# App Dev Team Requests PowerApp

This app is designed to replace the service request process for the Application Development team in SharePoint 2013. This app enables users to create, update, and view service requests as well as all associated documentation and related checklists.

## How it works:

PowerApps is creating a way to interface with many different SharePoint lists, and triggers workflows when needed. Each of the SharePoint lists can still be accessed via the [Web and Mobile SharePoint site](https://tdsinc.sharepoint.com/sites/WebandMobileDevelopment). Another way to get to this site is from the web and mobile Microsoft Team clicking the ellipsis in the upper right corner and choosing “Open in SharePoint”.

## Editing/Sharing:

To share this App, go to [Power Apps - TDS Corp - PROD Environment](https://make.powerapps.com/environments/55ad6321-9988-420e-9caf-8e9f817d539c/apps). Next to AppDev Team Requests, click the Ellipsis. From there you’ll have access to edit or share if you are a co-owner.

## Associated SharePoint lists/libraries used in the Power App:

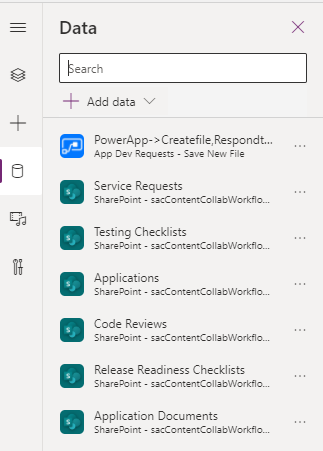
**Service Requests:** This is where each of the service requests are stored. Creating a new service request will automatically create associated application documents, a code review checklist, a testing checklist, and a release readiness checklist regardless of whether created via PowerApps or directly from the SharePoint list.

**Application Documents:** Storage location of all application documents

**Code Review Checklist, Release Checklist, Testing Checklist**: Storage locations of the checklists associated with each request.

**Application Document Templates:** This is where the templates used to create the application documents are stored. These can be edited, but it is important to not change the title of these documents, or remove/edit the document template properties as this will cause issues with the workflow that runs to create new documents from the template.

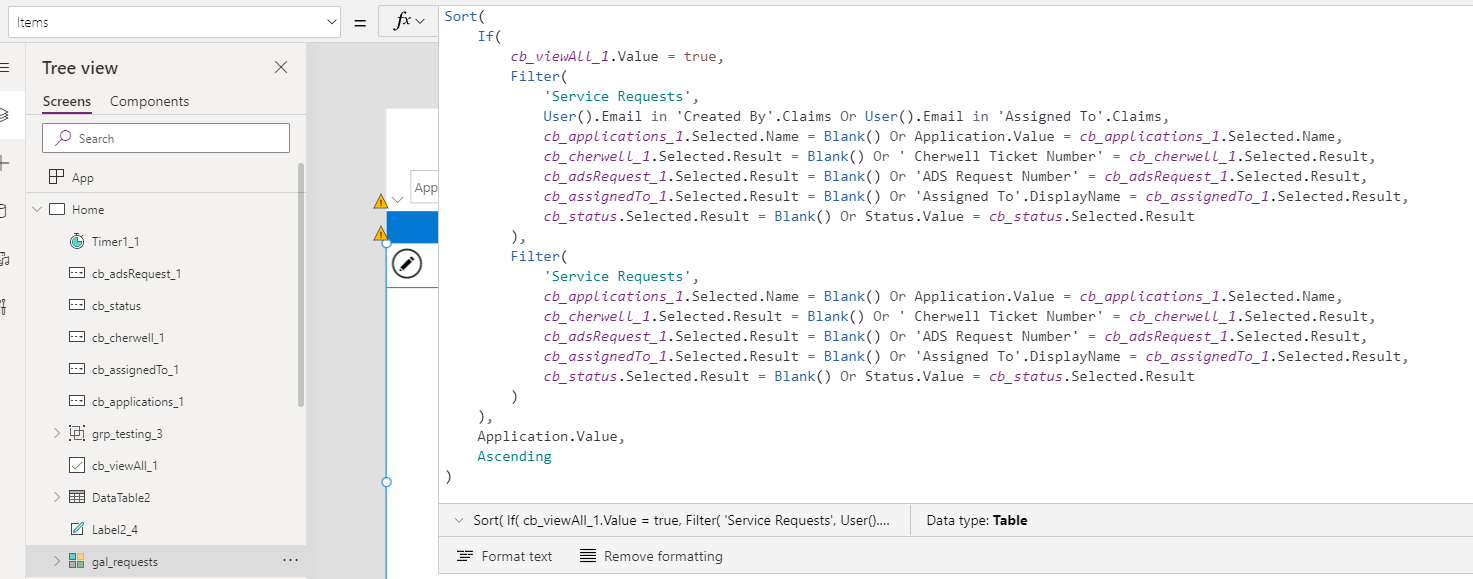
The data in these lists/libraries are connected to the app using Connectors. Connectors can be viewed by clicking the connectors button on the left side of the screen. The connectors in this app were configured using a service account owned by the Content and Collaboration team. The names of the connectors are also often used in formulas throughout the app.



## Screens

**Home**:

This is the default screen when opening the app. It shows a view of all requests, sorted by Application. Additional dropdowns are available to enable filtering by Application, Status, Assigned To, Cherwell Ticket #, and ADS Request #. You may also see your requests only by clicking the “My Requests” checkbox.

The following formula, which you can get to by clicking on gal\_requests (the big table where you can see all items) and then choosing the Items property, is where the sort/filter logic comes from: 

Another thing to note on the home screen is the Timer object. The object itself is hidden, but when it expires, it checks to see if the paramer “requestId” exists. If so, it will navigate to the detail screen, and set varSelectedItem to the request whose Id is the paramter that was added to the end of the url. This url is built on the Detail Screen page.

Pressing “New Request” will navigate to the Detail Screen and set varFormMode to “New”

Selecting the arrow next to an item in the gallery will navitage to the Detail Screen, set the form mode to “View”, and set varSelectedItem to the item selected.

**Detail Screen:**

This screen is where all viewing/editing/creating of requests occurs. The state of the form (frm\_requests) depends on the global variable varFormMode. If in view or edit mode, it will show the properties of the global variable varSelectedItem. The Save button triggers validation rules (requiring fields) and saves the data to the SharePoint list. When this save is successful, the “OnSuccess” event of frm\_requests occurs sets varSelectedItem to the most recently submitted item and changes varFormMode to “View”.

Creating a new item will also trigger the flow “Create Checklists and Documents” (detailed below)

When in view or edit mode, you’ll be able to copy a link directly to the item. The URL itself is built in the lbl\_URL control which is hidden on the screen, and then the lbl\_Link control uses the url to build the actual link. (It was done this way because of quote weirdness)

Above the form you’ll find links to the Code Review, Release, and Testing checklists. The color of these will change to green when completed using a formula that checks if the associated checklist has a Completed value of true: If(First(Filter('Code Reviews', 'Service Request Id'.Value = txt\_Id.Text)).Completed = true, Green, RGBA(0, 120, 212, 1))

On the right side of the screen is the Application Documents gallery. When a request is first created, the “Create Checklists and Documents” flow will automatically create templated documents, they just may take a minute or two to show up. This gallery will show all documents with the same service request id as the request displayed in the form. Clicking on the arrow icon will allow you to view the document.

Adding a new document is a little convoluted, so I’ll just share a link to the tutorial I followed to make this work: <https://www.youtube.com/watch?v=3QaiM8SeWfM>

Essentially, we are disguising an attachment control with an icon, converting it to an image control so we can get the contents, and then sending that information to Power Automate so that it can be converted to binary data and uploaded to SharePoint.

You should also be able to click and drag files onto this icon to upload new documents

**Checklists:**

The checklist pages all work very similarly. The form at the top of the page searches for a checklist with a Service Request Id that matches the id of varSelectedItem

## Workflows

These can be viewed by going to [Power Automate - TDS Corp - PROD Environment](https://us.flow.microsoft.com/manage/environments/55ad6321-9988-420e-9caf-8e9f817d539c/flows). Choose “Shared with me”.

**Create Checklists and Documents**

This workflow is triggers whenever a new item is added to the Service Requests list (regardless of whether PowerApps was used or not)

It first creates items in each of the Checklists, using the Service Request Id field as the link between the service request and the checklist

Next, it creates the default application documents. For each of the documents, the flow will find the template document in Application Document Templates and populate the dynamic content fields using the “Populate Word Document Template” action. Note that this is a premium connector, but since the connection was made under a service account with a premium license, and the flow isn’t directly connected to the Power App this shouldn’t be an issue.

Once the template is filled in, the file content is used to create a new file in the Application Documents library. The final step populates metadata associate with the Service Request to link the document to the request.

**App Dev Requests – Save New File**

This workflow is triggered from PowerApps. It takes the file content, name of a file, and Service Request ID as parameters. This flow converts the file content to binary, and then creates a new file in SharePoint with that content. It then updates the file properties to associate it with the service request