Proposal Manager

Extensibility Guide

Version 2.0

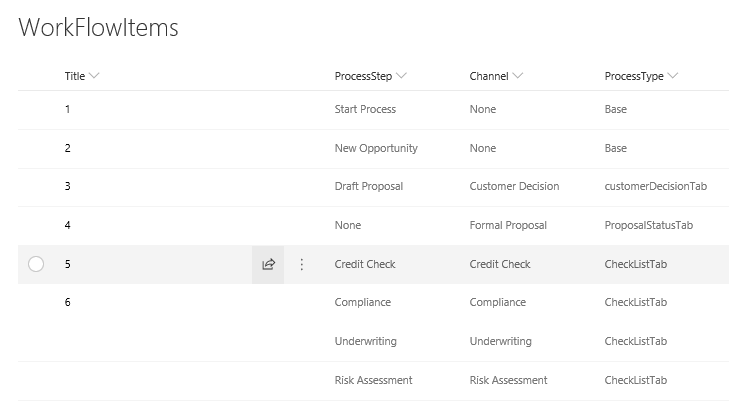
# About Extensibility

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. In this example, we will create a process named **Customer Feedback**:



Complete the following fields required for the new row. Leave empty the other fields:

1. Title: Can be anything. This will be ignored by Proposal Manager.
2. ProcessStep: The display name for the process step. For example, “Customer Feedback”.
3. Channel: The display name of the channel. Usually the same as ProcessStep.
4. ProcessType: The internal name of the tab. This will be referenced in the Proposal Manager solution. We will follow the naming convention and call it “customerFeedbackTab”.

Lastly, add the permissions needed to the **Permissions** list. The new permissions will be:

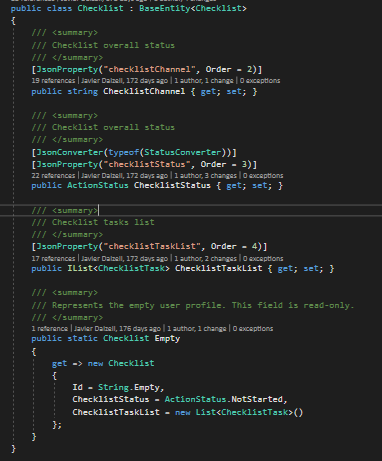
1. CustomerFeedback \_Read
2. CustomerFeedback \_ReadWrite

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

## Solution

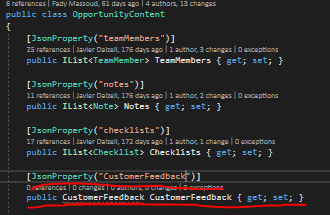
### Entity

Create a new class in (ApplicationCore\Entities) and name it *CustomerFeedback*. This class will inherit from the BaseEntity 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:



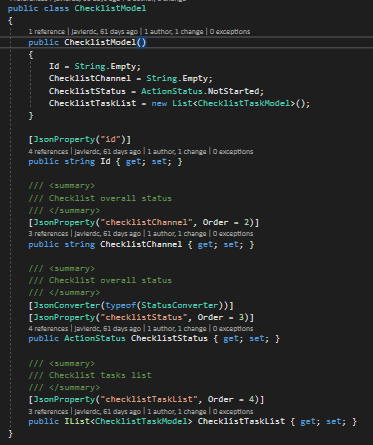
### Opportunity artifact

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 OpportunityContent class then add the CustomerFeedback there. This is an example of how it should look like:



### Model

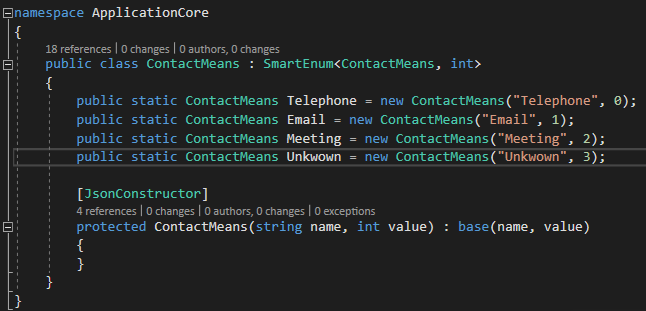
Next you would need to create a Model for the CustomerFeedback in (ApplicationCore\Models). This model should contain the data exposed that you want to expose to the client. This is an example of the model used for checklists:



This model needs to be referenced 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.

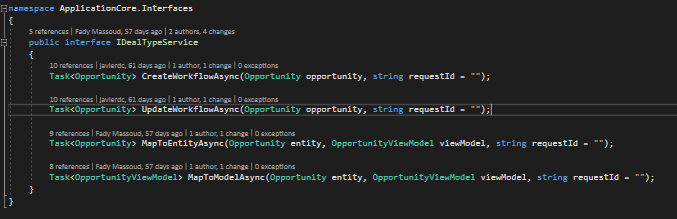
If necessary, you can create your own Dropdown-style field by writing two new classes mimicking ActionsStatus and StatusConverter. This is an example of a ContactMeans class used to store information regarding how the customer gave his/her feedback:



### Process Service

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 (You can follow the project workflow by first creating an empty interface in ApplicationCore\Interfaces. Just copy and modify accordingly the ICheckListProcessService.cs file). These are the methods that should be implemented:

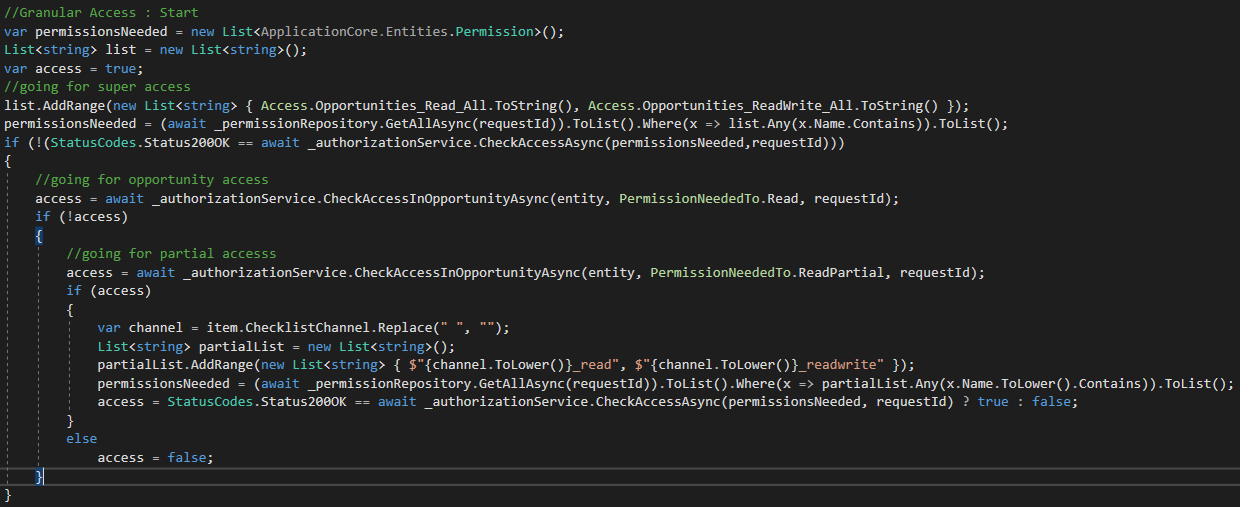


MapToEntityAsync will be used to map the Entity object created in the first step to the model object created in the third step, and MapToModelAsync will do the opposite. The CreateWorkflowAsync and UpdateWorkflowAsync methods will contain the business logic for CustomerFeedback when creating and updating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

Please note that MapToModelAsync should include the necessary permission check to verify that the current user has access to the data; either super access to the whole opportunity itself, or just access to the custom Process Step data.

### Access authorization

You can check out the other DealTypeServices for the steps involved in authorization. This is the code for the Checklist:



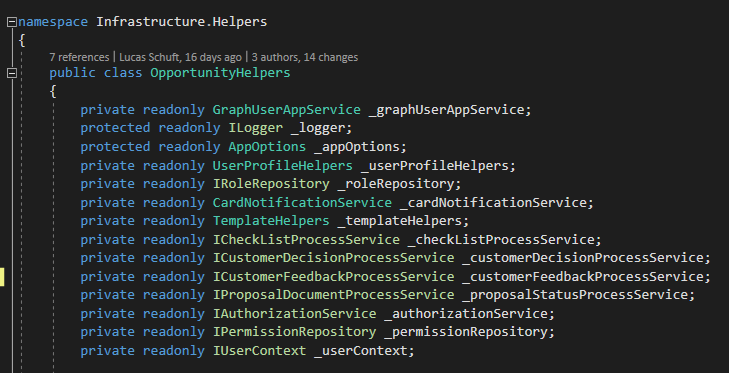
1. First it checks if the user has access to all opportunities, by checking the Opportunities\_Read/Write\_All permission.
2. If not, it checks if the user has access to the current opportunity, by checking the Opportunity\_Read/Write\_All permission.
3. If not, it checks if the user has partial access to the current opportunity. If yes, it checks if the user has access to the current channel (remember that the Checklist step can be associated by more than one channel). In our scenario, we will just check the customerFeedback\_Read and Write permissions.

If the user doesn’t have access, the ViewModel just won’t get filled with the solicited data.

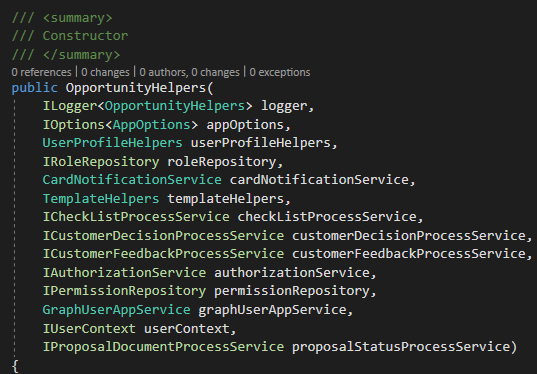
### Referencing the service

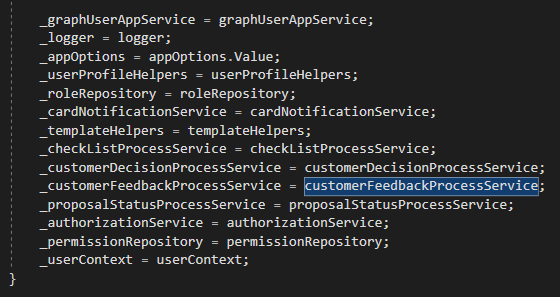
Now that we have our process service ready, we need to reference it.

The logic for mapping the opportunity ViewModel to Entity and viceversa is located at Infrastructure\Helpers\OpportunityHelpers.cs. First you need to add the ProcessService dependency by creating a new CustomerFeedbackProcessService property:

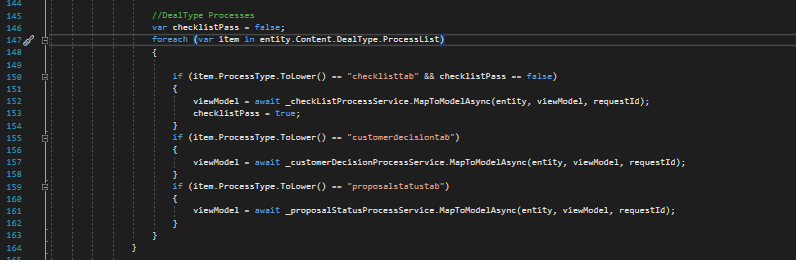


And then assign it in the Constructor:



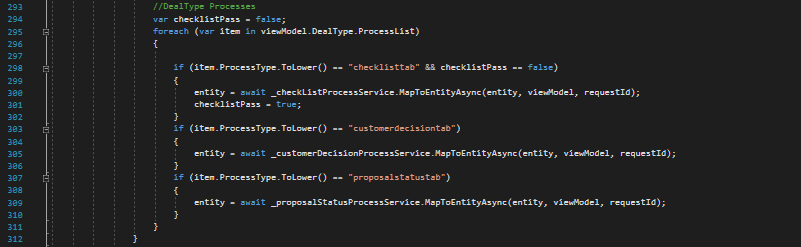


Next, you need to modify the OpportunityToViewModelAsync method in this foreach loop:



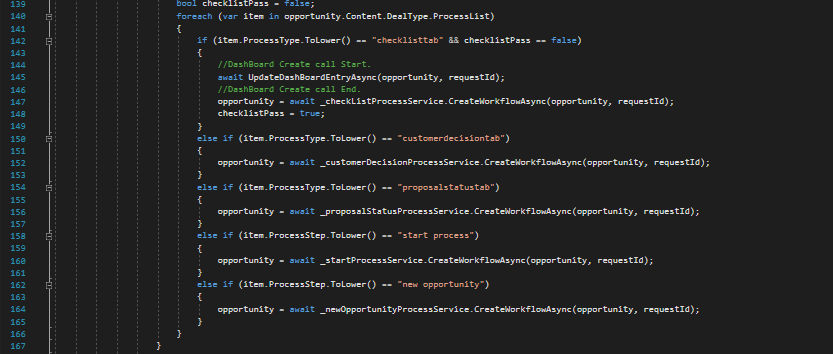
Another if statement needs to be added for our process step which calls the MapToModelAsync method created in the CustomerFeedback service. Use the internal name that you wrote in the Workflow Sharepoint list; in our case, *customerFeedbackTab*.

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:

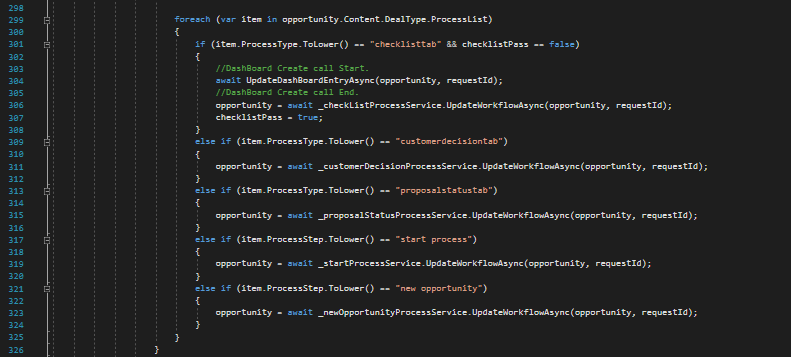


Next, we will need to add the business logic to create and update the CustomerFeedback. This logic will be added to the OpportunityFactory class, which is located at (Infrastructure\Services\ OpportunityFactory.cs). You will first need to add the ProcessService dependency, as in OpportunityHelpers.cs, by declaring the property and assigning it in the constructor.

Then, 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:



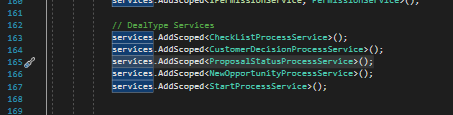
The only difference is that you will call the UpdateWorkflowAsync in the new service that you created.

### Injecting the service

The last thing you will need to do in the backend 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:

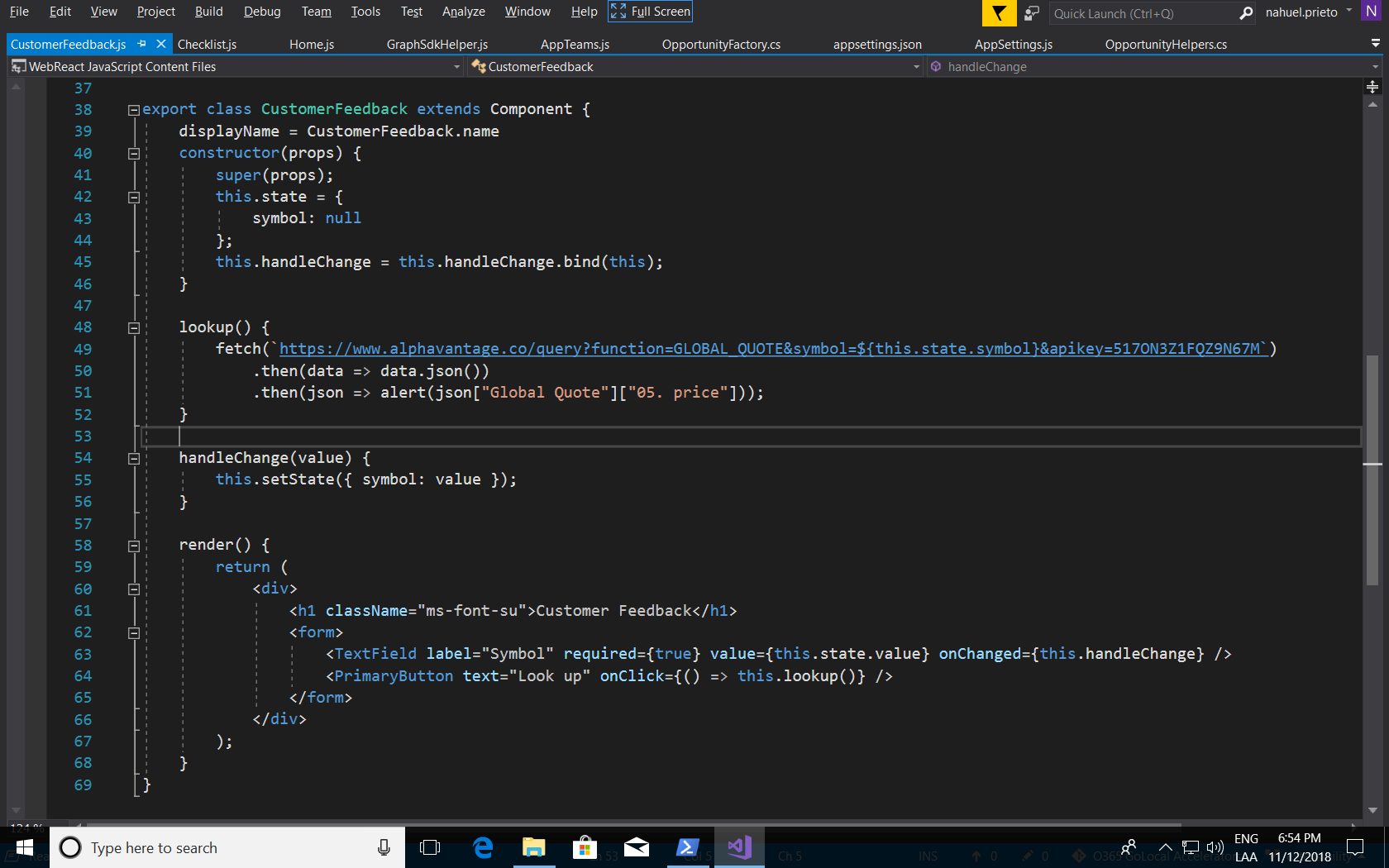
services.AddScoped<CustomerFeedbackService>();

This is an example of injecting the other DealTypes that we have:



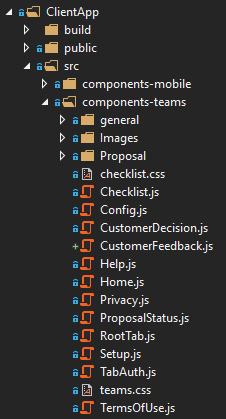
### Client-side UI

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. For the Example 1, 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:

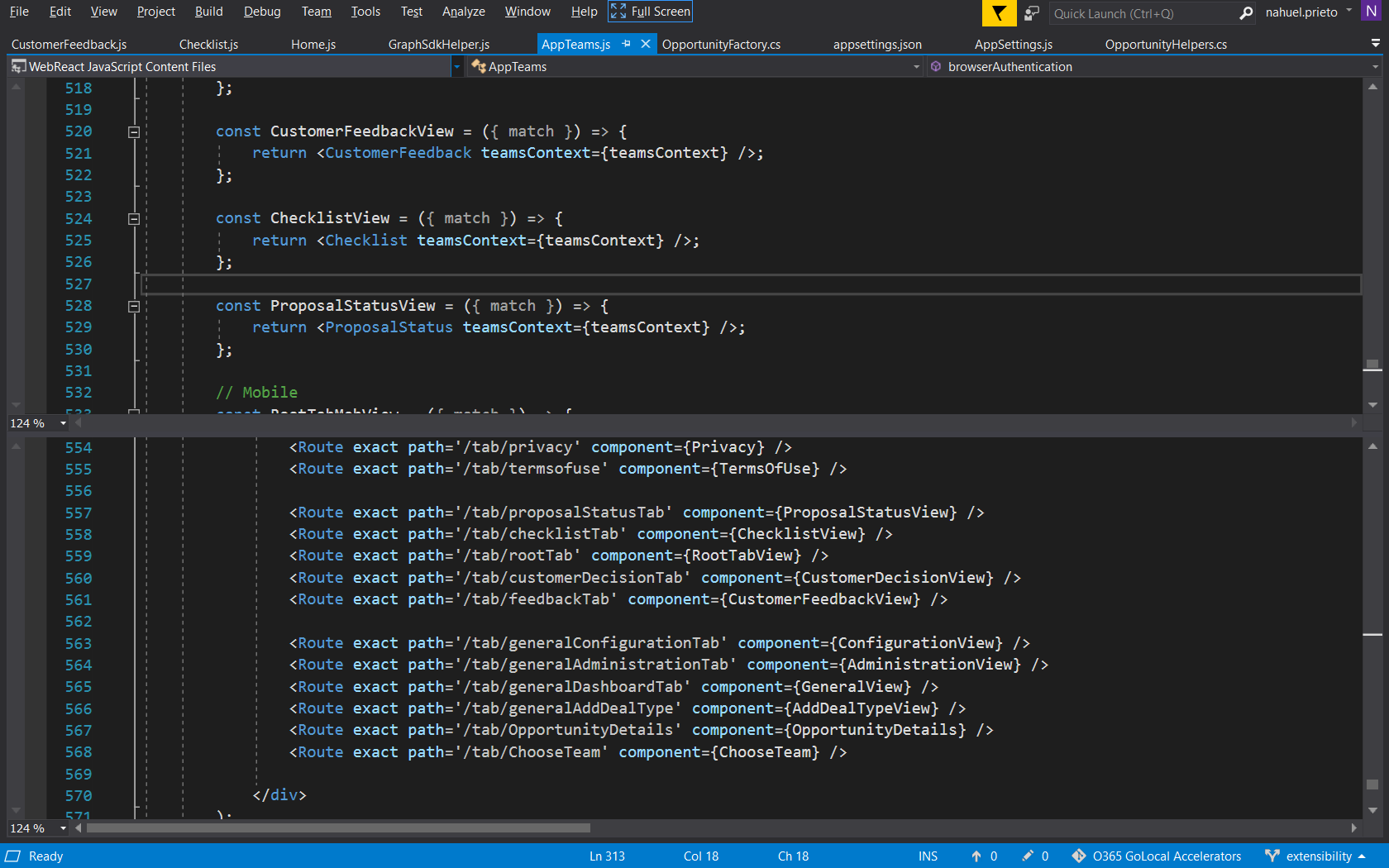


The Example 2 is based on the Checklist component. It shows a functional list where the user can record interactions with the customer that yielded feedback valuable for the team.

You can start to build your own component by checking the components for the other steps that ship in Proposal Manager:



Once you do this, you need to add the route in the AppTeams.js. Remember to add the Route element both as a mobile tab (path=’/tabmob/customerFeedbackTab’) a and desktop tab (path=’/tab/customerFeedbackTab’):



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.