Proposal Manager

Extensibility Guide

Version 1.0

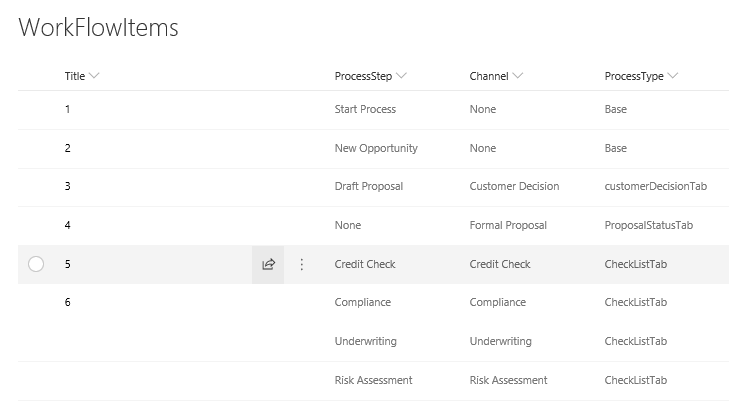
# About Extensibility

Proposal Manager lets you customize the process steps that your team uses to work on the different deal types. In Proposal Manager, a deal type is composed by process steps. Each process step has a channel in the opportunity team, and each can have a custom Microsoft Teams tab. In this document we show how to create your own extension to add personalized steps to Proposal Manager with your own code.

# How to extend

SharePoint:

* Add a new row to the WorkFlow Items List for example Customer Feedback



For customer feedback the new row might look like that:

|  |  |  |  |
| --- | --- | --- | --- |
| 9 | CustomerFeedback | Customer Feedback | FeedbackTab |

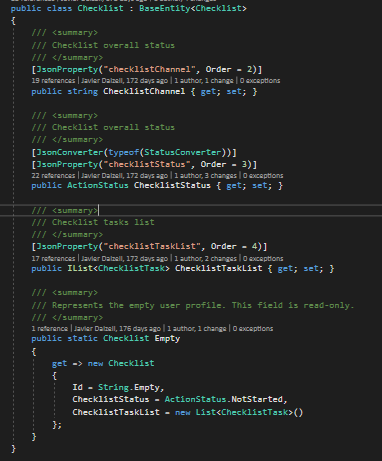
* Add the permissions needed to the permissions list.

The new permissions will be: (CustomerFeedback \_Read, CustomerFeedback \_ReadWrite)

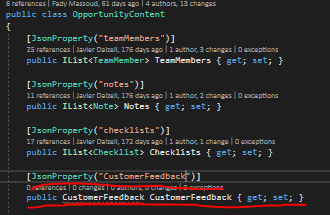
We are following the same naming convension used for Graph Api : https://developer.microsoft.com/en-us/graph/docs/concepts/permissions\_reference

MiddleTear:

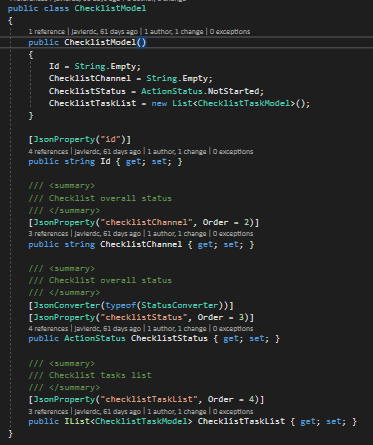
* Create a new class in (ApplicationCore\Entities) and name it CustomerFeedback this class will inherit from the base entity class. The CustomerFeedback is an object that contains all the information that we want to store in SharePoint for this process. We currently have one for checklist and CustomerDecision. This is a sample of the one that we currently have for checklist:



* After creating the CustomerFeedback Entity the next step would be to add this Entity to the opportunity artifact that gets saved as a json object into the Opportunities SharePoint list. The opportunity artifact is in (ApplicationCore\Artifacts\Opportunity.cs). Search for the opportunity content class then add the CustomerFeedback there. And this is an example of how it should look like:



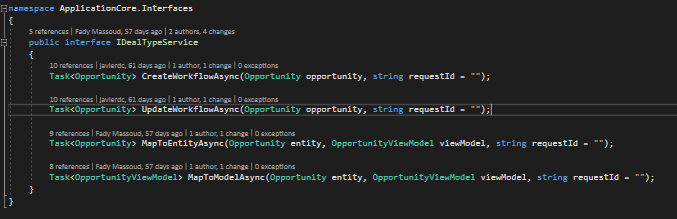
* Next you would need to create a Model for the CustomerFeedback in (ApplicationCore\Models). This model should contain the data exposed that you want to expose to the client. This is an example of the model used for checklists:



* The model Created in the previous step needs to be referenced in to the opportunity view model Located in (ApplicationCore\ViewModels\OpportunityViewModel).

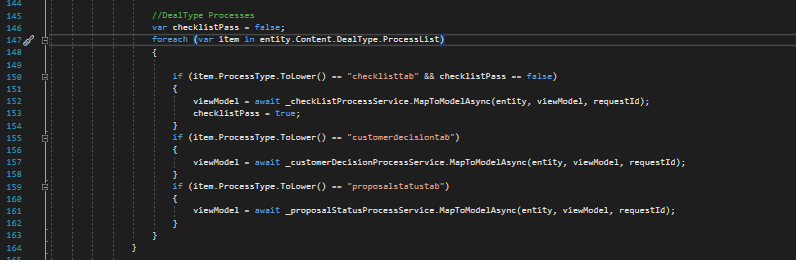
Note: Make sure when creating the entity, model and adding the references in the artifact and view model to use the correct json property name because this is needed when serializing the object at the API level. Also make sure to use the StatusConverter and the ActionStatus class when referencing anything related to status.

* After creating the entity and the model you would need to create the service that handles the mapping and the business logic for the CustomerFeedback process. This service will be added at (Infrastructure\DealTypeServices). This service should implement the IDealTypeService interface. These are the methods that should be implemented:



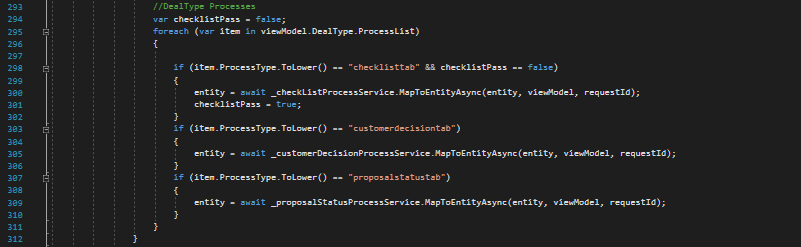
MapToEntityAsync will be used to map the Entity object created in the first step to the model object created in the third step and MapToModelAsync will do the opposite. The CreateWorkflowAsync and UpdateWorkflowAsync methods will contain the business logic for CustomerFeedback when creating and updating an opportunity.

* The logic for mapping the opportunity ViewModel to Entity and vise versa is located at (Infrastructure\Helpers\OpportunityHelpers.cs). What you will need to do is in the OpportunityToViewModelAsync method in this foreach loop:

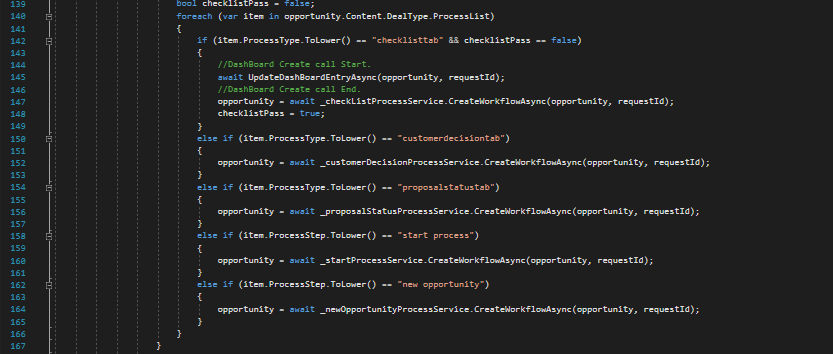


Another if statement needs to be added for the FeedbackTab which calls the MapToModelAsync method created in the CustomerFeedback service.

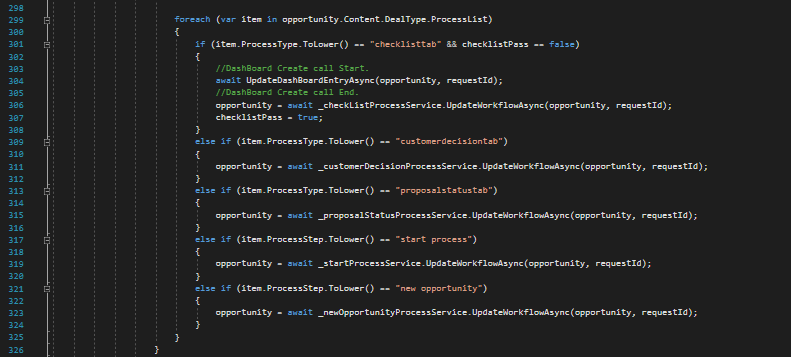
* The same logic in the previous step also needs to be implemented in the OpportunityToEntityAsync method but this type we will call the MapToEntitylAsync method:



* Next we will need to add the business logic to create and update the CustomerFeedback. This logic will be added to the OpportunityFactory which is located at (Infrastructure\Services\ OpportunityFactory.cs). to add the create logic you will need to update the CreateWorkflowAsync in the OpportunityFactory. You will simply add another if statement that calls the CreateWorkflowAsync method implemented in the service that you created:



You also need to do the same thing at the UpdateWorkflowAsync in the OpportunityFactory:

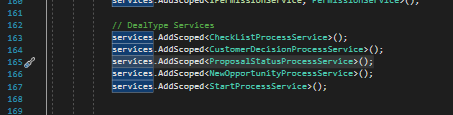


The only difference is that you will call the UpdateWorkflowAsync in the new service that you created.

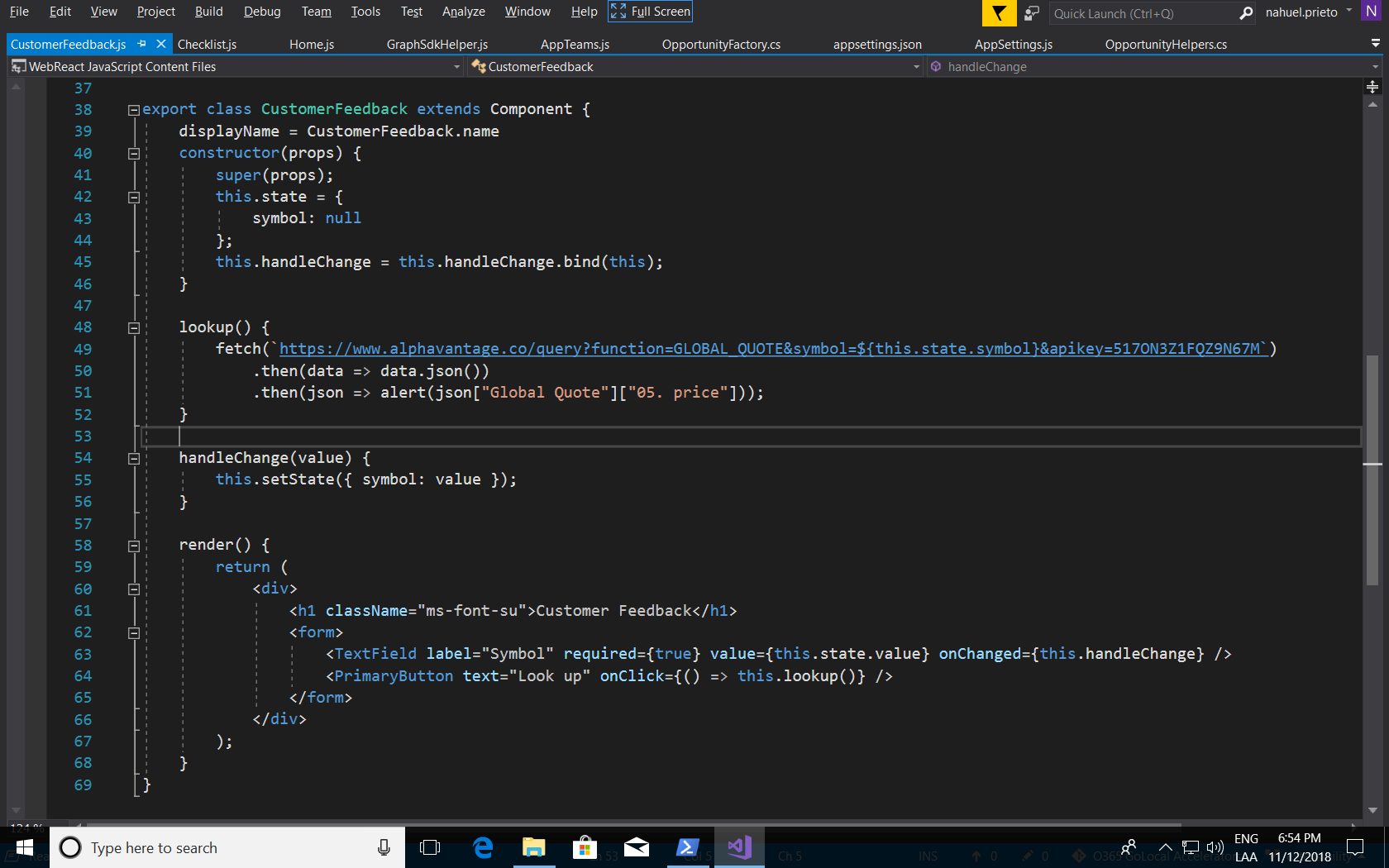
* The last thing you will need to do is to inject the new service at the start of the application. To do that you will need to add something like this to the StartUp.cs file

services.AddScoped<CustomerFeedbackService>();

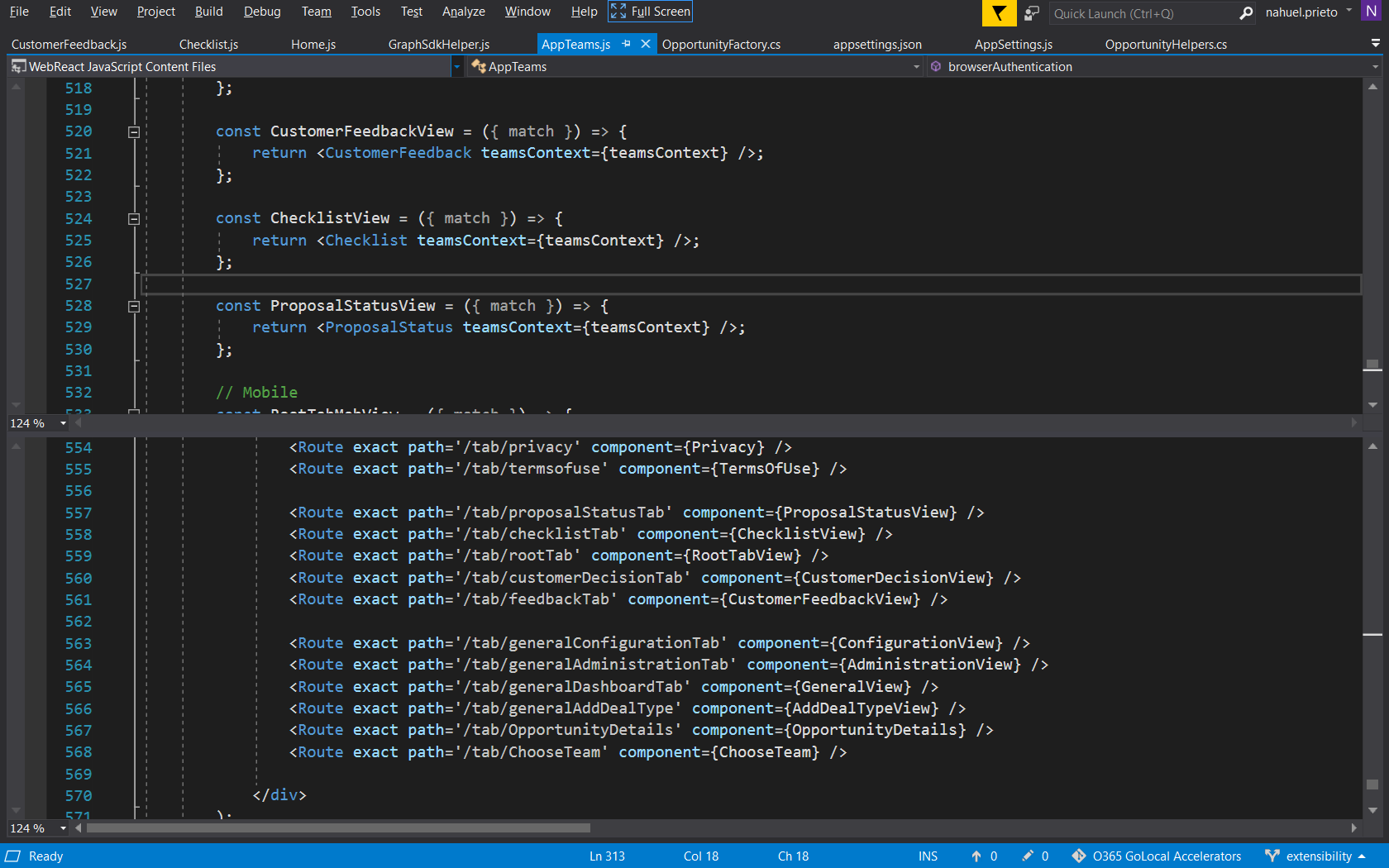
and this an example of injecting the other dealtypes that we have:



Now, in the client side, you need to add the component that will be your custom step’s tab. Add the component in WebReact\ClientApp\src\components-teams. In our case, we created a component that looks up a stock quote from the Alpha Vantage API. Yours might be totally different depending on your needs, but here is an example of the one we developed to showcase the feature:



Once you did this, you need to add the route in the AppTeams.js:



That’s it. Now, create a Deal Type and include this step. When you create an opportunity with that deal type, you will see your new component in the corresponding channel.

To see this specific sample in action, copy the contents of the “Samples\Extensibility” folder in this repo to the root of your solution and combine the contents with the existing folder.