

Dennis Kremenetsky

SharePoint Framework
Modern Script Editor
&
Chat Bot

Bachelor's Thesis
Bachelor of Engineering
Information Technology

2022



South-Eastern Finland
University of Applied Sciences

Author (authors)	Degree	Time
Dennis Kremenetsky	Bachelor of Engineering	May 2022
Title		61 pages
SPFx, Modern Script Editor and Chat bot		
Commissioned by		
Marskidata Oy		
Supervisor		
Reijo Vuohelainen		
Abstract		
<p>The objective of this thesis was to create a “How to” guide on using SharePoint framework, Modern Script Editor, SharePoint online modern, and creating and implementing a chat bot on a SharePoint site. The process required practically no coding to accomplish the end-product. It also covered extra features including QnA analytics, chat widgets and Omnichannel.</p> <p>SharePoint Online is used for building complex Intranets where information can be shared in the organization. SharePoint framework is needed to build web parts and extensions for SharePoint sites. The Modern Script Editor is needed to be able to add custom code into a SharePoint site. Using services like Azure QnA, Power Virtual Agent and Power Automate a chat bot was created with a knowledge base. Omnichannel can be used to extend the capabilities of Power Virtual Agent.</p> <p>With the combination of SharePoint Framework, Modern Script Editor and Microsoft applications, an Intranet was created with the capability of supporting a chat bot and live agent transfer. Omnichannel was able to provide live customer support straight from the chat bot. This thesis showed Marskidata how to build a chat bot for their own site or customers Intranet.</p>		
Keywords		
SharePoint Framework, Modern Script Editor, SharePoint online modern, Chat bot.		

CONTENTS

1	INTRODUCTION	5
2	THEORY.....	5
2.1	NVM and NodeJS	6
2.2	Gulp, Yeamon and SharePoint generator	6
2.3	SharePoint Framework and Modern Script Editor.....	7
2.4	Azure QnA and Power Virtual Agent.....	8
2.5	Omnichannel.....	9
2.6	Chat Widgets	9
2.7	Azure QnA Analytics.....	10
3	IMPLEMENTAION OF THE PRODUCT	10
3.1	SharePoint Framework	10
3.1.1	SharePoint Framework Setup	11
3.1.2	NVM	11
3.1.3	Gulp, SharePoint Generator and Yeamon (yo)	12
3.1.4	Visual Studio Code.....	14
3.1.5	Microsoft SharePoint Generator.....	15
3.1.6	Hello World web part.....	17
3.1.7	Conclusion to SharePoint Framework.....	19
3.2	Modern Script Editor	19
3.2.1	GitHub Repository	20
3.2.2	7Zip	20
3.2.3	React Script Editor Directory	21
3.2.4	SharePoint Management Command Shell	21
3.2.5	Modern Script Editor App	22
3.2.6	Modern Script Editor Deployment	24
3.2.7	SharePoint Admin Center.....	26
3.2.8	SharePoint Site	29

3.2.9	Conclusion to Modern Script Editor	31
3.3	Chat Bot.....	31
3.3.1	QnA Knowledge Base	31
3.3.2	Power Virtual Agent and Power Automate	35
3.3.3	SharePoint Online	43
3.3.4	Chat Widget	44
3.3.5	Omnichannel Live Chat.....	51
3.3.6	QnA Analytics.....	55
3.3.7	Conclusion to chat bot.....	56
4	CONCLUSION.....	57
5	REFERENCES	58
6	LIST OF FIGURES	60

1 INTRODUCTION

The objective of this thesis is to create a “How to” guide on using the SharePoint framework, Modern Script Editor and a chat bot. Marskidata Oy is an IT firm that provides many different types of IT solutions for companies. It offers services like printing, IT customer support, Microsoft solutions and SharePoint intranets. Marskidata commissioned me to create a FAQ chat bot for SharePoint online modern. The goal is to be able to use a chat bot widget in a SharePoint online modern site. I will be using NVM, NodeJS, NPM, Yeaman, Microsoft SharePoint generator, Gulp, SharePoint framework, Power Virtual Agent, Power Automate and Omnichannel. With these software and services, I hope to accomplish an acceptable solution.

These days a chat bot is practically on every website. The chat bot can provide answers to common questions that would have usually involved a call to customer support or time spent looking through the website for the correct information. The chat bot can also be used to transfer a customer to a live agent. The customer would be able to ask the agent in real time the questions that they have. The chat bot can be used as a FAQ bot, or it can be used for IT customer support.

To have the chat bot displayed in a web part only in a SharePoint online modern page, skip to chapter 3.3, which is called **Chat bot**. If the chat bot needs to be displayed as a widget, the information is provided in the section called **Implementation of the Product** and needs to be followed fully.

2 THEORY

This thesis uses several different softwares and services. All these softwares and services are needed for the final solution to work. Some of these are free and some require a license. During the process of following this “How to” guide, the trial versions can be used to accomplish everything that is needed. The main software that are needed are Azure QnA, Power Virtual Agent and Power Automate. Other software that will be used is NVM, NodeJS, Gulp, Yeaman, SharePoint generator, Modern Script Editor and Omnichannel.

2.1 NVM and NodeJS

The first software that is needed is NVM. NVM is a version manager for NodeJS. Kumar explains, "NVM (Node Version Manager) is the command line utility for installing Node.js on your system. It allows us to install multiple Node.js versions and switching between them. This is helpful for the system running multiple Node application that required different-2 node versions."(Kumar, 2022, pp. 1). The applications that will be created and built will require different versions of NodeJS and not the most recent version.

Another software that is needed is NodeJS and it automatically comes with NPM. NPM is a packet manager that will install javascript packages for applications. Wikipedia says:

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts (Wikipedia, 2020, pp. 1).

To start building an application, NPM will run scripts to install required javascript packages.

2.2 Gulp, Yeoman and SharePoint generator

The next software's are Gulp, Yeoman and the Microsoft SharePoint framework generator. Juvonen explains that "Gulp is a JavaScript-based task runner used to automate repetitive tasks. The SharePoint Framework build toolchain uses Gulp tasks to build projects, create JavaScript bundles, and the resulting packages used to deploy solutions." (Juvonen, 2022a, pp. 9).

Yeoman and Microsoft SharePoint generator are needed to run the SharePoint framework, and to create web parts and extensions for SharePoint. Yeoman is an open source scaffolding tool for applications. It is used for web applications and it will be used to launch the Microsoft SharePoint generator. The SharePoint generator is used to provide templates for the SharePoint framework, from which can be built web parts and extensions. Once the SharePoint framework is created, Gulp will be used to test and deploy the application. Gulp will be used to test the application in a local environment and to package the application into a ready solution. NPM and Gulp will be used to build the application into a ready solution that can be uploaded to SharePoint online. NPM and Gulp will also be used to build the Modern Script Editor into a solution that can be used on a SharePoint online site. The Modern Script Editor is built on a SharePoint framework and is an application that was built by Mikael Svenson. Modern Script Editor will be used in a SharePoint online modern site to allow custom code implementation.

2.3 SharePoint Framework and Modern Script Editor

To build web parts or extensions for SharePoint Online, the SharePoint framework is needed. "The Yeoman SharePoint web part generator helps you quickly create a SharePoint client-side solution project with the right toolchain and project structure." (Juvonen, 2022a, pp. 11). The SharePoint framework is an easy-to-use building tool for web parts and extensions. Questions and steps are provided to build the solution, when the generator is launched through Yeoman.

There is a large open-source repository on GitHub of many kinds of web parts for SharePoint. In the samples section of the repository, the react-script-editor folder can be found. It is also known as the Modern Script Editor, and it will be used for custom code insertion for SharePoint Online modern sites.

Coming from old classic SharePoint pages you might have existing script solutions you want to re-use on a modern page without having to repackage it as a new SharePoint Framework web part. This web part is similar to the classic Script Editor Web

Part, and allows you to drop arbitrary script or HTML on a modern page (Svenson, 2021, pp. 1).

Without the Modern Script Editor, SharePoint Online Modern does not have the ability to use custom code.

There may be complications with custom web parts showing up in SharePoint Online. A solution for the complications can be found on YouTube, which is provided by HelpingAll. There is one important part that needs attention, at the end of the video. A key feature to have the custom web part to show up is the command `-DenyAddAndCustomizePages 0` (HelpingAll, 2019).

2.4 Azure QnA and Power Virtual Agent

Microsoft offers many different types of services and softwares. This guide will use Azure, Power Virtual Agent, Power Automate and Omnichannel. These services are offered by Microsoft and are required for the final solution to work.

Azure is a cloud based computing service and offers many different types of cloud based services. In Azure a resource group is needed, which will contain the QnA knowledge base. Azure also offers analytics and app registration for the services that are created. It is an easy interface to use and does not require a lot of time to create a large knowledge base. "You can create a QnA Maker knowledge base (KB) from your own content, such as FAQs or product manuals. This article includes an example of creating a QnA Maker knowledge base from a simple FAQ webpage, to answer questions." (mrbullwinkle, 2022a, pp. 2).

Power Virtual Agent is a no-code chat bot solution. It is a bot framework that can be easily used and set-up for customers and employees. Power Automate is used to create workflows between different applications and services. One will use this to create a connection between the QnA knowledge base and Power Virtual Agent. "Power Virtual Agents empowers teams to quickly and easily create powerful bots using a guided no-code graphical experience - all without the need for data scientists or developers." (D'Souza-Wiltshire, 2022,

pp. 1). As this thesis has been commissioned by a Finnish company, it relies more heavily on the QnA knowledge base than on the Power Virtual Agent. The knowledge base supports the Finnish language, but the Power Virtual Agent does not.

2.5 Omnichannel

A great feature to have associated with the chat bot is the transfer to agent option in Power Virtual Agent. Customers who use the chat bot will be able to transfer from the chat bot to a live agent in Omnichannel customer service. Omnichannel provides messaging, voice and recording services to the chat bot. "Omnichannel for Customer Service offers a suite of capabilities that extend the power of Dynamics 365 Customer Service Enterprise to enable organizations to instantly connect and engage with their customers across digital messaging channels." (Nellimarla, 2022, pp. 1). Once the connections are made with Omnichannel, a chat bot widget will be available for SharePoint online. One of the main reasons to have the Modern Script Editor installed is to be able to use the code snippet from Omnichannel in the SharePoint online site.

2.6 Chat Widgets

There are several options for chat bot widgets that can be used in SharePoint online. Open-source solutions can be found on GitHub and are usually provided in an almost ready state. There are many repositories that can be downloaded from GitHub that are useful for SharePoint. One repository is provided by Renato Romão and can be downloaded from GitHub (Romão, 2020). The chat widget can be implemented on a SharePoint site. It is an application that needs to be built using NPM and Gulp. After the solution is built, an application is provided that can be uploaded to SharePoint online. It uses the Power Virtual Agent credentials to connect the chat bot to the chat widget.

Another solution for a chat bot widget is created by Stephan Bisser. The web part is called react-application-botframework-chat and can be installed to SharePoint online. The repository can be downloaded from GitHub and built

using NPM and Gulp (Bisser, 2019). This also uses the Power Virtual Agent credentials to connect the chat bot with the chat widget.

The last solution for a chat bot widget can also be found in the same repository as the chat bot widget by Stephan Bisser. This last solution is called react-application-pva-bot (Bernier, 2021). The same as the previous chat bot widgets, it uses the Power Virtual Agent credentials.

2.7 Azure QnA Analytics

An important service and feature that needs to be mentioned is Azure QnA Analytics. Using the analytics service, a record is created with the questions and answers that are entered into the chat bot. This record can later be viewed and exported into an excel file using the Logs option in the service. “QnA Maker stores all chat logs and other telemetry, if you have enabled Application Insights during the creation of your QnA Maker service. Run the sample queries to get your chat logs from Application Insights.” (mrbullwinkle, 2022c, pp. 1).

3 IMPLEMENTAION OF THE PRODUCT

This section of the thesis will cover the practical implementation of the software and services mentioned in the theoretical section. Step by Step instructions will be provided to have a functional chat bot. One will use SharePoint framework, Modern Script Editor, Azure QnA, Power Virtual Agent, Power Automate and Omnichannel.

This guide is a current working version of SPFx and Modern Script Editor as of 1st March 2022. All specified versions of software must be installed, and the steps followed exactly.

3.1 SharePoint Framework

Chapter 3.1 will cover the SharePoint Framework for SharePoint online. More commonly known as SPFx. There are two different versions of SharePoint online, classic and modern. Classic has more capabilities when it comes to

editing and entering code directly into SharePoint. Most companies have now switched to the modern version of SharePoint online. SharePoint online Modern is more limited to the things that can be done, for example, one cannot directly enter custom code into a SharePoint online modern page. This guide will provide instructions on how to install all the necessary software to be able to use the SharePoint Framework, create one's own web part and install it into a SharePoint online page.

After the installation of NodeJS with NVM, one must install exactly the versions of software that are specified. There are a lot of compatibility issues between the software's that are being installed. The SharePoint framework that is needed, supports only versions 8-10 of NodeJS. Installations of Gulp, Yo, NodeJS and SharePoint Generator also need to be the exact versions specified for applications to build correctly. There were problems with one software being outdated or the latest not working with another software. For example: newest version of Yo does not work with Gulp, certain versions of Yo and Gulp do not work with the latest version of SharePoint generator. It required a lot of testing to find versions that are compatible with each other, needed for the full project to work.

3.1.1 SharePoint Framework Setup

First, start with the SharePoint framework, or more commonly known as SPFx. Using this framework, one will be able to create an application and install the application into a SharePoint online site. The SharePoint framework uses NodeJS, NPM, Yeoman(yo) and SharePoint generator. One can install NodeJS and NPM using NVM. Using NVM, the version of NodeJS can be changed to what is needed at the time.

3.1.2 NVM

Using NVM, one can install NodeJS and NPM. NVM will be used to change between different versions of NodeJS. Navigate to the tutorial provided by Corey Butler to download NVM, a link is provided in the Reference section (Butler, 2022). NVM is necessary because the different applications that are needed to build the final solution require different versions of NodeJS. After

NVM is downloaded, one can check the installed version by typing *nvm* and show a list of installations by typing *nvm list* in the command prompt. To change between different versions of NodeJS, run *nvm install 10.19.0* and then *nvm use 10.19.0* to install and use the different versions. If a different version of NodeJS is needed, then just replace the version numbers.

```
C:\Windows\system32>npm list -g --depth=0
C:\Users\dennis.kremenetsky\AppData\Roaming\npm
-- (empty)
```

Figure 1. NPM command to list all global installations on the system

Before the other installations, one can check what global software are already installed on the system. Figure 1 shows the command that needs to be run in the command prompt. Open the command prompt in administrator mode and run the command *npm list -g --depth=0* (Juvonen, 2022a).

3.1.3 Gulp, SharePoint Generator and Yeamon (yo)

There are several software's that are needed to run the SharePoint Framework. First, install Gulp, which is used to create and build the applications for testing and for deployment. Then install Yeamon(yo) which is required to launch and run different generators. Last, will be installed the SharePoint generator, which will be the framework for creating new applications for SharePoint.

Open the command prompt in administrator mode and run the command *npm install -g gulp@4.0.2* to install gulp. If version 4.0.2 does not work, then install version 3.9.1 of gulp.

Next, install the Microsoft SharePoint Generator version 1.11.0. Use the command *npm install -g @microsoft/generator-sharepoint@1.11.0* to install the generator.

To launch the SharePoint generator, one will need to install Yeamon. Install Yeamon with the command *npm install -g yo@3.1.0*. When it comes to

Yeamon, there can be complications, which will be covered in chapter 3.1.3.1. The installation will also say that the version of Yo is outdated, ignore it.

```
C:\Windows\system32>npm list -g --depth=0
C:\Users\dennis.kremenetsky\AppData\Roaming\npm
+-- @microsoft/generator-sharepoint@1.11.0
+-- gulp@4.0.2
+-- node@10.19.0
+-- yo@3.1.0
```

Figure 2. Command to check global installations in the system

Figure 2 shows what global installations of software that are installed on the system. The command in Figure 1 can be entered, and it will show what software's are installed globally and the versions.

3.1.3.1 Yo Complications

The only latest version of software that is being used in this project is NPM, all the other software are not the latest versions. This can cause compatibility issues between the different software and for the software itself. One problem that one may come across is complications with Yeamon. There can be pathing issues with Yeamon, it could not find the correct path to be able to be installed correctly. There can be several different ways to fix this problem and provided in Figure 3 is a solution that works.

```
C:\Windows\system32>npm config get prefix
C:\Users\dennis.kremenetsky\AppData\Roaming\npm
```

Figure 3. Command to show the path of where npm modules are installed

Figure 3 shows the path of the where the software modules are installed with NPM. Using the command `npm config get prefix` will show the path of where the npm modules are installed. Open the Windows Search bar and type in *Environment Variable* in the search bar, choose to edit the system environment variables. Click on Environment Variables button and a new window will appear with some options. Under the tab System Variables, click on *Path* and click *Edit*.

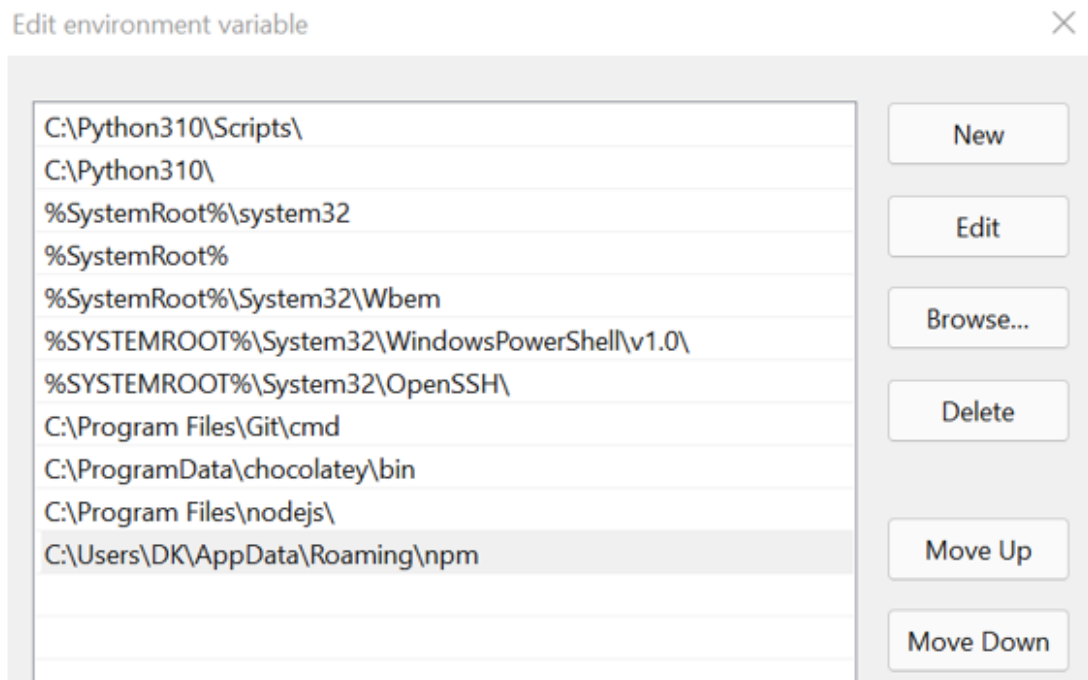


Figure 4. Edit environment variable

Figure 4 shows a window named Edit environment variable. In the edit environment variable tab click on *New* and add the path that was provided by the command in Figure 3. After the new path has been added, click ok, click ok again and then apply in the System Properties window (wjohnsto, 2015).

Enter the command `npm install -g yo@3.1.0` in the command prompt. This time it should install with no problems. If an error is received saying that Yo is outdated, ignore it.

3.1.4 Visual Studio Code

Visual studio is needed to edit code and files. It can be found by searching for it in google. Visual Studio Code is used in creating and editing a basic application with the Microsoft SharePoint Generator.

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and

install extensions that add additional functionality (Wikipedia, 2019, pp. 1).

Other software that can be used to edit code and files is Notepad++. One can install Notepad++ and use it instead of Visual Studio Code.

3.1.5 Microsoft SharePoint Generator

Now one will use the Microsoft SharePoint Generator to create a SharePoint framework for a web part. One can later use this as a web part on the SharePoint online modern page. First create a folder somewhere on your system for this application. Create a folder where the application will be and then open the command prompt in Administrator mode.

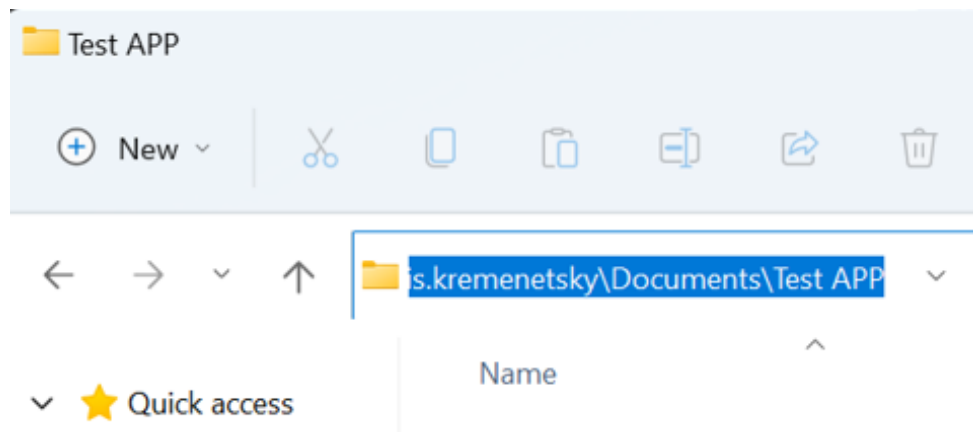


Figure 5. Folder path for the application

As shown in Figure 5, where one created the folder for the application, one can click on the address window, and it will give the exact path of where the folder is. Copy the path so it can be pasted it into Command Prompt. Use the `cd` command to move to the folder where the web part application will be built. Run the command `yo @microsoft/sharepoint` in the command prompt to start the creation process of the web application. If everything is installed correctly in the previous steps, there will be no errors and an interface will appear with the following options.

```
Let's create a new SharePoint solution.
? What is your solution name? test-app
? Which baseline packages do you want to target for your component(s)? SharePoint Online only (latest)
? Where do you want to place the files? Use the current folder
Found npm version 6.13.4
? Do you want to allow the tenant admin the choice of being able to deploy the solution to all sites immediately
running any feature deployment or adding apps in sites? No
? Will the components in the solution require permissions to access web APIs that are unique and not shared with
components in the tenant? No
? Which type of client-side component to create? WebPart
Add new Web part to solution test-app.
? What is your Web part name? HelloWorld
? What is your Web part description? HelloWorld description
? Which framework would you like to use? No JavaScript framework
```

Figure 6. SharePoint solution options example

Figure 6 shows the process and the steps that are chosen to build a web part application using the SharePoint generator. In the interface, answer the questions that are asked. Write the name of the solution for the web part application. Next choose the framework support for the web part. As this is being built specifically for SharePoint online, choose the SharePoint Online only (latest) option.

Choose where the solution should be installed. One can choose the current folder or choose a different location where the application will be created. Decide whether the application will be available to everyone in the tenant automatically or that everyone will upload the application to the SharePoint site individually. The default option is no, and everyone will need to install the application individually.

Select the default options and proceed through the building process. One can now choose what type of application it will be. For this example, choose the WebPart option. Choose the name of the web part. A description can be written if needed.

Make the choice of what type of framework one wants for the web part. After this option, the creation process will begin. In the folder that was provided earlier; the generator will start creating a framework for the web part.


```

_+#####!
#####!
###/  (## (@)
### ##### \
###/  /### (@)
##### ## /
###  /## (@)
#####!
*+=+#####!

```

Congratulations!
 Solution `web-part` is created.
 Run `gulp serve` to play with it!

Figure 7. Successful creation of the application

After all the options are chosen, the web part application will start the building process. Figure 7 shows when the installation is finished, a Congratulations message verifying that the application has been built successfully. The process has successfully created the framework for a web part application and can be tested with `gulp serve`.

3.1.6 Hello World web part

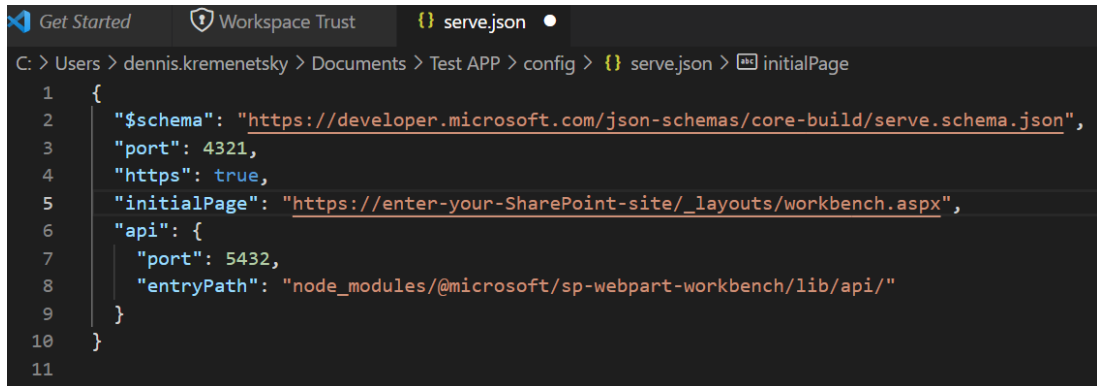
At this point, the framework building process of the solution is complete and Visual Studio Code can be used to edit the framework. Browse to the files in the folder where the solution is. One can now test the web part that was created.

Name	Date modified	Type
config	09/02/2022 10.46	JSON Source File
copy-assets	09/02/2022 10.46	JSON Source File
deploy-azure-storage	09/02/2022 10.46	JSON Source File
package-solution	09/02/2022 10.46	JSON Source File
serve	09/02/2022 10.46	JSON Source File
write-manifests	09/02/2022 10.46	JSON Source File

Figure 8. The `serve.json` file can be found in the config folder

There are two ways of setting up the testing environment. One can edit the `serve.json` file, if there is already a SharePoint site that can be used. If there is not site that can be used, one can launch the testing environment with the `gulp serve --nobrowser` command in the command prompt. Figure 8 shows the

location of the `serve.json` file. If a SharePoint site is available, locate the `./config/serve.json` file in the project. Use Visual Studio Code to edit the content of the file. Right click on the file and open the file using Visual Studio. To test the web part that has been created, one needs to change the `initialPage` option to the SharePoint online site.



```

C: > Users > dennis.kremenetsky > Documents > Test APP > config > serve.json > initialPage
1  {
2    "$schema": "https://developer.microsoft.com/json-schemas/core-build/serve.schema.json",
3    "port": 4321,
4    "https": true,
5    "initialPage": "https://enter-your-SharePoint-site/_layouts/workbench.aspx",
6    "api": {
7      "port": 5432,
8      "entryPath": "node_modules/@microsoft/sp-webpart-workbench/lib/api/"
9    }
10 }
11

```

Figure 9. InitialPage example

Figure 9 shows the contents of the `serve.json` file. Change the portion that says *enter-your-SharePoint-site* to your own SharePoint Site. Save the file and switch over to the Command Prompt. Make sure it is launched in Administrator mode and move to the location of the application folder.

Use the command `gulp trust-dev-cert` to give a trusted status to the SharePoint Framework. Let the process run its course and a window will appear that asks for verification, whether to install the certificate or not. Use the command `gulp serve` to start a test server to test the web part. A new browser window will appear where the web part can be tested.

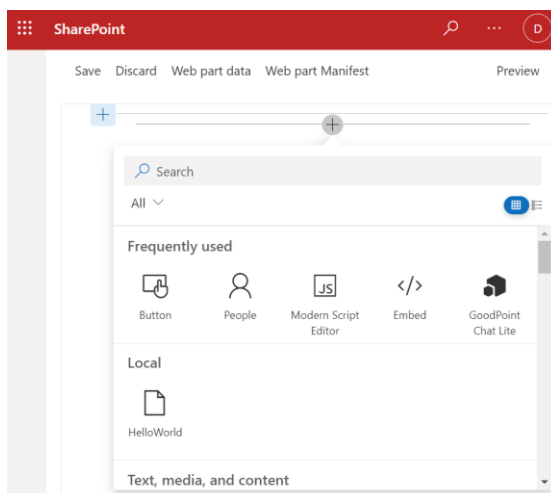


Figure 10. SharePoint test environment

Figure 10 shows how the test environment looks. Click on the plus symbol in the test environment to see if the web part shows up. One will be able to see the web part application load into the web part window.

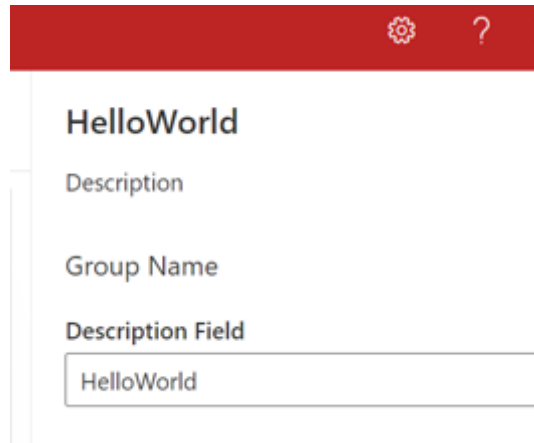


Figure 11. The web part description can be edited

Figure 11 shows the field where one can edit the web part. Edit the web part in different ways with the edit web part option. As this is just a basic web part that was created, one can only edit the Description Field (Juvonen, 2022b).

3.1.7 Conclusion to SharePoint Framework

This has been an example of how someone can create a web part for a SharePoint Online Modern site using the SharePoint Framework. There are many different types of guides and solutions online that can now be tested and experimented with. The framework has been created and all necessary software is installed. Chapter 3.2 shows how to create and deploy one of the most important web parts for a SharePoint online modern page.

3.2 Modern Script Editor

The Modern Script Editor is one of the most important web parts one can have for SharePoint online modern. With this web part, HTML, CSS and JavaScript code can be input into the web part and site directly. The modern version of SharePoint online does not have this capability without this web part. This greatly increases what one can do for a SharePoint online modern site. This part of the guide will show step-by-step on how to implement this solution. It is important that all the software one has installed are exactly the versions that

are specified. There are many version conflicts which each software, and each software must be the correct version for the final product to work. If one has followed the steps up until this point, then there should not be any problems making this work.

3.2.1 GitHub Repository

The web part GitHub repository has many web part solutions that have been built by other people and are open source. Navigate to the GitHub repository which is called sp-dev-fx-webparts (Bernier, 2021).

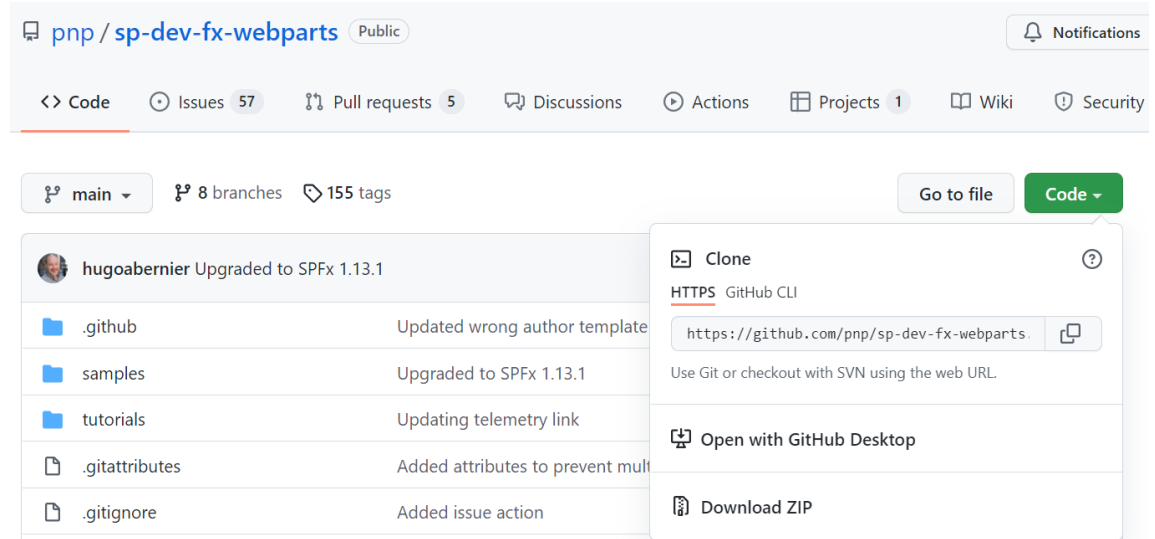


Figure 12. Code option to download as Zip file

Figure 12 shows where in GitHub one can download the Zip file. The zip file needs to be downloaded, because when the repository is cloned, it does not clone all the sample applications correctly.

3.2.2 7Zip

Using the built-in zip extractor in windows does not work correctly. There are errors saying that the path of some files is too long. The react-script-editor that is needed does not extract because of that. Search for 7zip in google and download it. Download and install the version that supports the local operating system. Extract the zip file that has been downloaded from the GitHub site using 7Zip and there will not be any errors or missing files.

3.2.3 React Script Editor Directory

Now that the Zip file is unpacked, explore what is in the folder. This directory offers many different web parts and extensions for SharePoint online. For this project, one needs to find the react-script-editor folder. Find the samples folder and open it. In the samples folder, find the react-script-editor folder. Open the react-script-editor folder and there should be similar files in it as one had in the web application that was built in chapter 3.1.6 (Svenson, 2021).

3.2.4 SharePoint Management Command Shell

This part is very important to make the web part show up in a SharePoint site. Even if the application is installed and on a SharePoint site, the web part will not show up. If one does not have a SharePoint site yet, this can be done later.

The SharePoint Management Command Shell needs to be downloaded. The command shell will be used to configure a SharePoint online site. Search for SharePoint management command shell in google and download and install it. After the SharePoint online management shell has been installed, open the windows search bar and look for it. Open it in administrator mode. One will need to do these steps for the custom applications to show up in the web parts section of a SharePoint site.

```
connect-sposervice https://mddemo365-admin.sharepoint.com/
```

Figure 13. The command will connect to a SharePoint account

Figure 13 shows the command that needs to be run to connect to a SharePoint Online admin account. The command *connect-sposervice* lets one connect to a SharePoint account through the command shell. A window will appear when one runs the command, type in the credentials and a connection will be formed to the SharePoint site.

```
Set-Sposite -Identity https://mddemo365.sharepoint.com/sites/Dennis  
-DenyAddAndCustomizePages 0
```

Figure 14. The command to change the restrictions on a SharePoint site

Figure 14 shows the command that needs to be run to enable custom content for SharePoint online. The command *Set-Sposite -Identity* sets the site one wants make modifications for. Provide the identity of the site one wants to modify permissions for and add the command *-DenyAddAndCutomizePages 0* after the identity that is provided. (HelpingAll, 2019). This will now allow custom applications to be seen that are added to a SharePoint online site. Remember, that without this step, custom applications will not show up in a SharePoint online site.

3.2.5 Modern Script Editor App

In chapter 3.2.4 one enabled custom content for SharePoint online. Navigate back to the react-script-editor folder and start the building process. There will be several steps, and they need to be done in the correct order or the application will not work as needed. Open Command Prompt in administrator mode and navigate to the folder where the react-script-editor folder is located. Use the *cd* command to navigate to the correct folder in the command prompt.

```
react-script-editor>npm install
```

Figure 15. The npm install command will start the building process

Figure 15 shows the command that needs to be run to start the building process. Navigate to the react-script-editor folder in the repository, use the command *npm install*. The process can take several minutes. At the end of the installation process, it will show how many seconds it took to complete the operation and a mention if there are any vulnerabilities. Ignore the vulnerabilities and continue with the process.

```
react-script-editor>gulp serve
```

Figure 16. The gulp serve command will start a local server

After the installation is finished, one can test the application locally to make sure it works. Figure 16 shows the command that needs to be run to start the local server. The command *gulp serve* starts a local server where it runs a

SharePoint test environment. A browser window will appear with a test environment server of SharePoint online.

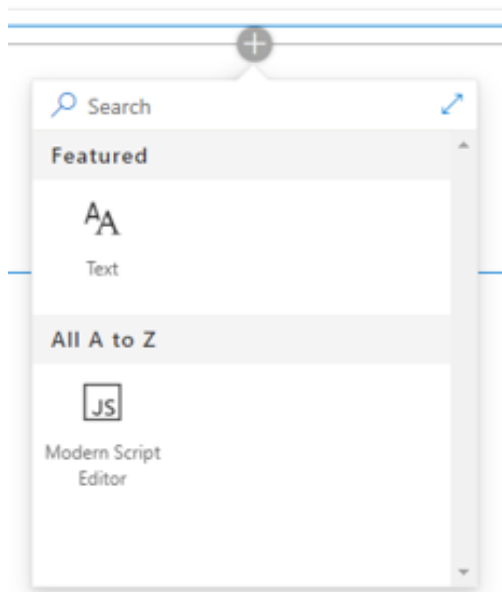


Figure 17. The application shows up in the web part section

Figure 17 shows that the application shows up in the test environment. Click on the plus and see if the Modern Script Editor shows up. At this point even without the SharePoint Management shell commands, one will see if the web part build was successful in the test environment.

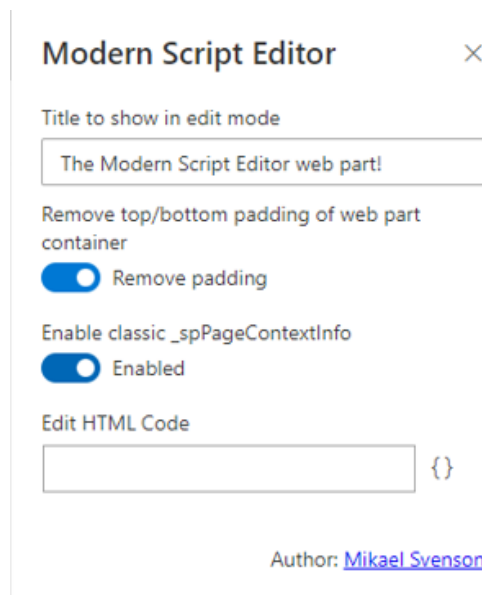


Figure 18. The modern script editor with the fields that can be edited

Figure 18 shows how one can edit the Modern Script Editor. Click on the Edit HTML Code section and a window will appear where one can enter an HTML code. Enter an html code and click save.



Figure 19. Random HTML code working in SharePoint online modern web part

Figure 19 shows that the custom code works in the web part using the Modern Script Editor. Press the preview button at the top right corner of the browser to show the working web part. In the command prompt where the server running, press *CTRL+C* in the command prompt to stop the test environment server from running (Svenson, 2021).

3.2.6 Modern Script Editor Deployment

One can now start the process of building the web part into an application that can be added to SharePoint online. Depending on when the edits are made, they can be overwritten by the build process. Open Command Prompt as administrator and navigate to react-script-editor folder with the *cd* command. Run the command *gulp clean*. Next run the command *gulp bundle --ship* to build the solution for distribution.


```

[13:41:58] =====[ Finished ]=====
[13:41:59] Project pnp-script-editor version:1.0.17
[13:41:59] Build tools version:3.12.1
[13:41:59] Node version:v10.19.0
[13:41:59] Total duration:17 s
The build failed because a task wrote output to stderr.
Exiting with exit code: 1

```

Figure 20. Confirmation that the bundle command worked correctly

Figure 20 shows confirmation that the building process was successful. Now one can edit the files in the directory. Most guides online do not say the exact order to do these things in. If one had done the edits to the files before the *gulp bundle --ship* command. The edits would be overwritten and go back to their default values. It is unclear if this is just a bug for this application or for others as well. Leave the command prompt open.

Only after one has done the *gulp bundle --ship* command, navigate to the react-script-editor folder and find the lib folder. Open the webparts folder. Then open the scriptEditor folder. Under the scriptEditor folder one will find a file named ScriptEditorWebPart.manifest.json file. At this point, either Visual Studio Code or Notepad++ can be used to edit the files. Since earlier, Visual Studio Code was used, now Notepad++ can be used.

```

8   "manifestVersion": 2,
9   "supportedHosts": ["SharePointWebPart", "SharePointFullPage", "TeamsTab"],
10  "requiresCustomScript": true,
11  "preconfiguredEntries": [{
12    "groupId": "3a328f0a-99c4-4b28-95ab-fe0847f657a3",

```

Figure 21. Change requiresCustomScript to false

Figure 21 shows the contents of the manifest.json file and the field that needs to be changed. Change the row to false instead of true (Svenson, 2021). Depending on the settings of the SharePoint site, this may not be necessary, the suggestion is to change it anyway. Make sure to save the changes to the file.

Figure 22 shows the contents of the package-solution.json file and the row that needs to be edited. One can choose for the application to be deployed to all sites automatically in the tenant or that each site must install the application individually. This step is not necessary if one does not need the feature. Go back to the react-script-editor folder and find the config folder. In the config folder, find the package-solution.json file.

```

7      "includeClientSideAssets": true,
8      "skipFeatureDeployment": false,
9      "isDomainIsolated": false

```

Figure 22. Change skipFeatureDeployment to true

Change the row to true instead of false, if one would like the option for the application to automatically deploy to all sites in the tenant (Svenson, 2021).

Make sure to save the file. One will be able to choose this option when the application is being uploaded to the SharePoint online admin center app catalog. Go back to the command prompt and run the command `gulp package-solution --ship`.

```

13:56:54] ALL DONE!
13:56:54]
13:56:54] Finished subtask 'package-solution' after 532 ms
13:56:54] Finished 'package-solution' after 538 ms
13:56:55] =====[ Finished ]=====
13:56:55] Project pnp-script-editor version:1.0.17
13:56:55] Build tools version:3.12.1
13:56:55] Node version:v10.19.0
13:56:55] Total duration:4.76 s

```

Figure 23. Confirmation that the application is built

Figure 23 shows the confirmation that the build process has been successful. Navigate back to the react-script-editor folder and find the sharepoint folder. Open the solution folder. In the solution folder, one now has a file called `pnp-script-editor.sppkg`. The sppkg application file is what will be upload to the SharePoint admin center. The steps to upload the application to the admin center will be covered in chapter 3.2.7.

3.2.7 SharePoint Admin Center

To be able to upload the application file to SharePoint admin center, administrator rights are needed. To install and deploy the application to SharePoint online, first the application needs to be added to the app catalog. There are several steps that need to be followed and some settings that need to be

changed for the custom applications and custom scripts to show up in SharePoint online. Navigate to SharePoint admin center which can be found by searching for it in google.

Sign into the Microsoft 365 admin center. There will be a menu on the left side of the admin center, with several different options, click on show all. Then click on SharePoint. A new tab will appear, and one will now be in the SharePoint admin center. In the SharePoint admin center on the left side menu, click on the settings tab. Under the Settings tab, one can see the classic settings page options at the bottom of the page, click on it.

- Prevent users from running custom script on personal sites
- Allow users to run custom script on personal sites

- Prevent users from running custom script on self-service created sites
- Allow users to run custom script on self-service created sites

- Enable preview features
- Disable preview features

Figure 24. Allow users to run custom scripts on a SharePoint site

Figure 24 shows the fields that need to be changed to allow custom scripts in SharePoint online. A new tab will appear and there will be many options. Scroll down until the Custom Script section is located.

Go back to the SharePoint admin center home page. On the left side menu find the More features option. Find the more features tab and click on it. Several options appear on the right side of the website. Find the section that says Apps. Click on open under Apps. A new tab will open, and one will see the section Apps. Find the App Catalog option and click on it. One will then be redirected to the SharePoint Apps page. Click on Share SharePoint applications option which will be a green folder icon in the middle of the page. On the new page, click on New to add a new application to the app catalog.



Add as a new version to previously created files

Figure 25. Upload the pnp-script-editor.sppkg file

Figure 25 shows the window where the application file should be uploaded and which application file. Find the react-script-editor folder and navigate to the *pnp-script-editor.sppkg* file that was created in chapter 3.2.6. Once the option *ok* is chosen, a window will appear to make sure that one wants to add the application to the app catalog. Click *enable* in the new window and the file will start to upload.

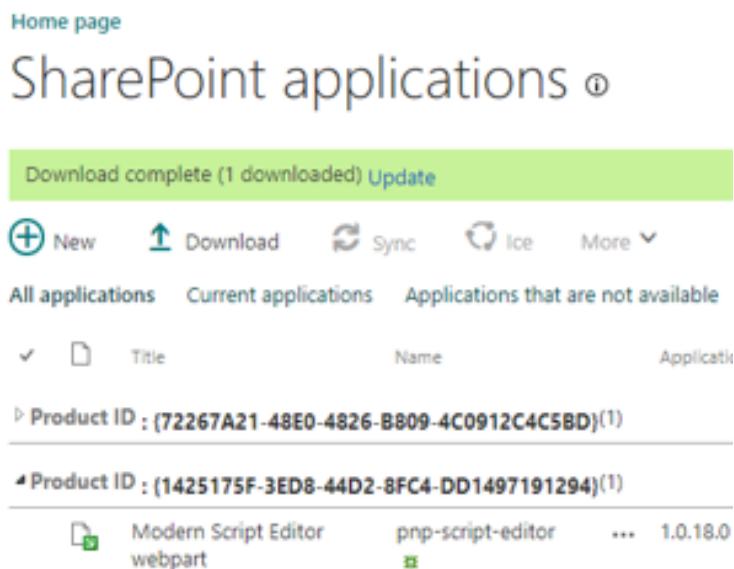


Figure 26. The application will be added with the Product ID and Download complete will be shown as confirmation

Figure 26 shows that the application file has been uploaded successfully. Wait until the section *Deployed* says *Yes* instead of *No*. It may take several seconds for the section *Deployed* to update. The next step is to install the application on the SharePoint Online site or page. The installation of the application into a SharePoint site will be covered in chapter 3.2.8.

3.2.8 SharePoint Site

The final steps are to implement the application on the SharePoint site and page. The most difficult task is already completed. Sign into a SharePoint site (regular SharePoint, not the admin center) and at the top left corner, there will be create a site option. There is an option between Teams site and Communication site. Choose either option, but Teams site is recommended because it has fewer limitations compared to the Communication site. Navigate to the SharePoint site that has just been created.



Figure 27. Site contents page

Figure 27 shows where the site contents page is in the navigation bar. The site contents page will either be at the top menu, or it can be accessed through the settings icon at the top right corner of the site. Navigate to the site contents page, click on New and choose the App option.

One will be redirected to My apps, from here one can browse to the SharePoint store to view other apps, or applications can be added and uploaded to SharePoint admin center. Under Apps that can be added, there will be the Modern Script Editor application that has been uploaded.

Click on Add under Modern Script Editor. One will see in green that the app has been added successfully. It may take a minute or two for the app to be enabled and fully added to the site. Navigate back to the site content page to see that the app has been added and enabled. If it has not been enabled yet, it will be highlighted in grey. Give it some time to finish enabling and turn into a solid color.

Go back to the home page of the SharePoint site. Click on New and choose the Page option to create a new SharePoint page for the site. Choose what type of template one wants for the page. One can choose the blank option and customize the page fully from scratch.

Under the new page, click on the plus sign and search for the web part application like it was shown in Figure 17. Look under Advanced to see the Modern Script Editor web part. Add the Modern Script Editor to the web part. Click on Edit web part at the side of the section and one will have a tab on the right side where the Modern Script Editor can be edited.

Edit HTML Code

```

1 <!-- TradingView Widget BEGIN -->
2 <div class="tradingview-widget-container">
3   <div id="tradingview_e7aa0"></div>
4   <div class="tradingview-widget-copyright"><a href="https://www.tra
5   <script type="text/javascript" src="https://s3.tradingview.com/tv.
6   <script type="text/javascript">
7     new TradingView.widget(

```

Figure 28. Random HTML code added to the script editor to show that it works

Figure 28 shows a sample custom code that has been entered into the Modern Script Editor. Click on the Edit HTML code section and enter a random HTML code. Save after the code has been added. At the top right side of the screen, one will see the republish option. Click on republish and it will publish the changes that have been made to the page.



Figure 29. Custom HTML code working in the script editor

Figure 29 shows that the custom code works in the Modern Script Editor and in the SharePoint online site. The web part will now display the code that was

entered into the Modern Script Editor. SharePoint online modern does not have this capability without the Modern Script Editor.

3.2.9 Conclusion to Modern Script Editor

SharePoint framework and Modern Script Editor improve the capability of customizing SharePoint sites and pages. SPFx lets one create new applications for SharePoint. One can build new web parts or extensions for SharePoint. The GitHub repositories provide many different web parts and extensions that are already built, one just needs to compile them into a SharePoint solution. Modern Script Editor lets anyone enter custom code directly into a SharePoint page. SharePoint online modern has limitations, and with SharePoint Framework and the Modern Script Editor one can lift these limitations. There may be random errors, and not every possibility is covered in this guide. SharePoint framework and Modern Script Editor use depreciated services and software's.

3.3 Chat Bot

Chat bots are becoming popular for companies. With chat bots one can save time and resources when it comes to customer service. In 2022 anyone can have a chat bot with automated responses to common questions, guidance to company resources, connecting a user to a live agent and many other options. This chapter will be going over the steps on how to create a database of questions and answers, how to use Power Virtual Agent to create a chat bot, use Power Automate to connect the database to the chatbot and how to add the chat bot to a SharePoint site. It will also show different types of chat widgets that can be added to a SharePoint site and how to use Omnichannel for live agent transfers.

3.3.1 QnA Knowledge Base

The first step is to create a database of questions and answers. Azure QnA maker will be used for the database. Navigate to Azure QnA maker by searching qnamaker.ai in google. When one is redirected to the website, sign into a Microsoft account to gain access to the QnA maker. After the credentials have been entered, one will be redirected to the home page. Click on Create a

knowledge base in the top menu, after this one will be redirected to qnamaker.ai/Create page.

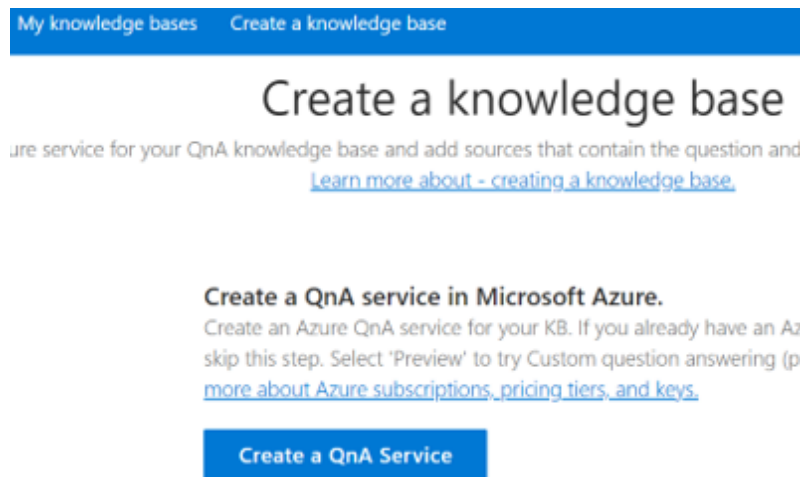


Figure 30. Introduction page to creating a new knowledge base in Azure

If one already has a QnA service in Microsoft Azure, skip this step. If one does not have an Azure service, then it needs to be created. Figure 30 shows the initial page of creating a knowledge base and the button where one should click to create a new QnA service. Click on the link [Create a QnA Service](#) and one will be redirected to Azure. Log into an Azure account and create an Azure service. If one does not have any Azure subscription, then this process can be done with a free trial.

On the Create QnA Maker page, create a resource group and the QnA Maker resource. Scroll down and one will see subscriptions, resource group, name and pricing tier.

Create ...

QnA Maker

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ Visual Studio Enterprise Subscription – MPN

Resource group * ⓘ QnAKB
[Create new](#)

Name * ⓘ QnAKB2 ✓

Pricing tier ([Learn More](#)) * ⓘ

Azure Search details - for data

You have already created 1 F0 for this subscription.
 Standard S0 (\$10 per month for unlimited documents. 3 transactions per second)

Figure 31. Creating a resource group for your QnA service

Figure 31 shows different options that one has when creating a new QnA Maker resource. Choose a subscription for this resource. If one does not have a resource group, then create a new one. Next choose a name and a pricing tier. Under the pricing tier section, there is one available option for a free subscription.

A resource group is a container/storage where all the services will be stored in. The resource group can be used for different services in Azure. It is recommended to create a resource group for each project and not use the same one for everything. Later, when the project is complete, one can delete the whole resource group, everything that is in it will be erased. It is important to know that Azure usually charges hourly for everything unless it is specified that it is free. When one is done with a project, make sure to delete the resource group if it is not needed anymore, so there are no additional fees.

Next, choose where the resource is to be located. Choose the region that one is in or closest too. Choose the location that is closest for processing performance and stability. Click on Review + create to view the summary of what will be created. After the information is reviewed, click create. After the resource is created, navigate back to the initial page of the QnA knowledge base (Figure 30).

One can now proceed to Step 2. In Step 2 the Azure subscription and services will be linked to the QnA Maker. Under Azure QnA service, one will find the service that has been created. Choose the resource and the language that the knowledge base will be in. This step is quite important, so make sure to choose the correct language. Next name the knowledge base.

Populate your KB.

Extract question-and-answer pairs from an online FAQ, product manuals, or other .tsv, .pdf, .doc, .docx, .xlsx, containing questions and answers in sequence. [Learn more](#). Skip this step to add questions and answers manually after creation. The you can add depends on the QnA service SKU you choose. [Learn more about QnA](#)

Enable multi-turn extraction from URLs, .pdf or .docx files. [Learn more.](#)

URL

+ Add URL

File name

+ Add file

Figure 32. Import questions and answers to the knowledge base

Figure 32 shows how one can populate the knowledge base with questions and answers. This is one of the most important parts of creating the knowledge base. Here a URL or file can be added from where the database will pull the questions and answers from. One can enter any website or file and it will automatically try to find questions and answers.

The next option in creating the knowledge base is the Chit-chat option. Choose something as it makes the bot seem more human-like. Finally, create the knowledge base. Depending on how many URLs or files have been chosen, this can take up to several minutes. The creation process will automatically find questions and answers and add them to the database.

Knowledge base

Search the KB 90 QnA pairs

Enable rich editor

Question	Answer
^ Source: qna_chitchat_Friendly.tsv	
I'm so bored of you <input type="text"/>	Swing and a miss.
You're dull <input type="text"/>	+ Add follow-up prompt
That is not exciting <input type="text"/>	
U basic <input type="text"/>	
You're boring <input type="text"/>	

Figure 33. Azure QnA knowledge base questions and answers

Figure 33 shows the question-and-answer pairs. The knowledge base is now created. Here it will show all the questions and answers it was able to find. New QnA (Questions & Answers) pairs can be added manually. On this page one can Test, Save and train, view the settings of the knowledge base (more URLs and files can be added), publish and edit. Click Publish and publish the working knowledge base.

Success! Your service has been deployed

You can always find the deployment details in your service's settings.

Create Bot

[View](#) all your bots on the Azure Portal.

Use the below HTTP request to call your Knowledgebase. [Learn more.](#)

Postman Curl

```
POST /knowledgebases/b947b374-aa39-4bab-87fb-8dac139df758/generateAnswer
Host: https://qnakb.azurewebsites.net/qnamaker
Authorization: EndpointKey 38936585-996b-46cc-b85d-77cfe7d1202c
Content-Type: application/json
{"question":"<Your question>"}
```

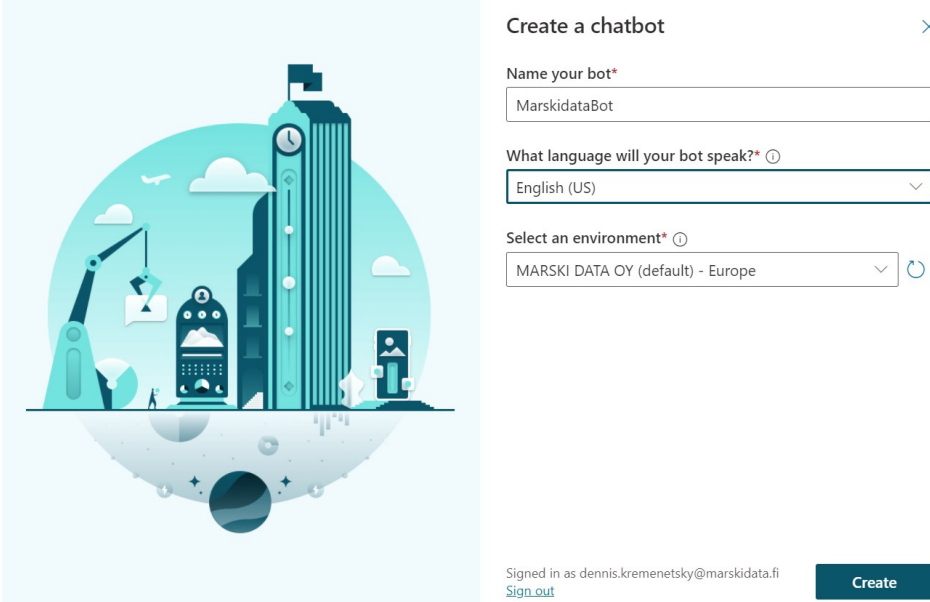
Figure 34. The knowledge base credentials

Figure 34 shows the credentials for the newly created knowledge base. This information will be needed for the chat bot to pull information from the knowledge base. This credentials can be viewed at any time, under My knowledge bases and View Code option. The QnA knowledge base is now ready, test and edit when needed. Power Virtual Agent and Power Automate will now be used to create the chat bot and the connection between the knowledge base and chat bot.

3.3.2 Power Virtual Agent and Power Automate

There are many ways of creating a chat bot but only one way will be shown in this guide. This chapter shows how to use Power Virtual Agent to create a chat bot interface. Navigate to Power Virtual Agent website by searching for it in google. Log into Power Virtual Agent with Microsoft credentials. One can also google for guides on how to create Power Virtual Agent chat bots

(D'Souza-Wiltshire, 2022). If this is not the first chat bot that has been created in the account, click on the bot icon in the top right corner and create a new bot.



Create a chatbot

Name your bot*

MarskidataBot

What language will your bot speak?*

English (US)

Select an environment*

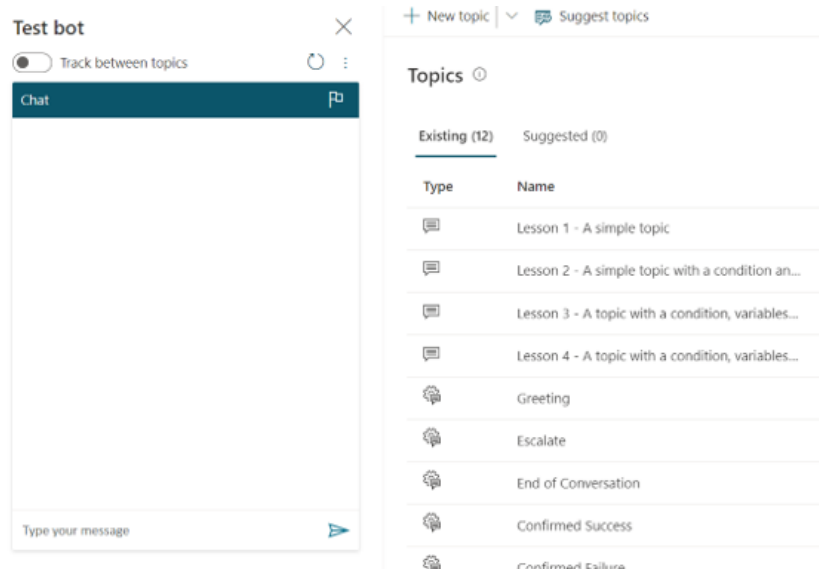
MARSKI DATA OY (default) - Europe

Signed in as dennis.kremenetsky@marskidata.fi
[Sign out](#)

Create

Figure 35. Choose a name for the chat bot and the language

Figure 35 shows where one can create the chat bot name and language. If it is the first chat bot, one will directly go to the create a chatbot window. Name the bot and choose the language that the chat bot will speak. Also select the Microsoft environment where the bot will be stored. The environment is the same as a Resource group in Azure, but for Microsoft. Click create and wait for the creation process to complete.



Test bot

Track between topics

Chat

Type your message

+ New topic | Suggest topics

Topics

Existing (12) | Suggested (0)

Type	Name
Lesson 1 - A simple topic	Lesson 1 - A simple topic
Lesson 2 - A simple topic with a condition an...	Lesson 2 - A simple topic with a condition an...
Lesson 3 - A topic with a condition, variables...	Lesson 3 - A topic with a condition, variables...
Lesson 4 - A topic with a condition, variables...	Lesson 4 - A topic with a condition, variables...
Greeting	Greeting
Escalate	Escalate
End of Conversation	End of Conversation
Confirmed Success	Confirmed Success
Confirmed Failure	Confirmed Failure

Figure 36. Chat bot and topics

Figure 36 shows the chat bot canvas where questions can be asked (left) and different topics (right) which are question and answer groups. The chat bot automatically comes with pre-made topics. One can explore the topics and see how they function. Open each topic and see the process of how a conversation flows with the bot. First start with the Trigger Phrases in the topic. A trigger phrase is what a client/customer types into the chat bot. This triggers a reaction from the chat bot, and all of this can be designed and customized. The chat bot can also route to another topic. There are many different topics one can create that also have multiple choice questions and answers.

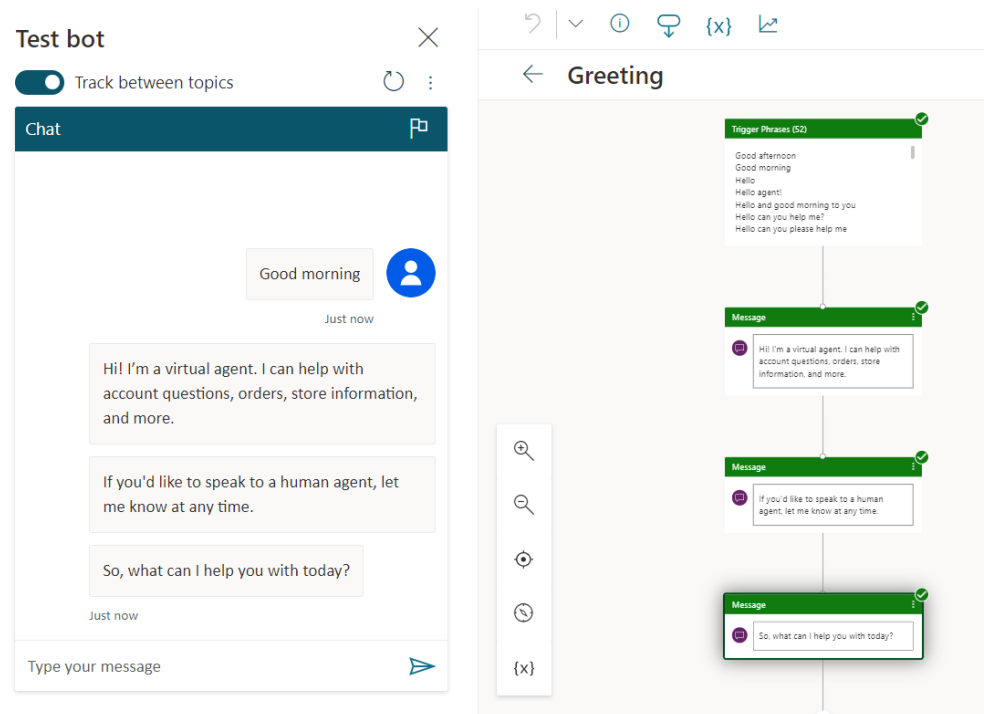


Figure 37. The live flow of the interaction with the chat bot

Figure 37 shows the window where one can enter the questions (left) and the progression through the topic (right) of how the flow goes through the topic. Test the chat bot by typing in the trigger phrases to see how the chat bot reacts. One can activate the Track between topics option to view in real time how the flow of the chat bot operates. One can also test the chat bot without the tracking option to just see the reaction in the chat box.

The next step is to connect the bot to the QnA knowledge base. At the top right corner of the website, click on the Settings icon. Then click on System fallback and click on Add to create a new topic. One will now create a new

system fallback topic, from which a connection will be made by using a Power Automate flow. After the topic is created, click on Go to fallback topic. This is the new topic where one will create the connection between Power Automate and Power Virtual Agent. Look at the new topic and see what connections there are.

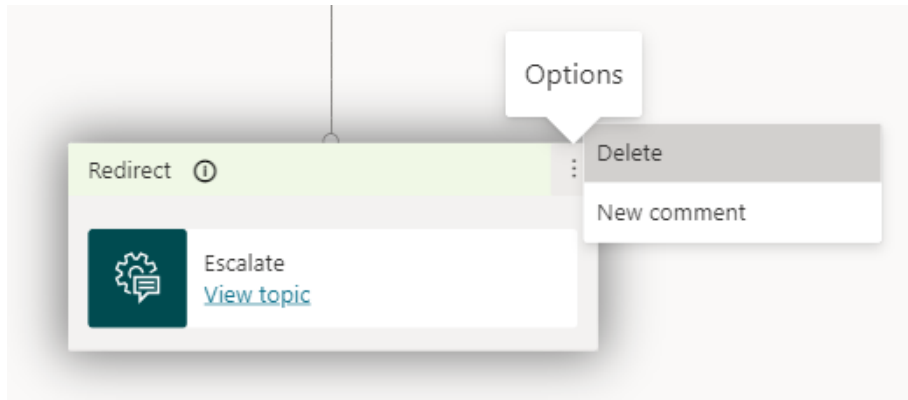


Figure 38. Showing from where to delete the escalate topic redirect

Figure 38 shows the connection in the topic that needs to be deleted. Delete the Escalate topic connection. Hover the mouse right under the Trigger Phrases on the line and one will see a plus sign. Click on the plus sign and add a new node. A new window will appear, click on Call an action and click Create a flow.

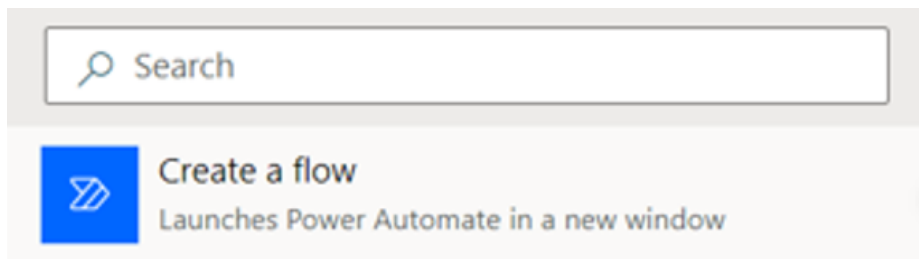


Figure 39. Shows the option to choose to create a new flow in Power Automate

Figure 39 shows a new window with the option to create a new flow. This flow will be the connection between Power Virtual Agent and Power Automate. Power Automate will be launched when the create a flow option is chosen. In Power automate, a flow will be created that will connect the Power Virtual Agent to the Q&A knowledge base (mrbullwinkle, 2022b).

If the account is not logged into Power Automate then first one needs to log in with Microsoft credentials. The flow template that one is redirected to will not

be used, because Power Automate does not automatically support connections with the QnA knowledge base. On the left side of the screen, click on Templates. Create a different template to form a connection between Power Automate, Power Virtual Agent and the knowledge base. Click on the template option, a window will appear asking if one wants to leave, click Leave. On the new page, type in QnA in the search bar and choose the option that appears. The option will say generate answers using QnA maker knowledge base from Power Virtual Agent. Choose the option and one will be redirected to a new page where a connection needs to be made to the Azure QnA resource.

Figure 40. Azure service credentials fields to fill

There may be already an option for the QnA maker connection, or sometimes it needs to be created. One will have to go to the Azure resource group that was created for the QnA knowledge base and retrieve the required credentials (mrbullwinkle, 2022b).

In the Azure resource group, navigate to the QnA maker service. In this service on the left side menu, find Keys and Endpoint. Copy the API key from this section. Figure 40 shows the window of where one needs to enter the API key. Make up a name and click Create and then Continue to create the connection.

A new Power Automate flow will be created. In this new flow one will need to create the connection with the Azure QnA knowledge base. Use the information from the QnA knowledge base credentials (Figure 34). The knowledge

base Id is the information in the POST row. The service host address is in the Host row. The Endpoint Key is in the Authorization row.

The screenshot shows a configuration window titled "Generate answer (Preview)". It contains the following fields:

- * Knowledge Base Id: b947b374-aa39-4bab-87fb-8dac139df758
- * Service Host: https://qnakb.azurewebsites.net/qnamaker
- * Endpoint Key: 38936585-996b-46cc-b85d-77cfefd1202c
- * Question: InputText x

At the bottom left, there is a link "Show advanced options" with a downward arrow.

Figure 41. QnA credentials in the Power Automate flow

Figure 41 shows the locations of where one needs to fill with the credentials for the Q&A knowledge base. Fill out the three fields in the Power Automate flow with the QnA knowledge base credentials. After the three fields have been filled out with the credentials, click Save at the bottom of the flow. (mrbullwinkle, 2022b)

The next step is to create a new solution with the flow. On the left side menu of Power Automate, find the Solutions option and open it. On the new page, click New solution. In the new window, add a name that will be easy to remember from other solutions. If this is in a company network, there may be many solutions. Create the solution after the information has been filled in.

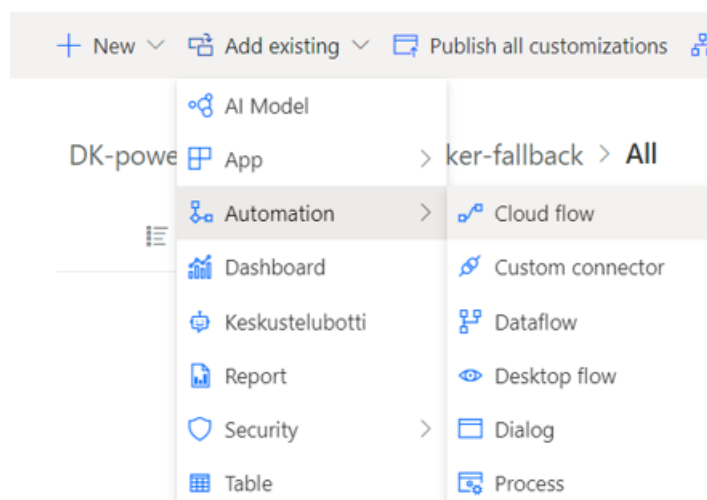


Figure 42. New solution with connection to the flow

Find the new solution that was just created. Figure 42 shows the location of the Add existing option, Automation option and Cloud flow option. Under Add existing option, choose Automation and Cloud flow. In the new window, choose Outside Dataverse and find the flow that has been created where you had added the QnA credentials. After the flow has been connected to the Solution, return to Power Virtual Agent (mrbullwinkle, 2022b).

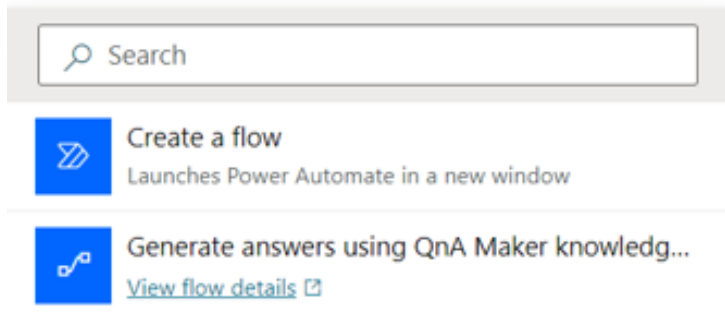


Figure 43. Solution in Power Virtual Agent that you created

As done earlier in this chapter, click on the plus to create a node, choose Call and Action (Figure 39). Figure 43 shows the newly created solution that was created in Power Automate. This is the solution that was created in Power Automate and is now going to link to the Power Virtual Agent.

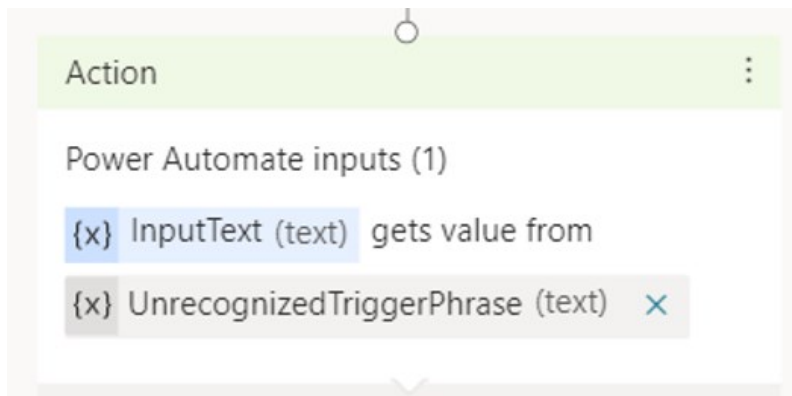


Figure 44. Unrecognized Trigger Phrase action

Figure 44 shows the action node where one needs to make some changes. Under the new Action, click on Enter or select a value and choose UnrecognizedTriggerPhrase. This option specifies that if a client enters something unknown to Power Virtual Agent, then it will seek for a question and answer in the knowledge base.

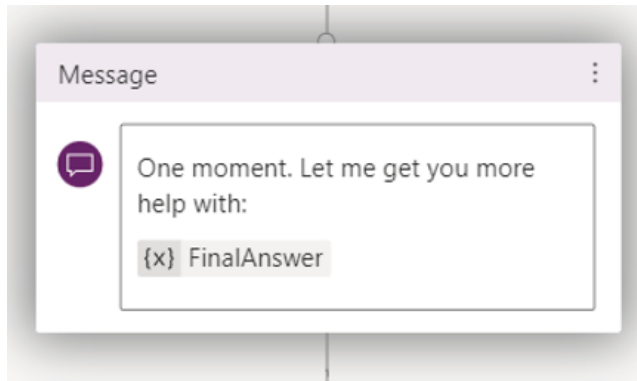


Figure 45. Final Answer function

Figure 45 shows the message node at the end of the flow where one needs to add the final answer option. At the bottom of the flow, one also needs to change the UnrecognizedTriggerPhrase choice. Delete UnrecognizedTriggerPhrase and add the FinalAnswer function. The topic is now ready, click Save. The chat bot can now be tested with the new connection that has been established between the QnA knowledge base and Power Virtual Agent.

Activate the Track between topics option if one would like to see the flow of the topics in real time. Type in a question that is stored in the QnA knowledge base and that is not in the Power Virtual Agent. One will be able to see that the chat bot will respond with answers from the QnA knowledge base.

With Power Virtual Agent it is easy to share the chat bot to websites and other platforms. In the Power Virtual Agent menu on the left, click on Publish. Then click Publish in the window that appears. Confirm that one wants to publish the bot. It may take several minutes for the bot to update and publish. Scroll down and under Optimize the chat bot, click on Go to Channels. In the new section click on Mobile App.

Bot ID

85bbf969-8783-4836-944a-25660708f834

Tenant ID

f73440f1-491a-43e6-8dd3-722c67d4f4f6

Web-based apps

If you're developing a web-based app, copy and paste the following code snippet to your embed web control HTML. If you do not have access to the HTML code, share the code with the person responsible for your web-based app.

Embed code

```
<!DOCTYPE html><html><body><iframe  
src="https://web.powerva.microsoft.com/environments/Default-f73440f1-491a-43e6-8dd3-  
722c67d4f4f6/bots/new_bot_85bbf96987834836944a25660708f834/webchat" frameborder="0"  
style="width: 100%; height: 100%;"></iframe></body></html>
```

Figure 46. Power Virtual Agent bot credentials

Figure 46 shows the credentials of the chat bot from the Mobile App section. Under Mobile App settings one will have the Bot ID, Tenant ID and the Embed code. Use these credentials in the SharePoint site to connect the chat bot to a web part. These credentials will also be used to connect Power Virtual Agent to a chat widget.

Connecting the QnA knowledge base to the Power Virtual Agent chat bot saves a lot of time. It takes less time to create questions and answers in the QnA than it does to create all the connections in Power Virtual Agent. The ability to import and gather questions and answers from URLs and files makes this the best option. One can have a large database of questions and answers for the chat bot. Chapter 3.3.3 shows how to implement the chat bot into a SharePoint online page.

3.3.3 SharePoint Online

Depending on what type of code one has, SharePoint online modern does not support all types of codes in a SharePoint site. To be able to enter different types of code one needs the Modern Script Editor for SharePoint Online modern. Without the Modern Script Editor, the web part supports only certain HTML and iframe codes.

Log into a SharePoint Online site and create a new page. Click on New and choose the Page option to create a new SharePoint page. Choose what type of template one wants for the page. Hover the mouse in the web part section and click on the plus. The plus will open a menu with all web parts that one can use (Figure 17). Choose the Embed web part option. Click on Add embed code and a window on the right will appear. Copy and paste the code from Power Virtual Agent for the chat bot into the section of embed code. One can edit the dimensions of the chat bot in the code section by changing the percentage of the height and width. Click Republish and one will see the chat bot in the web part.

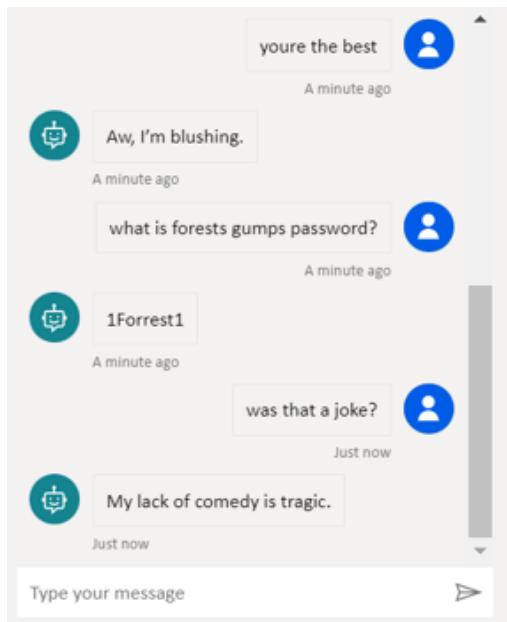


Figure 47. Testing the chat bot in a web part with questions

Figure 47 shows the chat bot working in a web part, and questions and answers from the Q&A knowledge base. The statement, “youre the best” and the question “was that a joke?” come from the chat chat option that was chosen when the knowledge base was first created. The question “what is forests gumps password?” comes from the QnA knowledge base. None of these options were programmed into a topic in Power Virtual Agent.

3.3.4 Chat Widget

There are no simple methods to implement a chat widget for a SharePoint online modern site. There is no solution in the SharePoint admin center or site settings sections. Some options are either to create an application with SPFx

from scratch or build an application using someone else's work. There are chat widgets that are made by Renatoromao, Bot framework and PVA bot host.

3.3.4.1 Renatoromao

One chat widget option is created by renatoromao. Navigate to powerusers website to gain access to his content and a tutorial (Romão, 2020). He also has instructions written in his guide on what to do. As one has already uploaded an application to the app catalog before, there is no need to explain the process again. With Renato's solution, the application is already built into a .sppkg file. All one needs to do is download it and upload it to the app catalog.

Chatbot Webpart X

Bot Settings

BOT ID
3a74a1b4-0c53-4919-90e3-cb7da2e2ba87

BOT Name
FAQ Chat Bot

BOT Logo
https://www.marsskidata.fi/files/2021/04/urn...

BOT Image
https://encrypted-tbn0.gstatic.com/images?...

Figure 48 Configuring renatoromao chat widget

Figure 48 shows the chat bot web part created by renatoromao and the different fields one can customize. After one has added the application to the app catalog and installed it on the site content page. Add a new web part to a page and one will see the option for Chatbot Webpart. Choose a location for it and edit the web part. One can choose the name of the bot, and change the bot logo and image. Enter the BOT ID information from Power Virtual Agent (Figure 46).



Figure 49. Chat bot widget

Figure 49 shows a working chat bot widget in SharePoint online modern page. When one clicks on the widget a chat window appears for the chat bot (Renato Romão 2020). It has the same connection as the web part chat bot from chapter 3.3.3.

3.3.4.2 Bot Framework

Another solution for a chat bot widget for a SharePoint online modern site is using the Bot Framework. Navigate to GitHub and download the repository which has many different types of extensions for SharePoint (Bisser, 2019). Download the repository as one has done earlier in the guide (Figure 12). Once the repository is downloaded, find the react-application-botframework-chat folder under samples. Open the Visual Studio Code software and click on *File* at the top left corner and open folder. Choose the react-application-botframework-chat folder and all the files that are in the folder will be loaded into visual studio code. Navigate to *src/extensions/components* section and find the BotFrameworkChatPopupApplicationChat.tsx file. Click on the file and open it.

```

15  import default class BotFrameworkChatP
16  constructor(props) {
17    super(props);
18    const styleOptions = {
19      hideScrollToEndButton: false,
20      rootHeight: '50%',
21      rootWidth: '50%',
22      hideUploadButton: true
23    };

```

Figure 50. In row 22 add the command to hide the upload button

Figure 50 shows the content of the tsx file and the location of where changes need to be made in the file. Under `cons styleOptions` add the `hideUploadButton: true` if one would like to hide the upload button in the chat window (Bisser, 2019). One can customize many things for the chat window in this file. When the changes are made, click on File and Save All.

Open the command prompt in administrator mode and navigate to the directory where one has the bot framework chat folder. Run the `npm install` command. Next run the `gulp bundle --ship` command. Then run `gulp package-solution --ship` to build the solution for SharePoint online. The gulp command will build the application file which will be upload to the SharePoint online admin center.

Go to the SharePoint admin center and under app catalog, add a new app. These steps were covered in chapter 3.2.7.

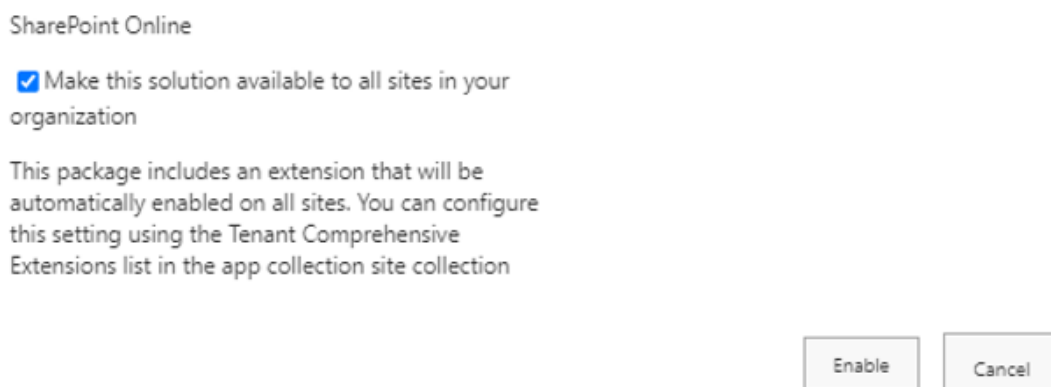


Figure 51. Put a check in the box to enable application for all sites in the organization

Figure 51 shows the process of when the application is being uploaded to the SharePoint admin center and the choices that need to be made. Refer to Figure 25 to remember the process of uploading an application to the SharePoint admin center. Because this is an extension, for the chat widget to work properly, one needs to add the solution to all sites in the tenant. In the next step one will have to edit a Tenant Wide Extension list. This list will not appear if one does not publish the solution to the whole tenant. On the left side of the SharePoint admin center, find the Site Content option. In this section one will find the Tenant-wide extensions folder. If the box in Figure 51 was not

checked, there will not be the component properties file in the extensions folder.



Figure 52. SharePoint site and direct line secret

Figure 52 shows the Component properties field where changes need to be made and credentials filled from Power Virtual Agent. Open the Tenant-wide extensions folder and edit the component properties in the BotFramework-ChatPopup section. Replace the URL address with a site and replace the direct line secret with the secret key from the Power Virtual Agent. To find the direct line secret, go to the Power Virtual Agent. Find the Security option in the left side menu and click on Web channel security. Under Web channel security one will find the Direct Line token named Secret 1. Use this secret code in the component properties. When all the information has been entered, including the SharePoint site, Stop editing to save (Bisser, 2019). Go to the site that was specified in the Component properties. In the bottom left corner, the chat widget will be seen.

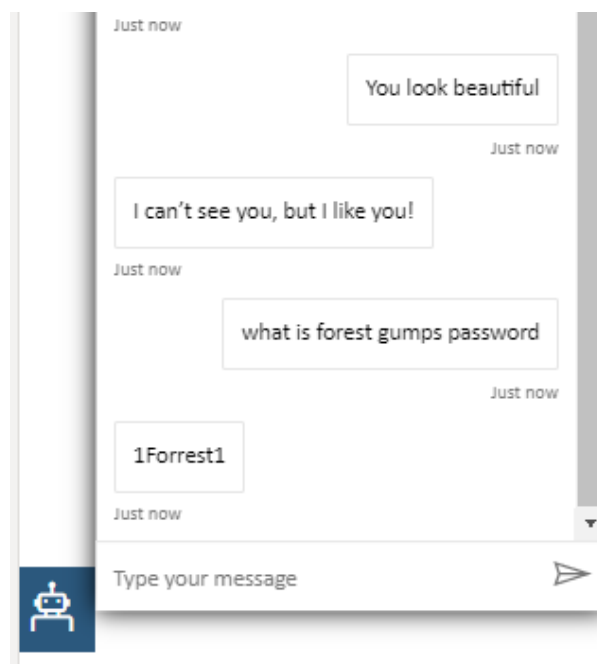


Figure 53. Chat bot widget extension

Figure 53 shows the working chat bot extension on the SharePoint site and that it has a connection to the QnA knowledge base. The chat bot widget is now connected to the Power Virtual Agent and the chat widget will show up on every page on the site. Because the site was specified, this will not show up on every site in the tenant.

3.3.4.3 PVA Bot Host

Another design to host the Power Virtual Agent bot in a SharePoint online site is to use the PVA bot host. Find the react-application-pva-bot folder in the extensions repository that was downloaded in chapter 3.3.4.2. One can also navigate to GitHub, from where the repository is downloaded and find the tutorial provided by Bernier (Bernier, 2021). This extension has two different ways of deploying the application. One can either deploy it to the whole tenant as an extension or deploy it to a single site collection. Chapter 3.3.4.3 shows how to deploy the single site collection chat widget.

Find the react-application-pva-bot folder in the repository. Open command prompt in administrator mode, navigate to the folder and run the command *npm install*.

```

sharepoint > assets > elements.xml
1  version="1.0" encoding="utf-8"?>
2  <?xml xmlns="http://schemas.microsoft.com/sharepoint/"?>
3  <CustomAction
4    Title="PowerVirtualAgentsHost"
5    Location="ClientSideExtension.ApplicationCustomizer"
6    ClientSideComponentId="651e5ed0-c115-4301-beca-b36a667
7    ClientSideComponentProperties="{&quot;botId&quot;: &qu
8  </CustomAction>
9  </elements>

```

Figure 54. Edit row 7 with the chat bot credentials from Power Virtual Agent.

Figure 54 shows the contents of the elements.xml file and the row that needs to be changed with the Power Virtual Agent credentials. Navigate to the folder of the repository and find the sharepoint/assets/elements.xml file. Open the file in Visual Studio and under row 7, edit the ClientSideComponentProperties row. Several changes need to be made in this row to connect the widget to

Power Virtual Agent. Make sure to make the changes carefully and only change the orange sections in the row.

Credentials from the Power Virtual Agent will be needed (Figure 46). Replace “YOUR-BOT-GUID” with the bot Id from Power Virtual Agent. Replace “Contoso Bot” with a chat bot name. Replace “Chat with Contoso Bot” with a chat button label name. Replace “YOUR-TENANT-GUID” with the tenant Id from Power Virtual Agent. Replace “https://your-bot-avatar-image-url” with an image URL. Replace “CB” with the initials for the chat bot. If one wants the chat bot to greet automatically, change the false to true option at the end of the row (Bernier, 2021).

Once the changes have been made in row 7 in Figure 54, click on File and make sure to save all the changes that have been made. There is a command called *gulp dist* that can be used to build the application into a SharePoint application. It runs *gulp clean*, *gulp build*, *gulp bundle --ship* and *gulp package-solution --ship* commands all at once. After the solution has finished building. Navigate to the SharePoint Admin center. Upload the application file in into the app catalog.

When the solution is uploaded, make sure that the check box is not checked. As this is the variation for a single site collection, one will install this as an application on a single site collection.

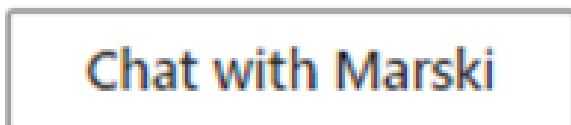


Figure 55. Chat bot widget extension appearance

Figure 55 shows what the chat bot widget looks like on a SharePoint online site. Navigate to the site contents page and add the application. After the application has been added, go to any page in the site. In the footer there will be a tab for the chat bot. Click on the tab and a chat window will open with a connection to Power Virtual Agent.

3.3.5 Omnichannel Live Chat

Omnichannel is a Microsoft 365 Customer Service software that offers the ability to chat with customers. There are a lot of features that are included in the software. Services like messaging, voice chat and accounting options. Omnichannel will be used as an agent transfer for the Power Virtual Agent chat bot. One needs Dynamics 365 Customer Service license for the Omnichannel option to work. If one does not have Dynamics 365 Customer Service license, test this process with a trial. A 30-day trial can be received for free. One of the first things that need to be done, is to go to the Power Platform admin center and install the Omnichannel Power Virtual Agent Extension. The Omnichannel Power Virtual Agent Extension is the only extension needed to get the messaging transfer to work. A tutorial can be found at Microsoft on the different things you can do with Omnichannel (Nellimarla, 2022).

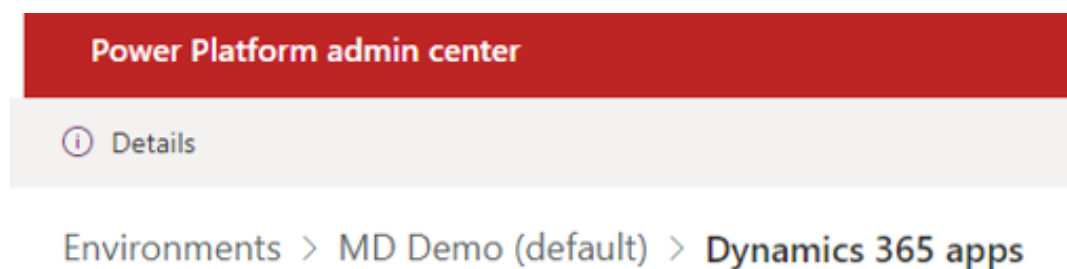


Figure 56. Power Platform admin center

Navigate to the Power Apps admin center. Choose the Environment that the Power Virtual Agent is created in. Choose Dynamics 365 Apps resource and if one does not see the Omnichannel options there, it needs to be added to the Dynamics 365 Apps by using the add option. Figure 56 shows the path of where one needs to go to access the extensions that need to be installed. The extensions need to be installed in the same environment as the Power Virtual Agent, for the Omnichannel to form a connection. Install the Omnichannel Power Virtual Agent Extension. The installation can take several minutes to complete.

Omnichannel for Customer Service - Telephony	...	Enabled
Omnichannel Power Virtual Agent Extension	...	Enabled
Omnichannel Voice Power Virtual Agent Extension	...	Enabled

Figure 57. Extensions needed for voice feature

Figure 57 shows that the extension is enabled, and the other extensions that are needed if one wants the voice feature enabled with the chat bot. In this example there will only be the message feature. For that to work, only the Omnichannel Power Virtual Agent Extension is needed.

Navigate to the Power Virtual Agent after the installation is complete. In the left side menu, under Manage, find Agent transfers option and then click on Omnichannel. If the extension is installed in the same environment as the Power Virtual Agent, then the Enable option will appear.

If this is done with a trial, a new Customer Service Trial environment will be created. One then needs to connect the Power Virtual Agent to an Azure resource; an app registration Id is needed for the Omnichannel to make a connection with the Power Virtual Agent. Follow the steps that are prompted, and one will make all the connections needed.

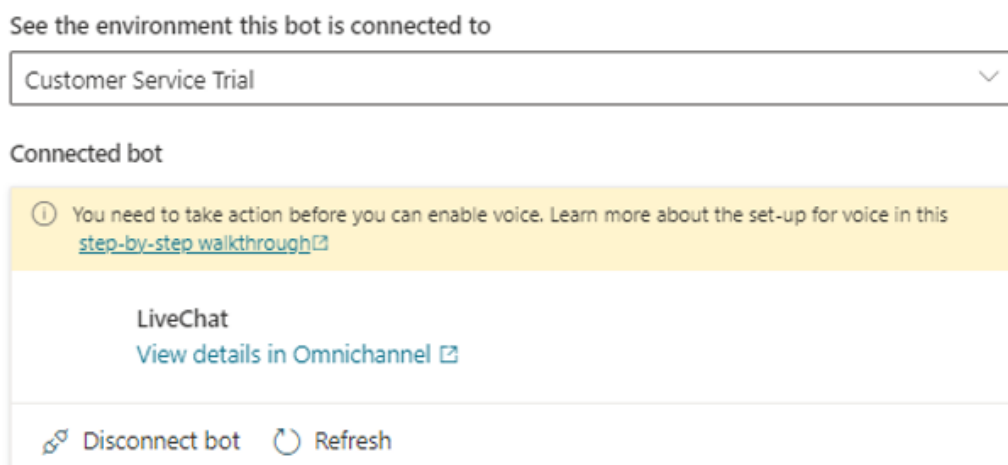


Figure 58. Power Virtual Agent connected to Omnichannel

Figure 58 shows that there is a connection with the chat bot and that it can be disconnected or refreshed. Once the chat bot is connected to the resource

and the Omnichannel, click on View details in Omnichannel. This option will navigate one to the Omnichannel service where one can customize the interactions between the live agent and a client. It can be quite confusing in the Omnichannel admin center at first. Take some time and explore the different options. Check out the Users, Queues and Workstreams.

Omnichannel Users ∨

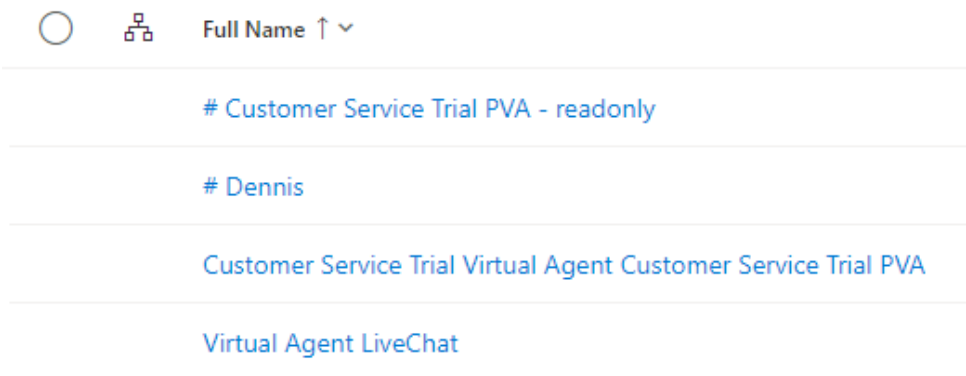


Figure 59. Omnichannel users

Figure 59 shows that the virtual agent chat bot is a user in Omnichannel. In the users section one will see yourself as a user and the Power Virtual Agent bot. Here one can add different people who you want as agents for the live chat function. A user can be an agent who directly talks to a customer, a manager who oversees the agents and a chat bot.

In the Queue section, users can be assigned to different groups. A queue is like a department at a company. When a new queue is created, it can be specified what type of function it will provide: messaging, recording or voice. At first, create a messaging queue and name it as customer service. Once the queue is created, then one can add users to that queue and specify the working hours. The last step is to create a workstream.



Figure 60. Workstream with code snippet and customizations

Figure 60 shows a section in the workstream where the code snippet is for the chat widget and the different behaviours that the chat widget can perform. When a workstream is created, it can be a recording or a messaging service. In the workstream one can customize the chat bot widget and settings. Connect the workstream to the queue and connect the chat bot user to the workstream. The first two things one should do is navigate under Routing rules, connect the queue that was created earlier. Then connect the Power Virtual Agent user in the bot section. After all these steps are done, edit the chat channel and the chat widget.

In the chat channel edit section one will see channel details, chat widget, behaviours and user features. Explore the different options in these sections. There are many customizations that can be done in these sections, explore and experiment with what settings are needed. Once the customizations are done, copy the code snippet. This is the code that needs to be entered into the Modern Script Editor on the SharePoint online site. Leave the Dynamics 365 website open as it will be needed later for testing.

Before the Power Virtual Agent will re-route traffic to Omnichannel one needs to change one more thing in Power Virtual Agent. Navigate to the Power Virtual Agent bot and find the topic called Escalate. At the end of the topic, a new node needs to be added. Add a node by clicking on a plus, choosing *end the conversation* option and transfer to agent. This is needed for the chat bot to re-route traffic from itself to a live agent. Make sure to republish the chat bot. Once the connection is created, go to the SharePoint site and paste the code snippet into the Modern Script Editor web part. Paste the code snippet and republished the site, the customized chat widget will be in the bottom right corner. After the chat widget is connected to the SharePoint site, go back to the Omnichannel site. Keep the SharePoint site open as it will be needed for testing soon.

In Omnichannel admin center, at the top left corner click on Omnichannel admin center and a window will appear with all the different applications. One can choose Customer Service Hub, Customer Service workspace or Omnichannel for Customer Service to test the connection with the SharePoint site. Use the Omnichannel for Customer Service and explore the other options

later. In the SharePoint site, open the chat widget and type in one of the triggers from the Escalate topic, for example, call customer support.

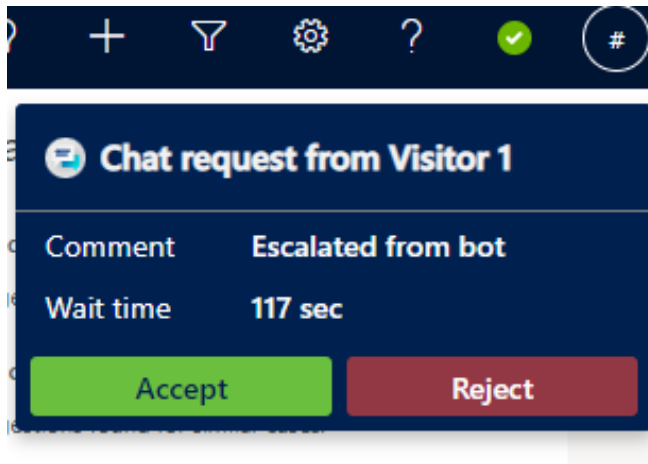


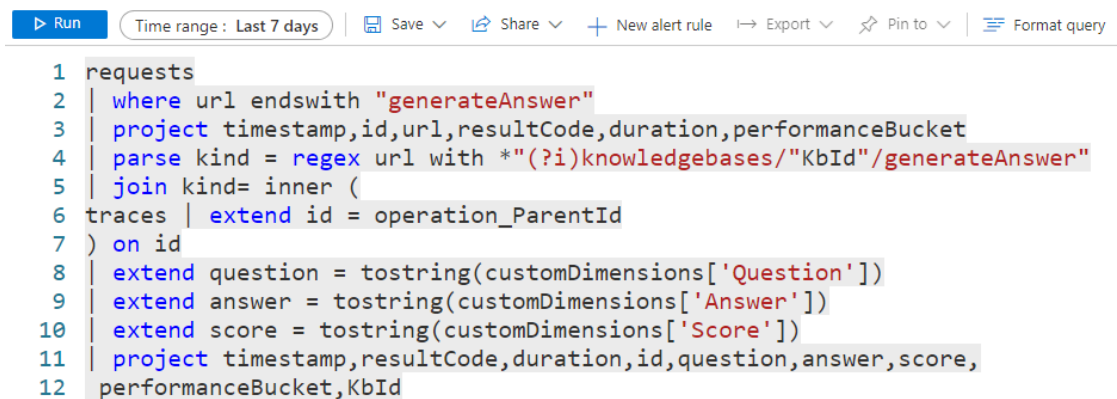
Figure 61. Chat request in Omnichannel through the SharePoint site

Figure 61 shows what the pop-up dialogue will look like when a customer contacts an agent through the chat bot. The Omnichannel will now receive a popup window saying that a visitor would like to chat with you. One can accept or reject the process. If accepted, a new tab will open for the conversation between you and the customer. One thing that should be noticed is that the whole history of the customer chat with the bot will be loaded. This gives the agent good knowledge about what the customer wants to know. There are many resources in the tab that one should explore. For example, the chat history, time used to chat, customer account and contact, new case options and conversation history. The Power Virtual Agent bot is not connected to Omnichannel (which is also connected to the Azure knowledge base) to a live agent through Omnichannel. Explore Omnichannel as there are many features that it can offer. The Omnichannel service can offer the company a lot of resources, organization and save time when dealing with customers.

3.3.6 QnA Analytics

An important feature that needs to be mentioned is the Azure QnA analytics service. It is a service that is already enabled with the QnA knowledge base, and it records and analyses the questions customers ask and the answers that the chat bot provides.

Navigate to Azure and find the resource group that was created for the QnA service. Find the resource called QnAKB-ai, which is the Application Insights. If the QnAKB-ai is not in the resource group, it means that the analytics services were not enabled when the Azure QnA service was created. Under Monitoring, one will see Logs, click on Logs and open it. In the logs option one can run different queries about the knowledge base. Figure 62 provides the query that will pull the questions and answers that were entered through the chat bot. This will give a good idea on what questions customers have been asking and what answers were provided. The questions that were not answered correctly or not answered at all can be added to the QnA knowledge base.



```

1 requests
2 | where url endswith "generateAnswer"
3 | project timestamp,id,url,resultCode,duration,performanceBucket
4 | parse kind = regex url with *"(?i)knowledgebases/"KbId"/generateAnswer"
5 | join kind= inner (
6 traces | extend id = operation_ParentId
7 ) on id
8 | extend question = tostring(customDimensions['Question'])
9 | extend answer = tostring(customDimensions['Answer'])
10 | extend score = tostring(customDimensions['Score'])
11 | project timestamp,resultCode,duration,id,question,answer,score,
12 performanceBucket,KbId

```

Figure 62. Query script to get logs from your chat bot entries (mrbullwinkle 2022)

Figure 62 shows the script that is needed to pull information from the chat bot for the QnA analytics. Type the script in the query window and run the command. One can specify the time range for the data. Once the command has been entered, the analytics of the knowledge base will appear. The data can be exported into an excel file to share or analyse the data and add it to the knowledge base. The code for the query is provided by mrbullwinkle and can be found in the Microsoft website as is his tutorial on the subject (mrbullwinkle, 2022c).

3.3.7 Conclusion to chat bot

Creating a chat bot is not a difficult process, the only thing it requires is time. Once the initial connections are made one can always add more information to the QnA knowledge base later. All the questions and answers need to be entered manually or pulled from a file or URL. Using the QnA knowledge base saves a lot of time compared to creating a topic for each set of questions and

answers in Power Virtual Agent. Power Virtual Agent has many other services it can offer, for example: topics that have questions with multiple choice options, analytics and several more options. Omnichannel can be confusing to set up at first, but it offers a lot of features. To fully understand everything in Omnichannel, time needs to be spent to learn it and experiment with the features. This guide has provided several options for chat widgets and that is because Omnichannel is an expensive service in Microsoft Dynamics 365. The chat widget variations from GitHub are free and open-source solutions.

4 CONCLUSION

SharePoint Framework, Modern Script Editor and a chat bot are great tools to have for SharePoint online modern. All the software and services are required to have the final solution with the Omnichannel working. NVM, NodeJS and NPM are needed to start installing the required software for the SharePoint Framework. Gulp, Yeoman and Microsoft SharePoint generator are installed with NPM to run the SharePoint framework and to build and deploy the Modern Script Editor successfully. Making a chat bot in Power Virtual Agent is easy but it requires a lot of time if one wants many questions and answers. That is why the Azure QnA knowledge base is a better solution, especially if one is making a chat bot for an unsupported language. Omnichannel is a great way to incorporate a live agent transfer and a chat widget into a SharePoint online modern site. Although, Omnichannel is an expensive service and that is why other solutions are provided for chat widgets. SharePoint Framework, Modern Script Editor and the chat bot are the main topics of this guide, everything else is a bonus to improve the base experience.

5 REFERENCES

Bernier, H (2021). sp-dev-fx-extensions/samples/react-application-pva-bot at main · pnp/sp-dev-fx-extensions. [online] Available at: <https://github.com/pnp/sp-dev-fx-extensions/tree/main/samples/react-application-pva-bot> [Accessed 19 Feb. 2022].

Bisser, S (2019). sp-dev-fx-extensions/samples/react-application-botframework-chat at main · pnp/sp-dev-fx-extensions. [online] Available at: <https://github.com/pnp/sp-dev-fx-extensions/tree/main/samples/react-application-botframework-chat> [Accessed 18 Feb. 2022].

Butler, C. (2022). NVM for Windows. [online] GitHub. Available at: <https://github.com/coreybutler/nvm-windows#installation--upgrades> [Accessed 5 Feb. 2022].

D'Souza-Wiltshire, I (2022). Quickstart: Create and deploy a bot on the portal (contains video) - Power Virtual Agents. [online] docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/power-virtual-agents/fundamentals-get-started> [Accessed 17 Feb. 2022].

HelpingAll (2019). How To Install React Script Editor Webpart in Modern Sharepoint Sites Step by Step Full Information. [online] Available at: <https://www.youtube.com/watch?v=Fz3NeLLKrpE> [Accessed 9 Feb. 2022].

Juvonen, V (2022a). Set up your SharePoint Framework development environment. [online] docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/sharepoint/dev/spfx/set-up-your-development-environment> [Accessed 7 Feb. 2022].

Juvonen, V (2022b). Build your first SharePoint client-side web part (Hello World part 1). [online] docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/sharepoint/dev/spfx/web-parts/get-started/build-a-hello-world-web-part> [Accessed 7 Feb. 2022].

Kumar, R (2022). How to Install NVM on Windows. [online] Available at: <https://tecadmin.net/install-nodejs-with-nvm-on-windows/> [Accessed 5 Feb. 2022].

mrBullwinkle (2022a). Quickstart: Create, train, and publish knowledge base - QnA Maker - Azure Cognitive Services. [online] docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/quickstarts/create-publish-knowledge-base> [Accessed 15 Feb. 2022].

mrBullwinkle (2022b). Tutorial: Integrate with Power Virtual Agents - QnA Maker - Azure Cognitive Services. [online] docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/tutorials/integrate-with-power-virtual-assistant-fallback-topic> [Accessed 15 Feb. 2022].

mrBullwinkle (2022c). Analytics on knowledgebase - QnA Maker - Azure Cognitive Services. [online] docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/azure/cognitive-services/qnamaker/how-to/get-analytics-knowledge-base> [Accessed 19 Feb. 2022].

Nellimarla, N (2022). Integrate a Power Virtual Agents bot. [online] docs.microsoft.com. Available at: <https://docs.microsoft.com/en-us/dynamics365/customer-service/configure-bot-virtual-agent> [Accessed 26 Feb. 2022].

Romão, R (2020). How to use Chatbots inside SharePoint pages (no-code) - Power Virtual Agents. [online] Available at: <https://powerusers.microsoft.com/t5/Power-Virtual-Agents-Community/How-to-use-Chatbots-inside-SharePoint-pages-no-code-Power/ba-p/779757> [Accessed 18 Feb. 2022].

Svenson, M (2021). sp-dev-fx-webparts/samples/react-script-editor at main · pnp/sp-dev-fx-webparts. [online] Available at: <https://github.com/pnp/sp-dev-fx-webparts/tree/main/samples/react-script-editor> [Accessed 5 Feb. 2022].

Wikipedia (2019). Visual Studio Code. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Visual_Studio_Code. [Accessed 7 Feb. 2022]

Wikipedia (2020). Node.js. [online] Available at: <https://en.wikipedia.org/wiki/Node.js>. [Accessed 7 Feb. 2022]

wjohnsto (2015). node.js - Fixing npm path in Windows 8 and 10. [online] Available at: <https://stackoverflow.com/questions/27864040/fixing-npm-path-in-windows-8-and-10> [Accessed 6 Feb. 2022].

6 LIST OF FIGURES

Figure 1. NPM command to list all global installations on the system	12
Figure 2. Command to check global installations in the system	13
Figure 3. Command to show the path of where npm modules are installed ...	13
Figure 4. Edit environment variable	14
Figure 5. Folder path for your application	15
Figure 6. SharePoint solution options example	16
Figure 7. Successful creation of the application	17
Figure 8. . Find the serve.json file	17
Figure 9. InitialPage example	18
Figure 10. SharePoint test environment	18
Figure 11. Edit the web part that you created	19
Figure 12. Code option to download as Zip file	20
Figure 13. Connect to SharePoint	21
Figure 14. The command to change the restrictions on your SharePoint site	21
Figure 15. Run npm install command	22
Figure 16. Run the gulp serve command	22
Figure 17. Testing if the application shows up	23
Figure 18. Showing that the web part loads into the web part section	23
Figure 19. Showing that the web part works with the html code	24
Figure 20. Confirmation that the bundle command worked correctly	25
Figure 21. Change requiresCustomScript to false	25
Figure 22. Change skipFeatureDeployment to true	26
Figure 23. Confirmation that the application solution has been built correctly	26
Figure 24. Allow custom scripts	27
Figure 25. Uploading process of the .sspkg file that was created in chapter 3.2.6	28
Figure 26. Confirmation that the application was successfully added	28
Figure 27. Site contents page	29
Figure 28. Add HTML code to your script editor	30
Figure 29. Confirmation that the Modern Script Editor works in a live site	30
Figure 30. Create a new knowledge base	32
Figure 31. Creating a resource group for your QnA service	32
Figure 32. Import questions and answers to your knowledge base	34
Figure 33. Azure QnA knowledge base	34

Figure 34. Your knowledge base credentials.....	35
Figure 35. Name your bot and choose the language.....	36
Figure 36. Chat bot and topics	36
Figure 37. The live flow of your interaction with the bot.....	37
Figure 38. Delete the escalate topic redirect	38
Figure 39. Create a new flow in Power Automate	38
Figure 40. Azure service credentials	39
Figure 41. QnA credentials in the Power Automate flow	40
Figure 42. New solution with connection to the flow.....	40
Figure 43. Solution in Power Virtual Agent that you created	41
Figure 44. Unrecognized Trigger Phrase action	41
Figure 45. Final Answer function	42
Figure 46. Power Virtual Agent bot credentials	43
Figure 47. Testing the chat bot in a web part with questions.....	44
Figure 48. Configuring renatoromao chat widget.....	45
Figure 49. Chat bot widget	46
Figure 50. Hide the upload button	46
Figure 51. Enable application for all sites in your organization.....	47
Figure 52. SharePoint site and direct line secret.....	48
Figure 53. Chat bot widget extension	48
Figure 54. Configure your chat bot extension with your bot credentials	49
Figure 55. Chat bot widget extension	50
Figure 56. Power Platform admin center	51
Figure 57. Extensions needed for voice feature	52
Figure 58. Power Virtual Agent connected to Omnichannel	52
Figure 59. Omnichannel users	53
Figure 60. Workstream with code snippet and customizations.....	53
Figure 61. Chat request in Omnichannel through the SharePoint site.....	55
Figure 62. Query script to get logs from your chat bot entries (mrbullwinkle 2022).....	56