filter. It would lead to a more compact/consice formula and perf improvment	Add comment	VIEW DETAILS	PASS
🔴	Mantainability **Code Redability** Avoid code/formulas that are long and hard to read. Examples include nested if/esle, nested operators (or, and) and other formulas.	Code review tool : 1221 carachters long code without comments.	VIEW DETAILS	FAIL
🔴	Performance **Use of Concurrent function** Considere using concurrent function for parallel independent data request.	Code review tool : This app does not does not make use of Concurrent function. Please consider use it to ensure external database or API requests are performed in parallel.	VIEW DETAILS	FAIL
🔴	Coding Standards **Error Handling** Review Error handling. Use of IfError when using Patch commad. Use of OnSuccess/OnFail for Forms	Code review tool : Patch functions on external datasources. Consider using IfError() to implement error handling	VIEW DETAILS	FAIL
🔴	Performance **App Settings flags** Review app settings. Ensure Delayed Load and Explicit Column Selection is On.	Code review tool : Advanced Settings. Please consider turning on the following settings : keeprecentscreensloaded, formuladataprefetch, enableappembeddingux, enhanceddelegation	VIEW DETAILS	FAIL
🔴	Performance **LookUp vs Filter Usage - First(Filter(..))**	Code review tool : Errors or warnings from App Checker Results : app-InefficientDelayLoading	VIEW DETAILS	FAIL

Presentation last saved: Just now



Range: 0 – 9.26 s

76 ms	🟦 Loading
1904 ms	🟨 Scripting
299 ms	🟪 Rendering
97 ms	🟩 Painting
550 ms	⬜ System
6329 ms	⬜ Idle
9255 ms	**Total**

9255 ms

Total blocking time: 539.93ms (estimated) Learn more

6:17 PM
5/25/2022

Questions, Comments, Feedback

Common sources of slow performance for a canvas app - Power Apps | Microsoft Docs

Common canvas apps performance issues and resolutions - Power Apps | Microsoft Docs

Tips and best practices to improve performance of canvas apps - Power Apps | Microsoft Docs

Understand delegation in a canvas app - Power Apps | Microsoft Docs

Coding Standards & Guidelines - https://pahandsonlab.blob.core.windows.net/documents/PowerApps canvas app coding standards and guidelines.pdf

Analyze telemetry of a canvas app using Application Insights - Power Apps | Microsoft Docs

Monitor overview - Power Apps | Microsoft Docs

Power Apps Code Review Tool | Microsoft Power Apps