7
Using Power Automate and AI to Build PowerPoint Presentations

If you've ever been faced with a last-minute requirement to prepare a presentation for your boss or an activity club, you're not alone. Whether it's familiar content or something you've never seen before, condensing information and making it accessible to an audience is no small task.

In this chapter, we'll start with everyone's favorite open source knowledge base, Wikipedia, and use ChatGPT to summarize sections of it and insert it into a PowerPoint deck.

With a little bit of tweaking, you can reuse this same type of summarization tooling on documents stored in SharePoint, internal corporate websites, or other digital content sources.

Licensing prerequisites

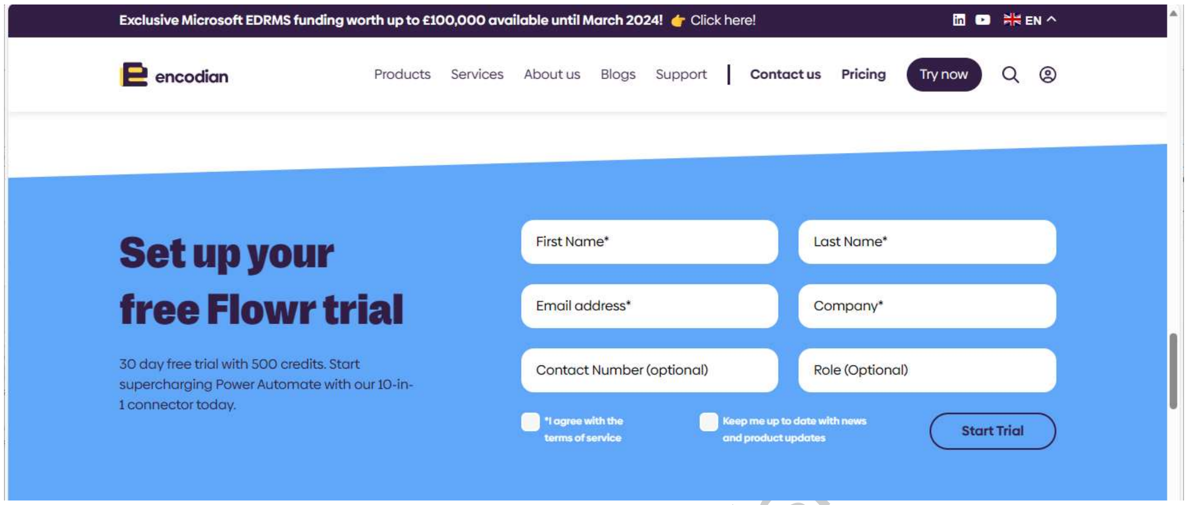
Using AI models and connectors in Power Platform has several prerequisites:

- A subscription that includes Microsoft Dataverse
- AI Builder capacity (or trial capacity)
- Power Apps or Power Automate premium licensing

If you haven't already enabled Dataverse and AI Builder capacity, see *Chapter 2*, *Configuring an Environment to Support AI Services*, and *Chapter 6*, *Processing Data with Sentiment Analysis*.

Since Power Automate does not have a native connector for PowerPoint, this particular solution requires a third-party connector product by software vendor **Encodian**. This connector allows you to transform standard Power Automate objects into PowerPoint content.

Encodian offers a free 30-day trial, including 500 credits, of the Flowr connector at `https://www.encodian.com/product/flowr/`:


Exclusive Microsoft EDRMS funding worth up to £100,000 available until March 2024! Click here!
EN
encodian
Products
Services
About us
Blogs
Support
Contact us
Pricing
Try now
Start Trial

Set up your free Flowr trial

30 day free trial with 500 credits. Start supercharging Power Automate with our 10-in-1 connector today.

First Name*

Last Name*

Email address*

Company*

Contact Number (optional)

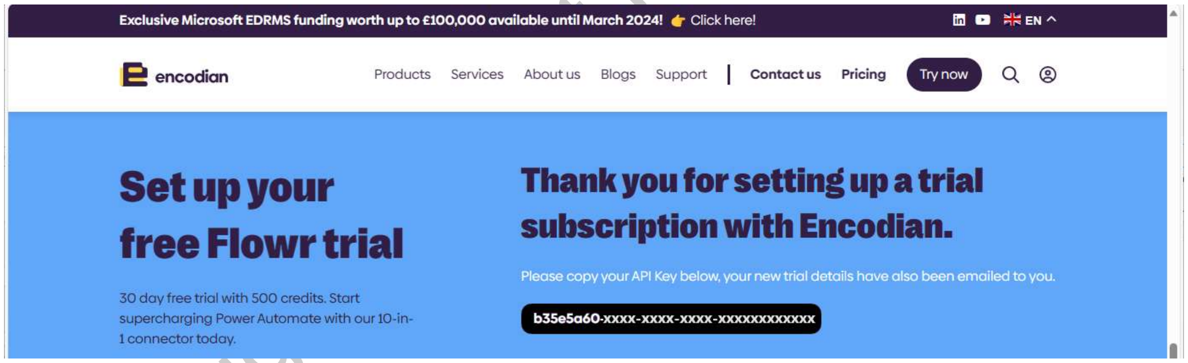
Role (Optional)

*I agree with the terms of service

Keep me up to date with news and product updates

Figure 7.1 – Establishing a Flowr trial

After signing up, you will receive an API key (sample shown in *Figure 7.2*) that you can use to configure the Encodian connector in Power Automate:


Exclusive Microsoft EDRMS funding worth up to £100,000 available until March 2024! Click here!
EN
encodian
Products
Services
About us
Blogs
Support
Contact us
Pricing
Try now

Set up your free Flowr trial

30 day free trial with 500 credits. Start supercharging Power Automate with our 10-in-1 connector today.

Thank you for setting up a trial subscription with Encodian.

Please copy your API Key below, your new trial details have also been emailed to you.

b35e5a60-xxxx-xxxx-xxxx-xxxxxxxxxxxx

Figure 7.2 – Sample Encodian API key

Once that's activated, you'll want to build a generic PowerPoint template file that can be saved and reused whenever you need it!

Learning about the Encodian Flowr connector

Before you get started working on a flow, you'll need to understand how to use the Encodian Flowr connector and its actions. In this section, we'll look at the critical pieces that are needed to make this flow work. To successfully work with the connector and actions, you'll need to understand the following concepts, terminology, and actions:

- Input formatting
- Tokens
- The **Populate PowerPoint** action
- The **Merge Presentations** action

Let's take a quick look at each of these areas!

Input formatting

Many of Encodian's content manipulation actions require the use of structured data. In this case, structured data is typically supplied in the **JavaScript Object Notation** (**JSON**) syntax:

```
{
   "object1" : "value1",
   "object2" :  "value2—electric boogaloo"
}
```

As shown in this simple example, the JSON text is formatted as a **key/value** (think *term:definition*) pair. JSON, like many programming constructs, also has the concept of **arrays** or **collections**. An array or collection is a group of similar objects.

Let's say you wanted to list a collection containing various kinds of foods. It might look something like this when formatted as JSON:

```
{
  "foods": [{
    "category": "fruit",
    "data": [
      {"type": "orange","count": 1},
      {"type": "strawberry","count": 7},
      {"type": "apple","count": 3}]
    },
  {
    "category": "grain",
    "data": [
      {"type": "sorghum","count": 1},
        {"type": "wheat","count": 3}]
    }]
  }
```

In the `foods` JSON object, you can see several examples of how data values are specified. There are both standard key/value pairs (as in `"category":"fruit"`), which identify a single object, as well as arrays (such as `"data": [ {"type": "sorghum","count":1"},{"type":"wheat","count": 3}]`), which are used to identify groups of key/value pairs.

A JSON object can include a variety of data types, including **strings** (text), **integers** (numbers), **arrays** (collections of strings or key/value pairs), **Boolean** values (True or False), as well as other JSON objects.

Further reading

JSON is an extraordinarily flexible format for working with structured content since it's both human-readable as well as platform and language-agnostic. For more information on working with JSON objects and syntax, see `https://www.w3schools.com/JS/js_json_intro.asp`. You can also use tools such as `https://jsonformatter.org/` to help ensure your JSON is formatted correctly.

Tokens

Several of the Flowr actions work with specially formatted **tags** (or **tokens**, as Encodian refers to them), which the connector will use in a find-and-replace fashion with the content you supply.

Let's say, for example, you wanted Flowr to place the words or phrase "The quick brown fox did some things" inside a document. You might decide to assign that phrase to a token named `fox`. Inside the document template, you would indicate the locations you want this text to appear in by typing the token name surrounded by a series of square and angled brackets:

```
<<[fox]>>
```

See *Figure 7.3* for an example of a token inside a PowerPoint slide:

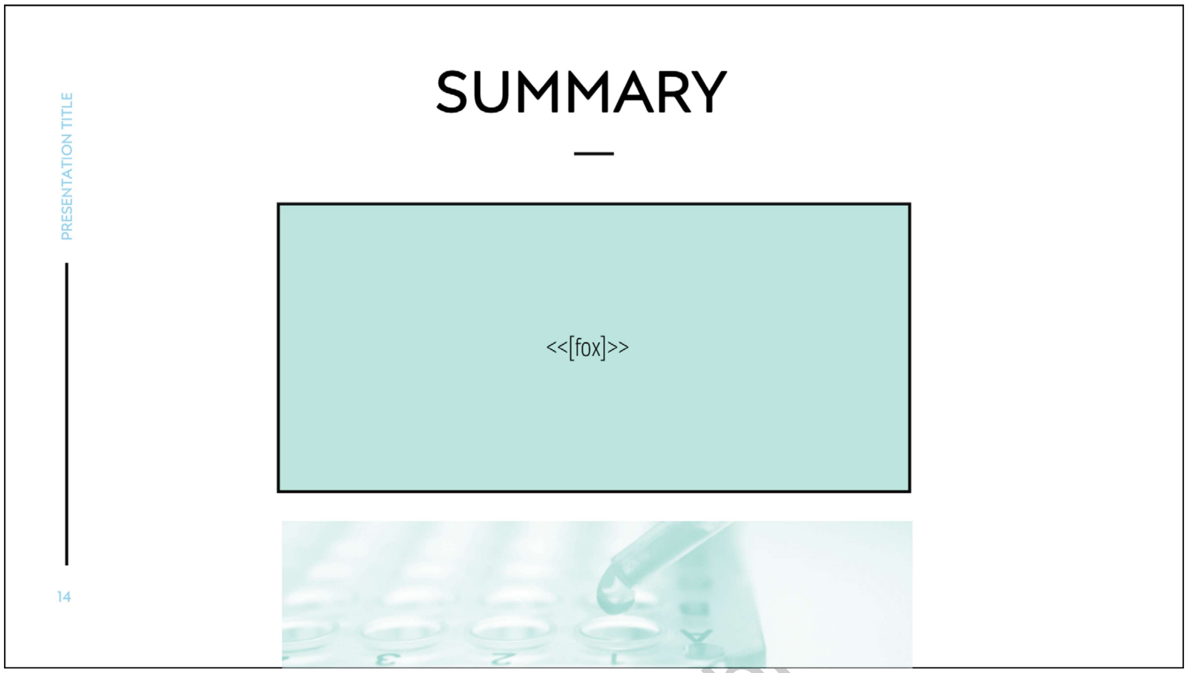


Figure 7.3 – Viewing the <<[fox]>> token inside a PowerPoint slide

Populate PowerPoint

The **Populate PowerPoint** action does just what it says – it's the primary action that's used to place content into a PowerPoint slide. After assigning values to the tokens, the **Populate PowerPoint** action will search the template files, replacing the tokens with the values. The inputs to this will be the token/content key/value pairs and any source PowerPoint template files. The output will be a completed PowerPoint slide.

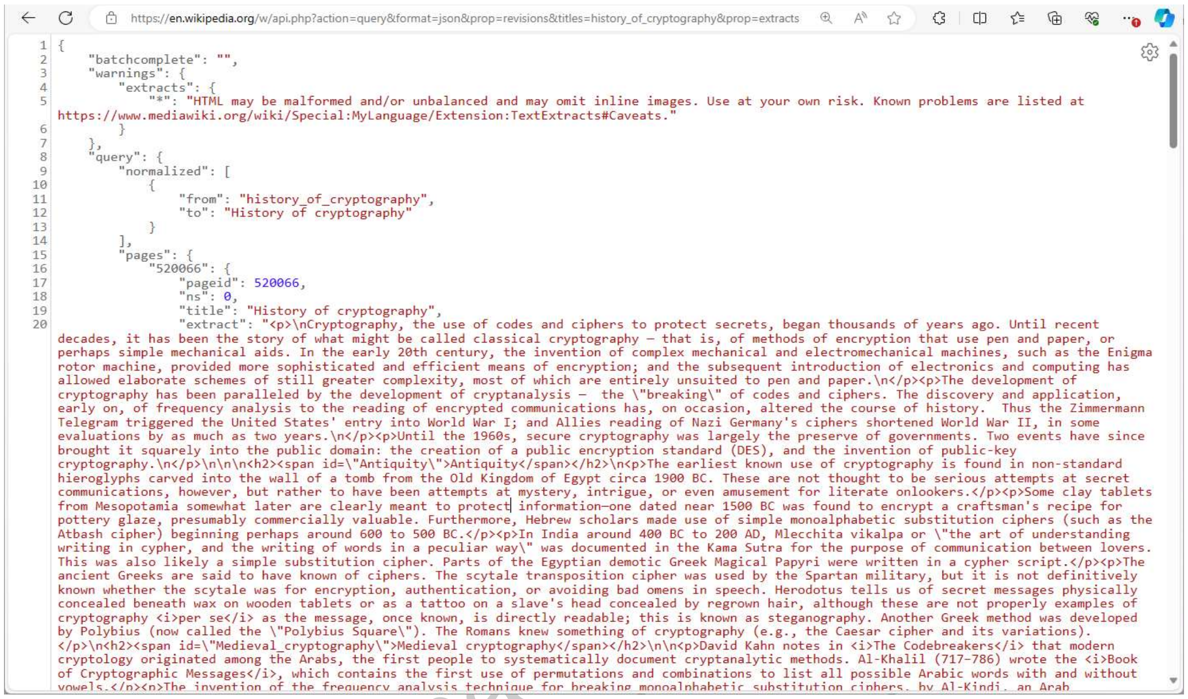
Merge Presentations

Finally, the **Merge Presentations** action can be used to compile the modified PowerPoint pages into a single, cohesive document. The inputs will be the generated PowerPoint files (typically, one slide per file) and the output will be a single multi-slide PowerPoint deck.

Now that you're familiar with the core concepts that we'll be using during the flow, it's time to start working!

Interacting with Wikipedia articles

Since the source of our content for this example is going to be a Wikipedia article, it's important to understand how to gather the data. Wikipedia has an API endpoint (`https://en.wikipedia.org/w/api.php`) that returns the page content as JSON, as shown in *Figure 7.4*:



https://en.wikipedia.org/w/api.php?action=query&format=json&prop=revisions&titles=history_of_cryptography&prop=extracts

```
{
    "batchcomplete": "",
    "warnings": {
        "extracts": {
            "*": "HTML may be malformed and/or unbalanced and may omit inline images. Use at your own risk. Known problems are listed at
https://www.mediawiki.org/wiki/Special:MyLanguage/Extension:TextExtracts#Caveats."
        }
    },
    "query": {
        "normalized": [
            {
                "from": "history_of_cryptography",
                "to": "History of cryptography"
            }
        ],
        "pages": {
            "520066": {
                "pageid": 520066,
                "ns": 0,
                "title": "History of cryptography",
                "extract": "<p>\nCryptography, the use of codes and ciphers to protect secrets, began thousands of years ago. Until recent
decades, it has been the story of what might be called classical cryptography — that is, of methods of encryption that use pen and paper, or
perhaps simple mechanical aids. In the early 20th century, the invention of complex mechanical and electromechanical machines, such as the Enigma
rotor machine, provided more sophisticated and efficient means of encryption; and the subsequent introduction of electronics and computing has
allowed elaborate schemes of still greater complexity, most of which are entirely unsuited to pen and paper.\n</p><p>The development of
cryptography has been paralleled by the development of cryptanalysis —  the \"breaking\" of codes and ciphers. The discovery and application,
early on, of frequency analysis to the reading of encrypted communications has, on occasion, altered the course of history.  Thus the Zimmermann
Telegram triggered the United States' entry into World War I; and Allies reading of Nazi Germany's ciphers shortened World War II, in some
evaluations by as much as two years.\n</p><p>Until the 1960s, secure cryptography was largely the preserve of governments. Two events have since
brought it squarely into the public domain: the creation of a public encryption standard (DES), and the invention of public-key
cryptography.\n</p>\n\n\n<h2><span id=\"Antiquity\">Antiquity</span></h2>\n<p>The earliest known use of cryptography is found in non-standard
hieroglyphs carved into the wall of a tomb from the Old Kingdom of Egypt circa 1900 BC. These are not thought to be serious attempts at secret
communications, however, but rather to have been attempts at mystery, intrigue, or even amusement for literate onlookers.</p><p>Some clay tablets
from Mesopotamia somewhat later are clearly meant to protect information—one dated near 1500 BC was found to encrypt a craftsman's recipe for
pottery glaze, presumably commercially valuable. Furthermore, Hebrew scholars made use of simple monoalphabetic substitution ciphers (such as the
Atbash cipher) beginning perhaps around 600 to 500 BC.</p><p>In India around 400 BC to 200 AD, Mlecchita vikalpa or \"the art of understanding
writing in cypher, and the writing of words in a peculiar way\" was documented in the Kama Sutra for the purpose of communication between lovers.
This was also likely a simple substitution cipher. Parts of the Egyptian demotic Greek Magical Papyri were written in a cypher script.</p><p>The
ancient Greeks are said to have known of ciphers. The scytale transposition cipher was used by the Spartan military, but it is not definitively
known whether the scytale was for encryption, authentication, or avoiding bad omens in speech. Herodotus tells us of secret messages physically
concealed beneath wax on wooden tablets or as a tattoo on a slave's head concealed by regrown hair, although these are not properly examples of
cryptography <i>per se</i> as the message, once known, is directly readable; this is known as steganography. Another Greek method was developed
by Polybius (now called the \"Polybius Square\"). The Romans knew something of cryptography (e.g., the Caesar cipher and its variations).
</p>\n<h2><span id=\"Medieval_cryptography\">Medieval cryptography</span></h2>\n\n<p>David Kahn notes in <i>The Codebreakers</i> that modern
cryptology originated among the Arabs, the first people to systematically document cryptanalytic methods. Al-Khalil (717–786) wrote the <i>Book
of Cryptographic Messages</i>, which contains the first use of permutations and combinations to list all possible Arabic words with and without
vowels.</p><p>The invention of the frequency analysis technique for breaking monoalphabetic substitution ciphers, by Al-Kindi, an Arab
```

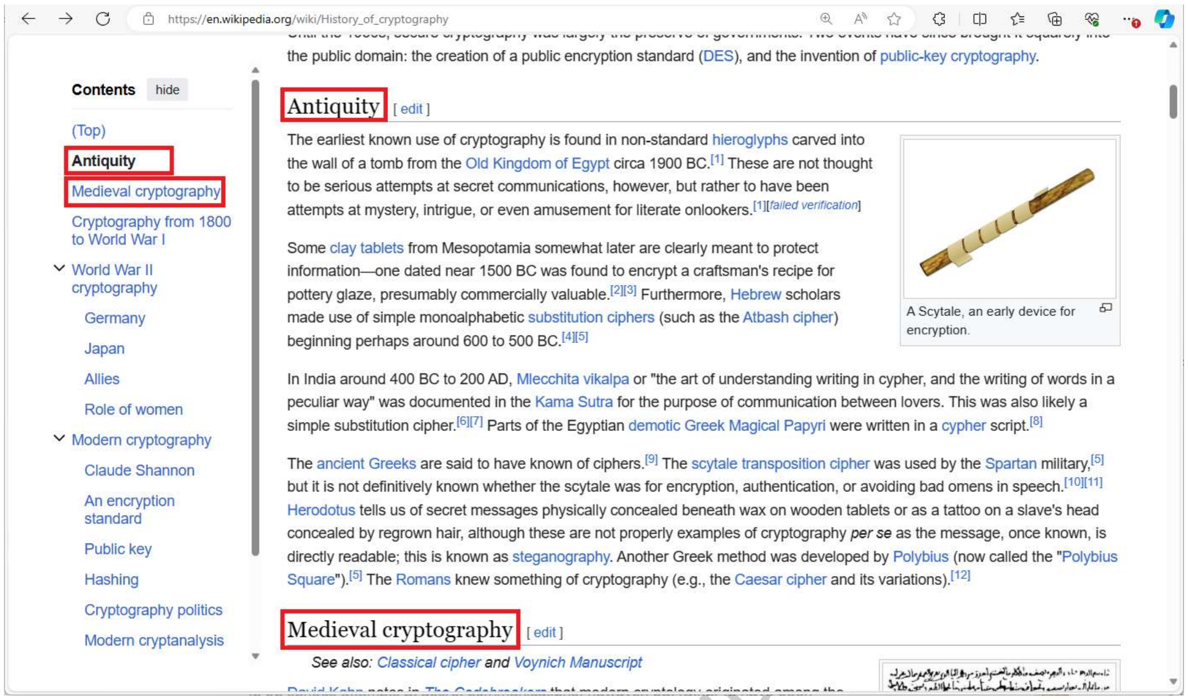
Figure 7.4 – Viewing the JSON output of a Wikipedia article

The link is constructed using the following components:

- API URL: `https://en.wikipedia.org/w/api.php`
- **Action** parameter: Query
- **Format** parameter: JSON
- **Titles** parameter: The title of the Wikipedia article
- Properties (**Prop**) parameter: Extracts (the full content of the article)

The data itself that we'll be using is in the **extract** node, nested inside the **query** JSON object.

When reviewing Wikipedia articles, you might notice that the headings are a mix of heading level 2 (`<H2>`) tags, which are used as main topic headings, and heading level 3 (`<H3>`) tags, which denote subtopics. For this flow, we'll focus on treating the content inside a `<H2>` tag as one unit. *Figure 7.5* shows examples of text being displayed as `<H2>` headings:



https://en.wikipedia.org/wiki/History_of_cryptography

Contents hide

- (Top)
- **Antiquity**
- Medieval cryptography
- Cryptography from 1800 to World War I
- World War II cryptography
  - Germany
  - Japan
  - Allies
  - Role of women
- Modern cryptography
  - Claude Shannon
  - An encryption standard
  - Public key
  - Hashing
  - Cryptography politics
  - Modern cryptanalysis

the public domain: the creation of a public encryption standard (DES), and the invention of public-key cryptography.

Antiquity [edit]

The earliest known use of cryptography is found in non-standard hieroglyphs carved into the wall of a tomb from the Old Kingdom of Egypt circa 1900 BC.[1] These are not thought to be serious attempts at secret communications, however, but rather to have been attempts at mystery, intrigue, or even amusement for literate onlookers.[1][failed verification]

A Scytale, an early device for encryption.

Some clay tablets from Mesopotamia somewhat later are clearly meant to protect information—one dated near 1500 BC was found to encrypt a craftsman's recipe for pottery glaze, presumably commercially valuable.[2][3] Furthermore, Hebrew scholars made use of simple monoalphabetic substitution ciphers (such as the Atbash cipher) beginning perhaps around 600 to 500 BC.[4][5]

In India around 400 BC to 200 AD, Mlecchita vikalpa or "the art of understanding writing in cypher, and the writing of words in a peculiar way" was documented in the Kama Sutra for the purpose of communication between lovers. This was also likely a simple substitution cipher.[6][7] Parts of the Egyptian demotic Greek Magical Papyri were written in a cypher script.[8]

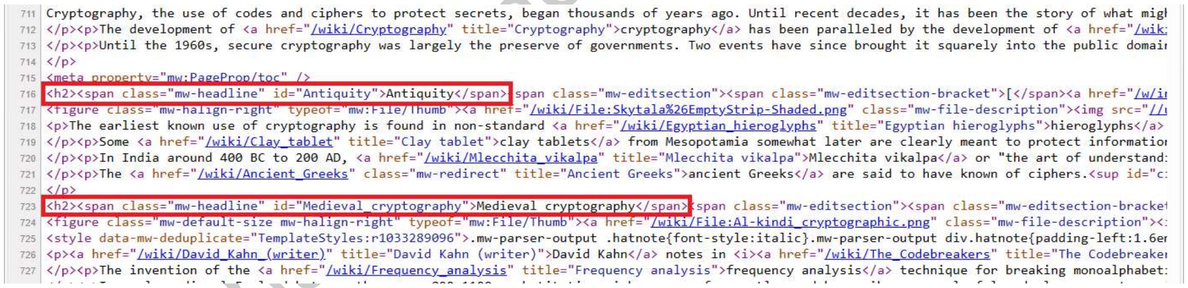
The ancient Greeks are said to have known of ciphers.[9] The scytale transposition cipher was used by the Spartan military,[5] but it is not definitively known whether the scytale was for encryption, authentication, or avoiding bad omens in speech.[10][11] Herodotus tells us of secret messages physically concealed beneath wax on wooden tablets or as a tattoo on a slave's head concealed by regrown hair, although these are not properly examples of cryptography *per se* as the message, once known, is directly readable; this is known as steganography. Another Greek method was developed by Polybius (now called the "Polybius Square").[5] The Romans knew something of cryptography (e.g., the Caesar cipher and its variations).[12]

Medieval cryptography [edit]

See also: Classical cipher *and* Voynich Manuscript

Figure 7.5 – Viewing a Wikipedia article

You can confirm this by viewing the document source in your chosen web browser, as shown in *Figure 7.6*:



```
Cryptography, the use of codes and ciphers to protect secrets, began thousands of years ago. Until recent decades, it has been the story of what migl
</p><p>The development of <a href="/wiki/Cryptography" title="Cryptography">cryptography</a> has been paralleled by the development of <a href="/wik:
</p><p>Until the 1960s, secure cryptography was largely the preserve of governments. Two events have since brought it squarely into the public domaiı
</p>
<meta property="mw:PagePropr/toc" />
<h2><span class="mw-headline" id="Antiquity">Antiquity</span><span class="mw-editsection"><span class="mw-editsection-bracket">[</span><a href="/w/iı
<figure class="mw-halign-right" typeof="mw:File/Thumb"><a href="/wiki/File:Skytala%26EmptyStrip-Shaded.png" class="mw-file-description"><img src="//ı
<p>The earliest known use of cryptography is found in non-standard <a href="/wiki/Egyptian_hieroglyphs" title="Egyptian hieroglyphs">hieroglyphs</a>
</p><p>Some <a href="/wiki/Clay_tablet" title="Clay tablet">clay tablets</a> from Mesopotamia somewhat later are clearly meant to protect informatioı
</p><p>In India around 400 BC to 200 AD, <a href="/wiki/Mlecchita_vikalpa" title="Mlecchita vikalpa">Mlecchita vikalpa</a> or "the art of understand:
</p><p>The <a href="/wiki/Ancient_Greeks" class="mw-redirect" title="Ancient Greeks">ancient Greeks</a> are said to have known of ciphers.<sup id="c:
</p>
<h2><span class="mw-headline" id="Medieval_cryptography">Medieval cryptography</span><span class="mw-editsection"><span class="mw-editsection-brackeı
<figure class="mw-default-size mw-halign-right" typeof="mw:File/Thumb"><a href="/wiki/File:Al-kindi_cryptographic.png" class="mw-file-description"><:
<style data-mw-deduplicate="TemplateStyles:r1033289096">.mw-parser-output .hatnote{font-style:italic}.mw-parser-output div.hatnote{padding-left:1.6eı
<p><a href="/wiki/David_Kahn_(writer)" title="David Kahn (writer)">David Kahn</a> notes in <i><a href="/wiki/The_Codebreakers" title="The Codebreakeı
</p><p>The invention of the <a href="/wiki/Frequency_analysis" title="Frequency analysis">frequency analysis</a> technique for breaking monoalphabet:
```

Figure 7.6 – Viewing the source of a Wikipedia article

In terms of how we'll use this information, we'll be summarizing the content using the following structure:

```
{
     "title" : "value of the <H2> tag"
     "page" : "1 of number of <H2> tags"
     "content" : "AI-generated summary of the content following an
<H2> tag"
}
```

Translating that to how the Flowr connector works, we'll create three tokens:

```
<<[title]>>
<<[page]>>
<<[content]>>
```

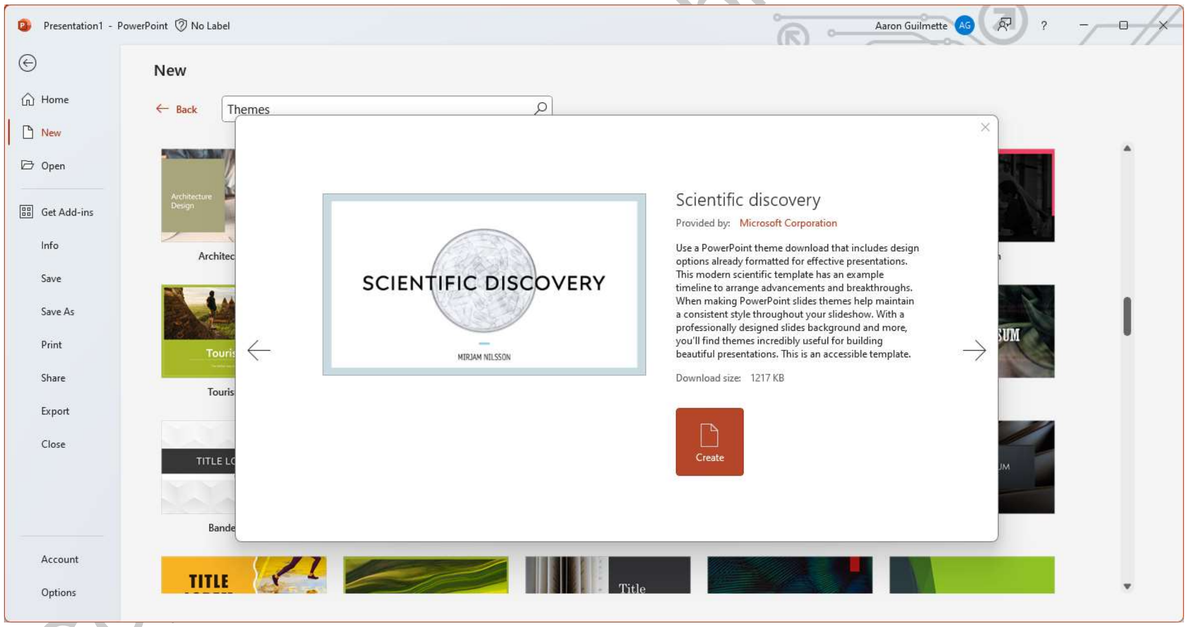
Now that you've got a basic understanding of how this is going to work, let's create a PowerPoint template to hold the generated content!

Creating a PowerPoint template

Before you execute any flows, you need to create a template so that the **Populate PowerPoint** action has something to manipulate. For this example, we'll just use a single slide in the template file.

To create a simple template file, follow these steps:

1. Launch PowerPoint and select **New**. You can either choose **Blank Presentation** or use a themed template file:



Scientific discovery

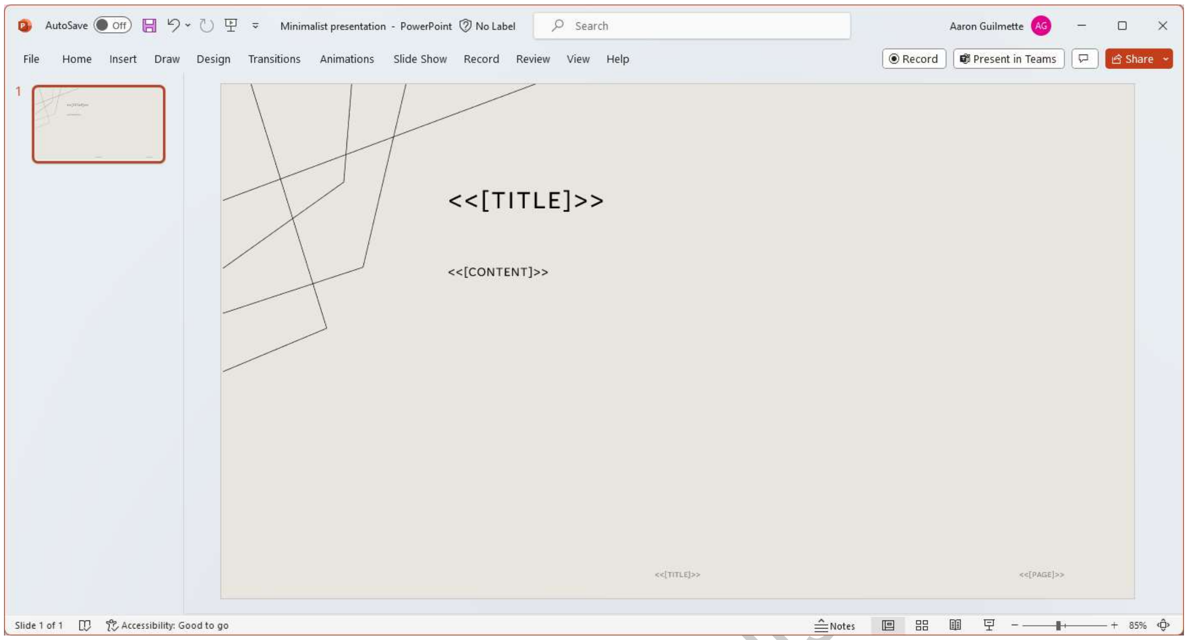
Provided by: Microsoft Corporation

Use a PowerPoint theme download that includes design options already formatted for effective presentations. This modern scientific template has an example timeline to arrange advancements and breakthroughs. When making PowerPoint slides themes help maintain a consistent style throughout your slideshow. With a professionally designed slides background and more, you'll find themes incredibly useful for building beautiful presentations. This is an accessible template.

Download size: 1217 KB

Figure 7.4 – Creating a new template file

2. Edit the content of the file, placing it in the tokens you identified earlier. You can apply formatting such as **bolding** or *italics* to the tokens:


AutoSave Off
Minimalist presentation - PowerPoint No Label
Search
Aaron Guilmette
File Home Insert Draw Design Transitions Animations Slide Show Record Review View Help
Record Present in Teams Share
Slide 1 of 1 Accessibility: Good to go
Notes 85%

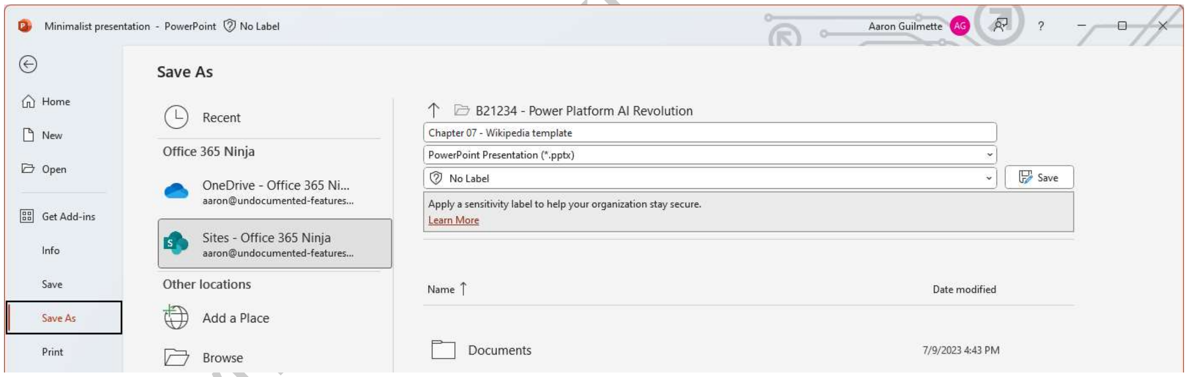
<<[TITLE]>>

<<[CONTENT]>>

<<[TITLE]>> <<[PAGE]>>

Figure 7.5 – Updating the template with the content tokens

3. Save the document in a SharePoint or OneDrive for Business site:



B21234 - Power Platform AI Revolution

Chapter 07 - Wikipedia template

PowerPoint Presentation (*.pptx)

No Label

Apply a sensitivity label to help your organization stay secure.
Learn More

Name	Date modified
Documents	7/9/2023 4:43 PM

Figure 7.6 – Saving a PowerPoint template file

With that done, it's time to start building our flow!

Creating the flow

The flow is going to be comprised of two sections performing discrete actions, divided into scopes to help manage them:

- **Scope 1**: Generating the content summaries
- **Scope 2**: Adding content to the PowerPoint templates

About scopes

Power Automate includes an oft-overlooked control object called a **scope**. A scope is essentially a logical container that can be used to group actions together. Scopes can be expanded and collapsed, allowing you to more easily visualize and manipulate parts of complex flows.

While this flow features the use of scopes, they're essentially organization objects. If you don't feel comfortable adding them, you don't need to.

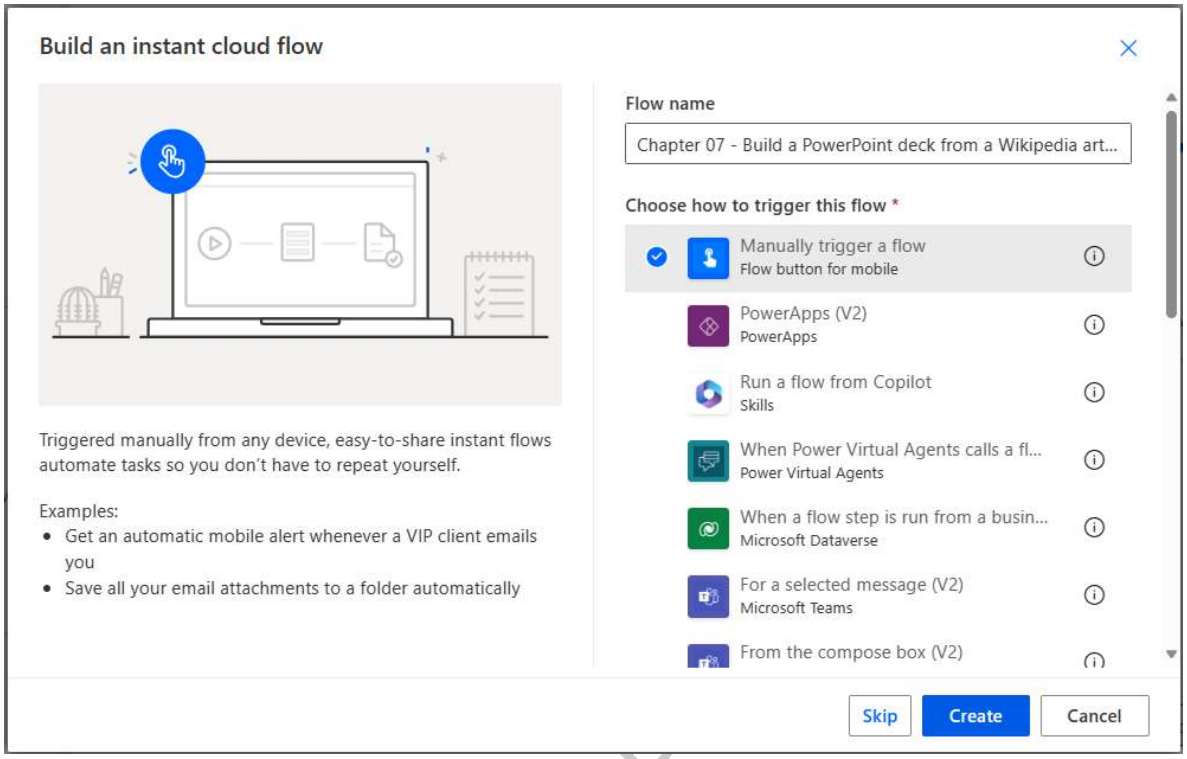
What do I do if I'm stuck?

If you run into a roadblock for some reason (can't find a feature, an option isn't showing up, or something is unclear), help is only a click away! You can download this chapter's artifacts from our GitHub site: `https://github.com/PacktPublishing/Power-Platform-and-the-AI-Revolution`.

Creating the Generate Content Summaries scope

To begin creating the flow, follow these steps:

1. Navigate to the Power Automate Maker Portal (`https://make.powerautomate.com`).
2. From the navigation pane, select **Create**. Then, under **Start from blank**, select **Instant cloud flow**.
3. On the **Build an instant cloud flow** page, enter a **Flow name** value.
4. Under **Choose how to trigger this flow**, select **Manually trigger a flow**. Then, click **Create**:



Flow name

Chapter 07 - Build a PowerPoint deck from a Wikipedia art...

Choose how to trigger this flow *

Manually trigger a flow
Flow button for mobile

PowerApps (V2)
PowerApps

Run a flow from Copilot
Skills

When Power Virtual Agents calls a fl...
Power Virtual Agents

When a flow step is run from a busin...
Microsoft Dataverse

For a selected message (V2)
Microsoft Teams

From the compose box (V2)

Triggered manually from any device, easy-to-share instant flows automate tasks so you don't have to repeat yourself.

Examples:
- Get an automatic mobile alert whenever a VIP client emails you
- Save all your email attachments to a folder automatically

Figure 7.7 – Creating a new flow

5. Click the **Manually trigger a flow** action to expose the **Manually trigger a flow** flyout.
6. Select the **Parameters** tab and choose **Add an input**:

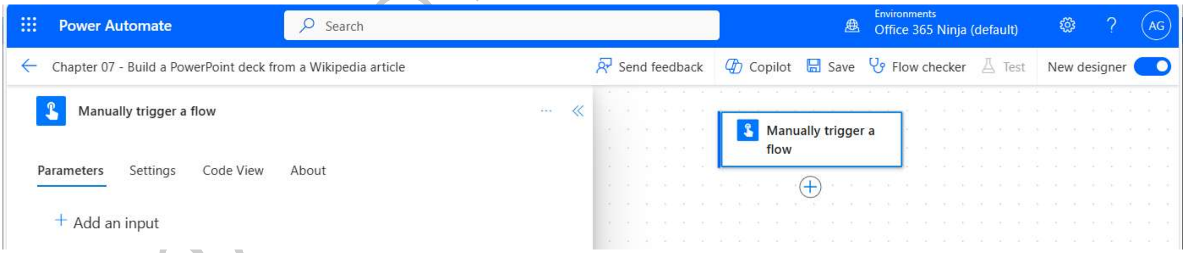


Figure 7.8 – Adding an input

7. For the **Choose the type of user input** prompt, select the **Text** input type.

8. If you wish, modify the input prompt with a description that describes the type of content that should be supplied – for example, `Enter Wikipedia article URL`:

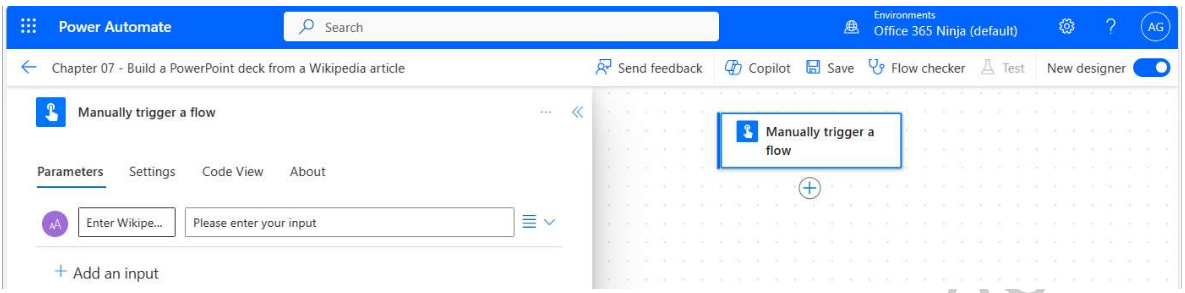


Figure 7.9 – Customizing the text entry prompt

9. Under the **Manually trigger a flow** card, click + and then **Add an action**.
10. In the **Search** box, type `Scope` and select the **Scope control** action:

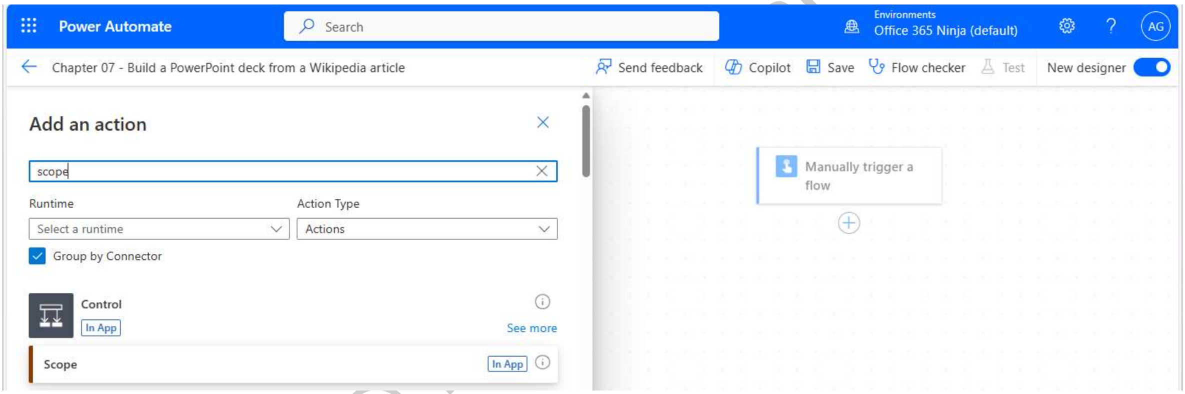


Figure 7.10 – Adding the Scope control action

11. Click the maroon **Scope** title bar to open the **Scope** flyout. Rename the scope to `Scope – Generate Summaries` by clicking on the word **Scope** and editing the field:

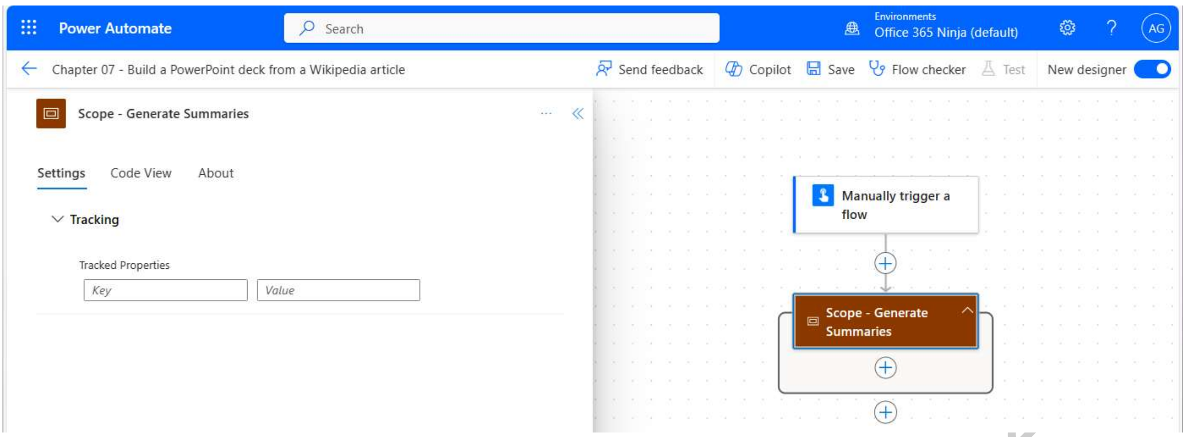


Figure 7.11 – Updating the scope's name

12. Inside the **Scope** card on the canvas, click the + icon and then select **Add an action**.
13. In the **Add an action** flyout, select the **Compose** action:

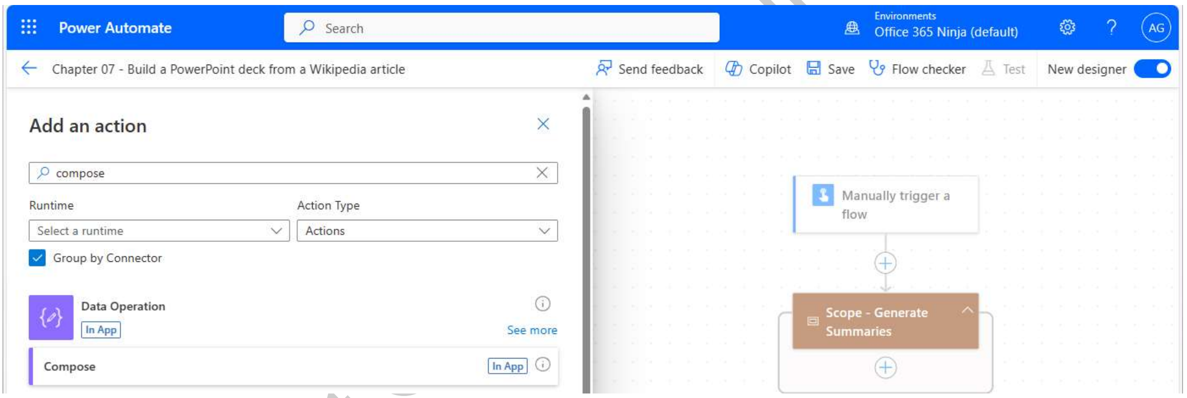
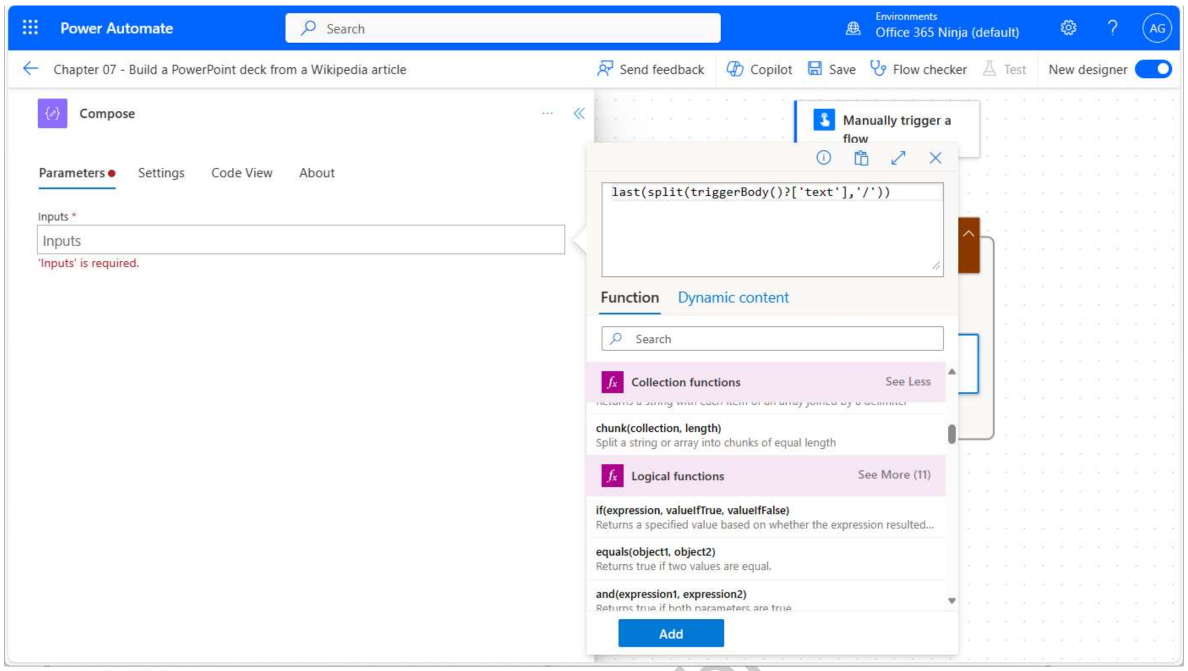


Figure 7.12 – Adding the Compose action

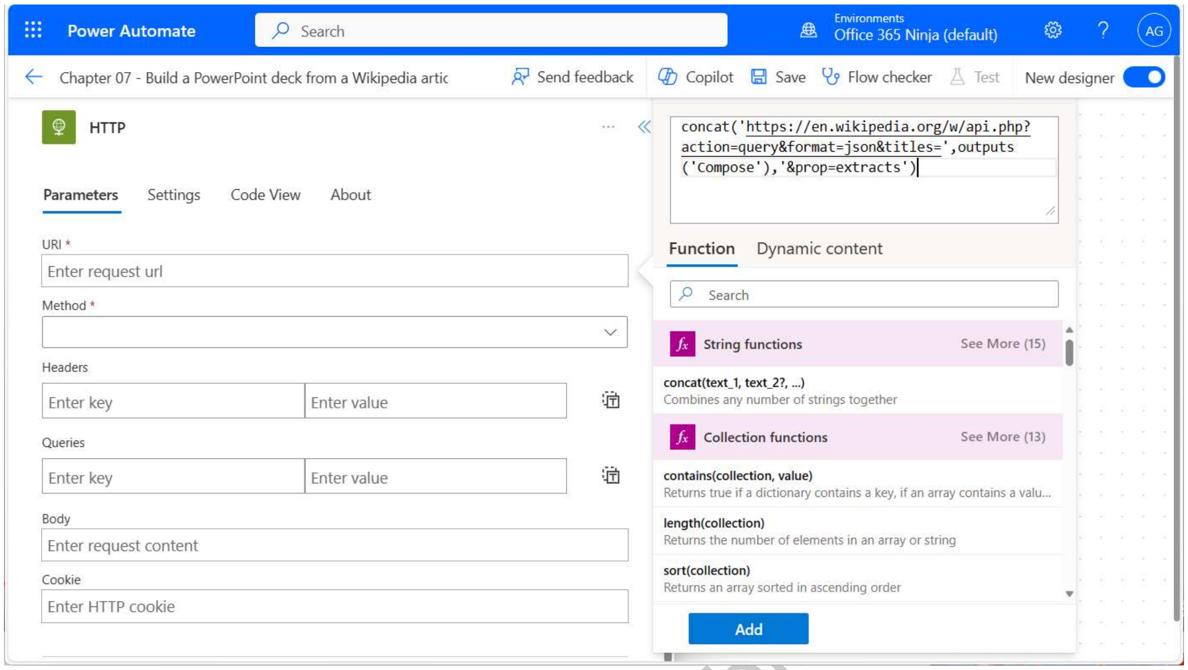
14. In the **Input** box, type a / character or click the *fx* icon to open the **Expression** flyout. In the flyout, enter `last(split(triggerBody()?['text'],'/'))` and click **Add**. Using the / character as a separator, the expression will take the last value from the text input string (from the **Manually trigger a flow** action) and extract the title of the supplied Wikipedia article. For example, when using `https://en.wikipedia.org/wiki/History_of_cryptography`, the expression will return `History_of_Cryptography`. See *Figure 7.13*:


Power Automate
Search
Environments
Office 365 Ninja (default)
Chapter 07 - Build a PowerPoint deck from a Wikipedia article
Send feedback
Copilot
Save
Flow checker
Test
New designer
Compose
Parameters
Settings
Code View
About
Inputs *
Inputs
'Inputs' is required.
Manually trigger a flow
Function
Dynamic content
Search
Collection functions
See Less
chunk(collection, length)
Split a string or array into chunks of equal length
Logical functions
See More (11)
if(expression, valueIfTrue, valueIfFalse)
Returns a specified value based on whether the expression resulted...
equals(object1, object2)
Returns true if two values are equal.
and(expression1, expression2)
Returns true if both parameters are true
Add

```
last(split(triggerBody()?['text'],'/'))
```

Figure 7.13 – Adding the expression

15. Inside the **Scope – Generate Summaries** control card, after the **Compose** card, click + and choose **Add an action**.
16. In the **Add an action** flyout, select the **HTTP** action.
17. On the **Parameters** tab, inside the URI, type a / character or click the *fx* icon to open the **Expression** flyout. In the flyout, enter `concat('https://en.wikipedia.org/w/api.php?action=query&format=json&titles=',outputs('Compose'),'&prop=extracts')` and click **Add**. This expression uses the `CONCAT` function to combine the Wikipedia API endpoint, the query and formatting parameters, and the extracted title value:



```
concat('https://en.wikipedia.org/w/api.php?
action=query&format=json&titles=',outputs
('Compose'),'&prop=extracts')
```

concat(text_1, text_2?, ...)
Combines any number of strings together

contains(collection, value)
Returns true if a dictionary contains a key, if an array contains a valu...

length(collection)
Returns the number of elements in an array or string

sort(collection)
Returns an array sorted in ascending order

Figure 7.14 – Customizing the HTTP action

18. Under the **Method** dropdown, select **GET**.
19. Click **Save**. Do not exit the Power Automate flow canvas.

Next, we'll start configuring the content processing.

Configuring the JSON parameters

In this section, we'll import the JSON schema or definition of how the content is structured. To get the schema output, you'll need to manually construct the API endpoint with the necessary parameters and then insert an article name. Follow these steps:

1. Launch a new browser tab, navigate to `https://en.wikipedia.org`, and search for any article of your choosing. In this example, **History of Cryptography** has been selected. Copy (*Ctrl* + *C*) the last portion of the URL after the final / character – for example, `History_of_cryptography`. See *Figure 7.15*:

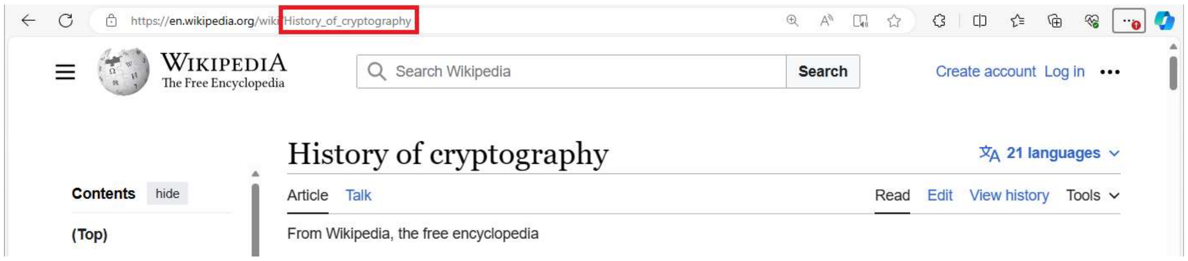
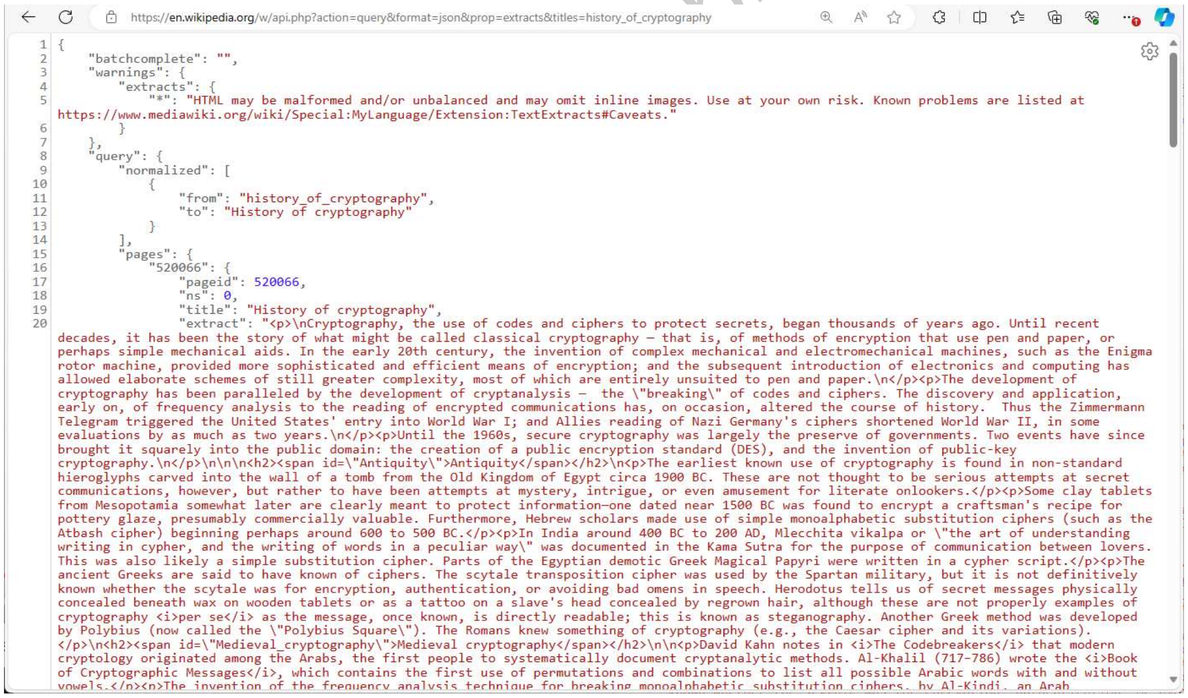


Figure 7.15 – Extracting the relative URL of a Wikipedia article

2. Open a new browser tab and navigate to the Wikipedia API endpoint: `https://en.wikipedia.org/w/api.php`.
3. At the end of the URL, append `?action=query&format=json&prop=extracts&titles=` and then paste the copied Wikipedia article value at the end of the URL bar – for example, `?action=query&format=json&prop=extracts&titles=History_of_cryptograpy`. The result should be a JSON-formatted object that contains the text of the Wikipedia article, as shown in *Figure 7.16*:


https://en.wikipedia.org/w/api.php?action=query&format=json&prop=extracts&titles=history_of_cryptography

```
{
    "batchcomplete": "",
    "warnings": {
        "extracts": {
            "*": "HTML may be malformed and/or unbalanced and may omit inline images. Use at your own risk. Known problems are listed at
https://www.mediawiki.org/wiki/Special:MyLanguage/Extension:TextExtracts#Caveats."
        }
    },
    "query": {
        "normalized": [
            {
                "from": "history_of_cryptography",
                "to": "History of cryptography"
            }
        ],
        "pages": {
            "520066": {
                "pageid": 520066,
                "ns": 0,
                "title": "History of cryptography",
                "extract": "<p>\nCryptography, the use of codes and ciphers to protect secrets, began thousands of years ago. Until recent
decades, it has been the story of what might be called classical cryptography — that is, of methods of encryption that use pen and paper, or
perhaps simple mechanical aids. In the early 20th century, the invention of complex mechanical and electromechanical machines, such as the Enigma
rotor machine, provided more sophisticated and efficient means of encryption; and the subsequent introduction of electronics and computing has
allowed elaborate schemes of still greater complexity, most of which are entirely unsuited to pen and paper.\n</p><p>The development of
cryptography has been paralleled by the development of cryptanalysis —  the \"breaking\" of codes and ciphers. The discovery and application,
early on, of frequency analysis to the reading of encrypted communications has, on occasion, altered the course of history.  Thus the Zimmermann
Telegram triggered the United States' entry into World War I; and Allies reading of Nazi Germany's ciphers shortened World War II, in some
evaluations by as much as two years.\n</p><p>Until the 1960s, secure cryptography was largely the preserve of governments. Two events have since
brought it squarely into the public domain: the creation of a public encryption standard (DES), and the invention of public-key
cryptography.\n</p>\n\n\n<h2><span id=\"Antiquity\">Antiquity</span></h2>\n<p>The earliest known use of cryptography is found in non-standard
hieroglyphs carved into the wall of a tomb from the Old Kingdom of Egypt circa 1900 BC. These are not thought to be serious attempts at secret
communications, however, but rather to have been attempts at mystery, intrigue, or even amusement for literate onlookers.</p><p>Some clay tablets
from Mesopotamia somewhat later are clearly meant to protect information—one dated near 1500 BC was found to encrypt a craftsman's recipe for
pottery glaze, presumably commercially valuable. Furthermore, Hebrew scholars made use of simple monoalphabetic substitution ciphers (such as the
Atbash cipher) beginning perhaps around 600 to 500 BC.</p><p>In India around 400 BC to 200 AD, Mlecchita vikalpa or \"the art of understanding
writing in cypher, and the writing of words in a peculiar way\" was documented in the Kama Sutra for the purpose of communication between lovers.
This was also likely a simple substitution cipher. Parts of the Egyptian demotic Greek Magical Papyri were written in a cypher script.</p><p>The
ancient Greeks are said to have known of ciphers. The scytale transposition cipher was used by the Spartan military, but it is not definitively
known whether the scytale was for encryption, authentication, or avoiding bad omens in speech. Herodotus tells us of secret messages physically
concealed beneath wax on wooden tablets or as a tattoo on a slave's head concealed by regrown hair, although these are not properly examples of
cryptography <i>per se</i> as the message, once known, is directly readable; this is known as steganography. Another Greek method was developed
by Polybius (now called the \"Polybius Square\"). The Romans knew something of cryptography (e.g., the Caesar cipher and its variations).
</p>\n<h2><span id=\"Medieval_cryptography\">Medieval cryptography</span></h2>\n\n<p>David Kahn notes in <i>The Codebreakers</i> that modern
cryptology originated among the Arabs, the first people to systematically document cryptanalytic methods. Al-Khalil (717–786) wrote the <i>Book
of Cryptographic Messages</i>, which contains the first use of permutations and combinations to list all possible Arabic words with and without
vowels.</p><p>The invention of the frequency analysis technique for breaking monoalphabetic substitution ciphers, by Al-Kindi, an Arab
```

Figure 7.16 – Viewing the JSON output of the Wikipedia API

4. Select all the content (*Ctrl* + *A*) and copy it to the buffer (*Ctrl* + *C*).
5. Switch back to the browser tab containing the Power Automate flow.

6. Inside the **Scope – Generate Summaries** control card, after the **HTTP** card, click + and choose **Add an action**.
7. In the **Add an action** flyout, select the **Parse JSON** action:

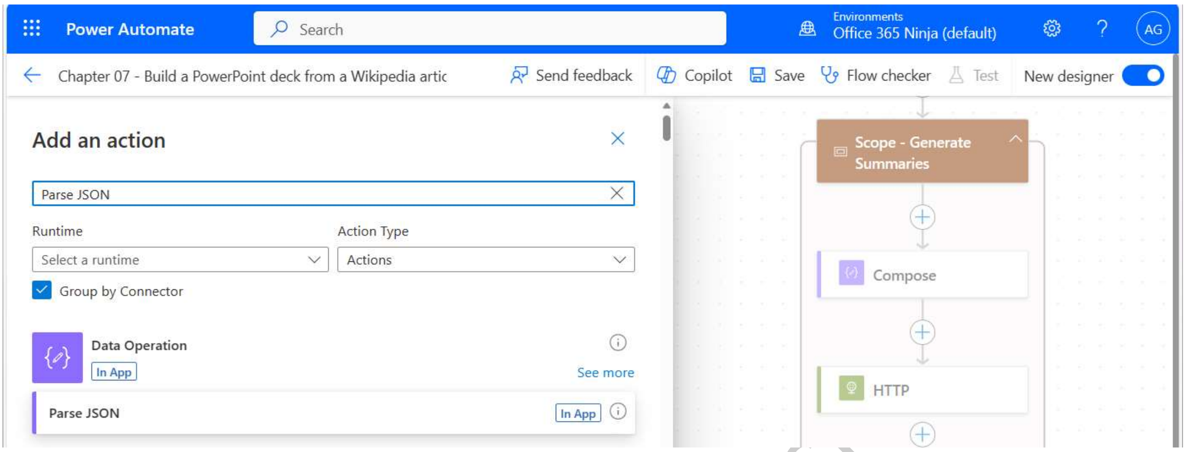


Figure 7.15 – Adding the Parse JSON action

8. In the **Parse JSON** flyout, select the **Parameters** tab.
9. Click inside the **Content** text area, select the dynamic content icon, and then choose the **Outputs** object of the **HTTP** action:

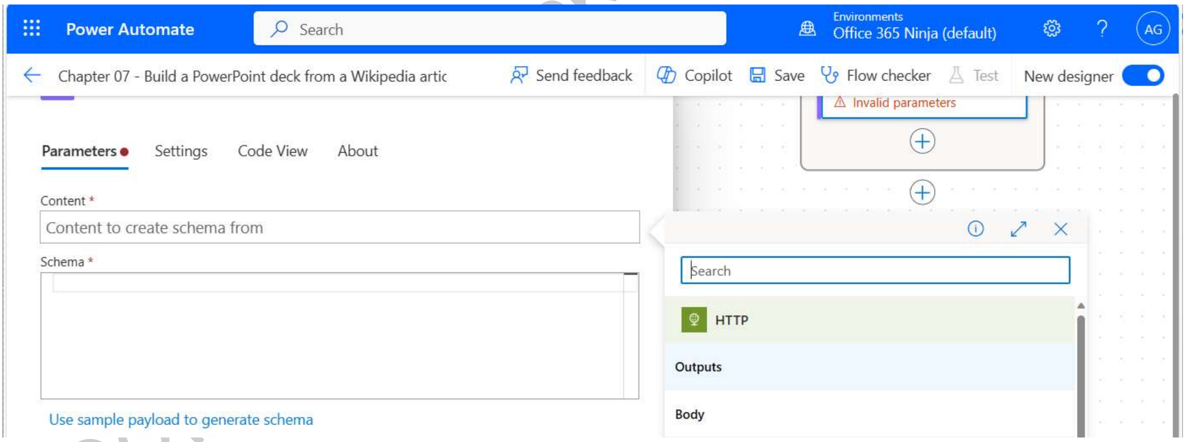
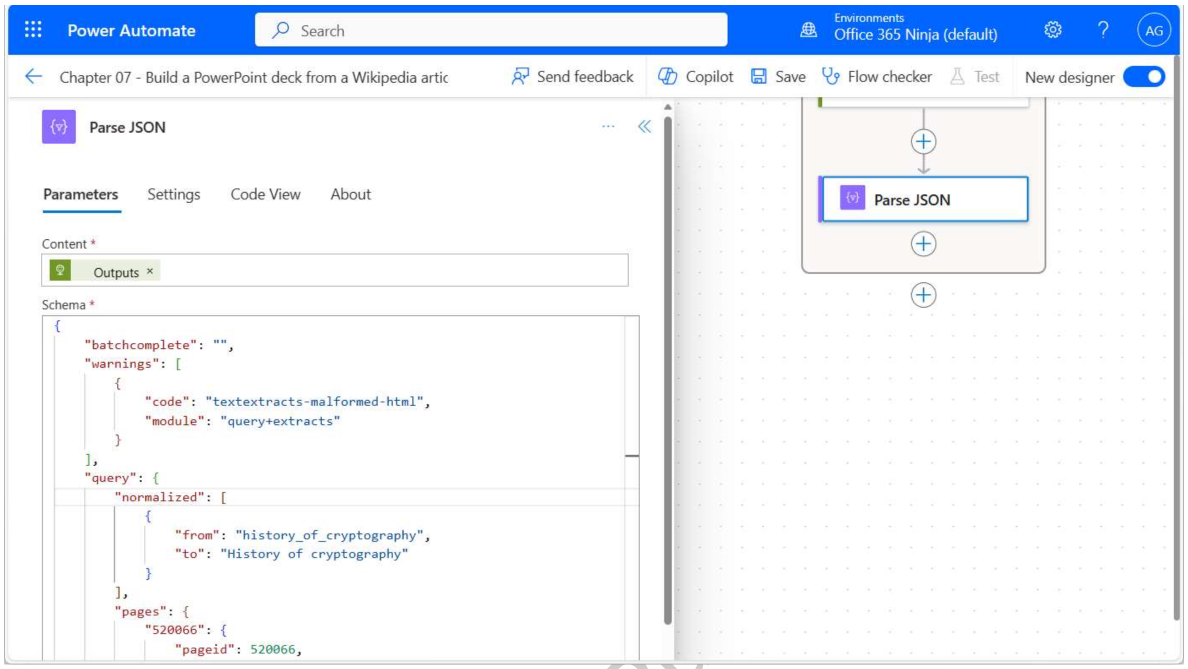


Figure 7.16 – Adding the HTTP Outputs object

10. Paste the contents of the buffer into the **Schema** area, as shown in *Figure 7.17*:



```
{
    "batchcomplete": "",
    "warnings": [
        {
            "code": "textextracts-malformed-html",
            "module": "query+extracts"
        }
    ],
    "query": {
        "normalized": [
            {
                "from": "history_of_cryptography",
                "to": "History of cryptography"
            }
        ],
        "pages": {
            "520066": {
                "pageid": 520066,
```

Figure 7.17 – Populating Content with the Wikipedia API output

11. Click the **Save** icon. Do not exit the Power Automate flow canvas.

Customizing the GPT prompt

Working with completion prompts is as much about art as it is about science. It can take a lot of refinement to get consistently good results.

Prompt frameworks

As the old saying goes, "Garbage in, garbage out." AI models are becoming increasingly more sophisticated and can interpret instructions. To get good results, you need to provide good instructions. Just as people can mimic and learn through examples, so can AI models. Prompt frameworks are one way of helping explain the type of results that meet your expectations. For examples of common prompt frameworks, go to `https://www.undocumented-features.com/2023/12/15/chatgpt-patterns-practices-and-prompts/`.

In this section, you'll create a custom prompt that ChatGPT can use to build the data object. Follow these steps:

1. Inside the **Scope – Generate Summaries** control card, after the **Parse JSON** card, click + and choose **Add an action**.
2. In the **Add an action** flyout, select the **Create text with GPT using a prompt** action:

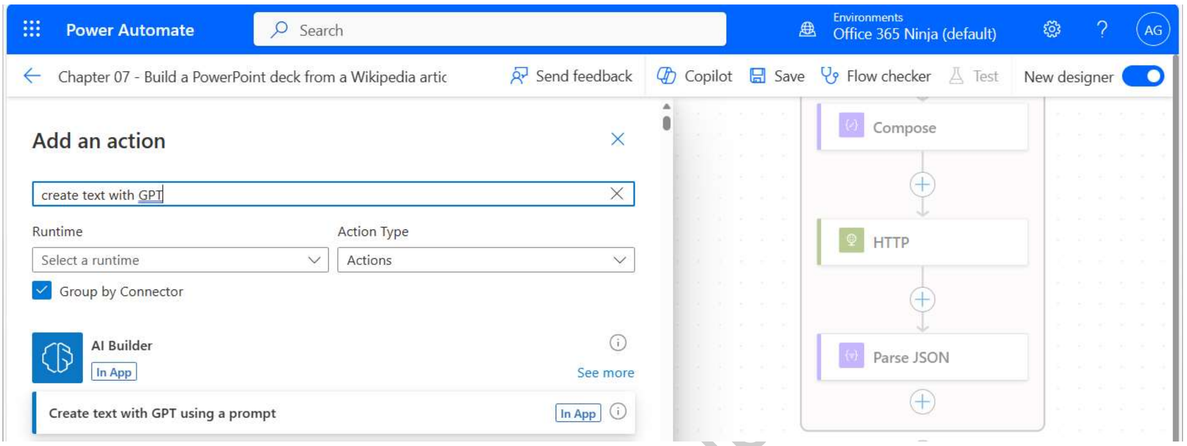


Figure 7.18 – Adding the Create text with GPT using a prompt action

3. In the **Prompt** dropdown, select **New custom prompt**:

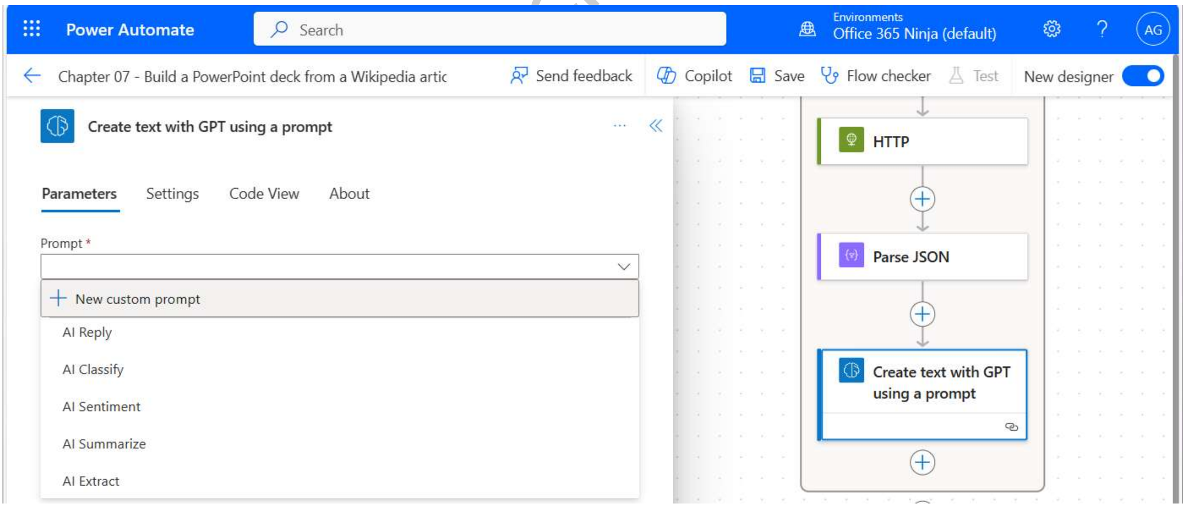


Figure 7.19 – Selecting the New custom prompt option

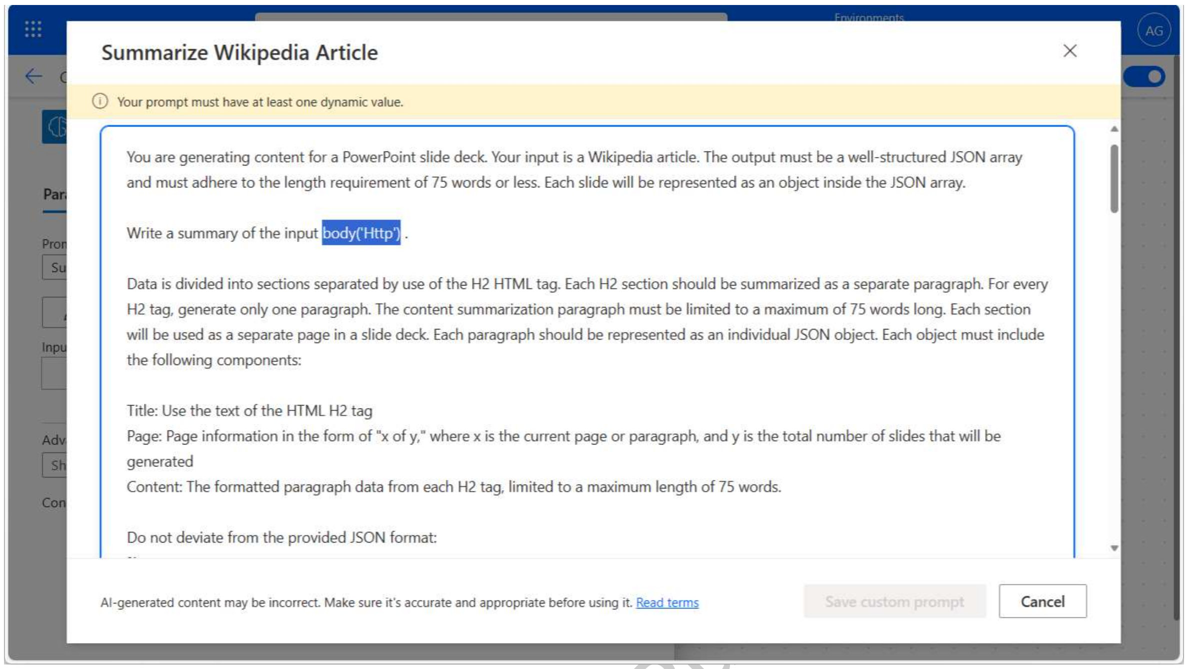
4. Give the prompt a name, such as `Summarize Wikipedia article`.

5. In the **Prompt** area, paste text similar to the following prompt:

```
You are generating content for a PowerPoint slide deck. Your
input is a Wikipedia article. The output must be a well-
structured JSON array and must adhere to the length requirement
of 75 words or less. Each slide will be represented as an object
inside the JSON array. Write a summary of the input body('Http')
.
Data is divided into sections separated by use of the H2
HTML tag. Each H2 section should be summarized as a separate
paragraph. For every H2 tag, generate only one paragraph. The
content summarization paragraph must be limited to a maximum
of 75 words long. Each section will be used as a separate page
in a slide deck. Each paragraph should be represented as an
individual JSON object. Each object must include the following
components:
Title: Use the text of the HTML H2 tag
Page: Page information in the form of "x of y," where x is the
current page or paragraph, and y is the total number of slides
that will be generated
Content: The formatted paragraph data from each H2 tag, limited
to a maximum length of 75 words.


Do not deviate from the provided JSON format:
[{
"title" : "Hanging gardens of Babylon",
"page" : "1 of 1"
"content" : "The hanging gardens of Babylon are one of the seven
wonders of the ancient world. The exquisite, tiered gardens
contained a wide variety of trees, shrubs, flowers, and vines.
According to legend, the Hanging Gardens were built by King
Nebuchadnezzar for his wife, Queen Amytis, because she missed
the gardens and landscape from her homeland. The exact location
of the Hanging Gardens has never been definitively established."
}]
Ensure the output of the JSON array is well-formatted, with a
separate object for each paragraph and the content keyword not
to exceed 75 words.
```

The following screenshot shows this:


Summarize Wikipedia Article
Your prompt must have at least one dynamic value.
AI-generated content may be incorrect. Make sure it's accurate and appropriate before using it. Read terms
Save custom prompt
Cancel

You are generating content for a PowerPoint slide deck. Your input is a Wikipedia article. The output must be a well-structured JSON array and must adhere to the length requirement of 75 words or less. Each slide will be represented as an object inside the JSON array.

Write a summary of the input body('Http') .

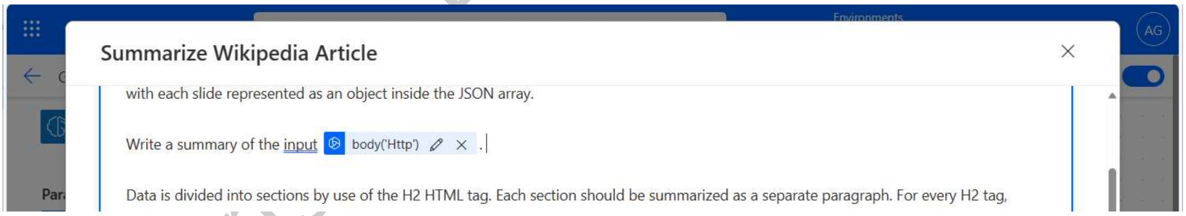
Data is divided into sections separated by use of the H2 HTML tag. Each H2 section should be summarized as a separate paragraph. For every H2 tag, generate only one paragraph. The content summarization paragraph must be limited to a maximum of 75 words long. Each section will be used as a separate page in a slide deck. Each paragraph should be represented as an individual JSON object. Each object must include the following components:

Title: Use the text of the HTML H2 tag
Page: Page information in the form of "x of y," where x is the current page or paragraph, and y is the total number of slides that will be generated
Content: The formatted paragraph data from each H2 tag, limited to a maximum length of 75 words.

Do not deviate from the provided JSON format:

Figure 7.20 – Adding a prompt value

1. Notice the banner, indicating that a dynamic value is required. Copy the value highlighted in *Figure 7.20*, **body('Http')**, by pressing *Ctrl* + *C*. Then, click **Add dynamic value**. The highlighted portion of the prompt should be replaced with a dynamic value placeholder. See *Figure 7.21*:



with each slide represented as an object inside the JSON array.

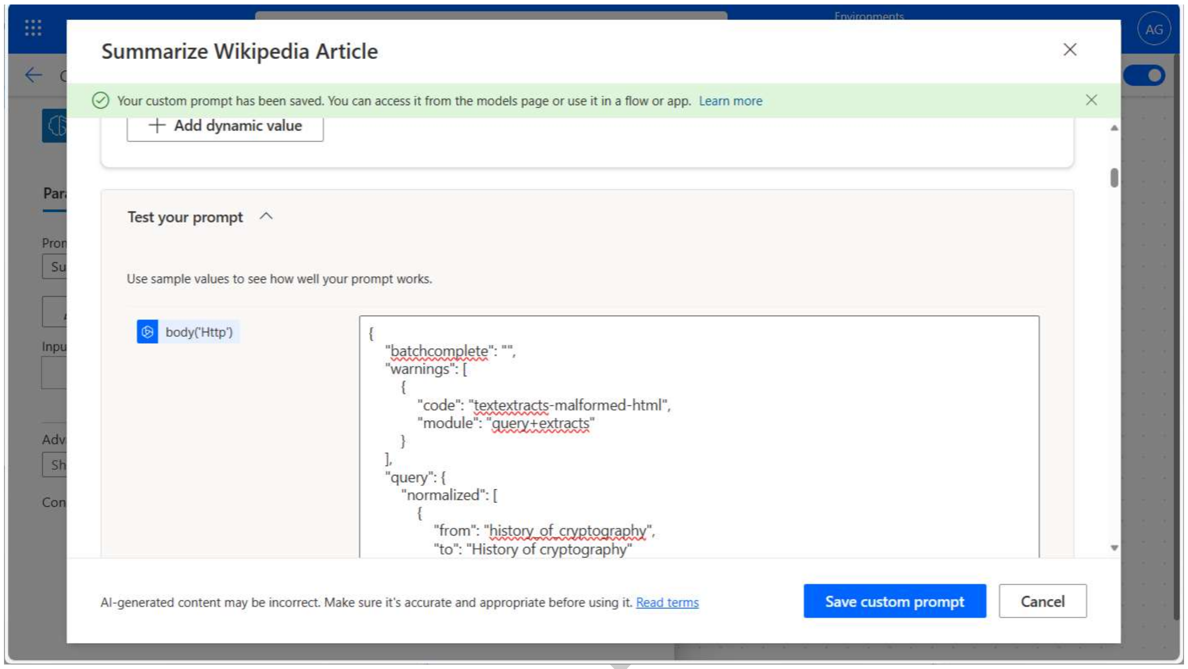
Write a summary of the input body('Http') .

Data is divided into sections by use of the H2 HTML tag. Each section should be summarized as a separate paragraph. For every H2 tag,

Figure 7.21 – Viewing the updated dynamic value token

2. To test the prompt, scroll to the bottom of the prompt customization area. Switch to the browser tab containing the Wikipedia API content, select it all, and copy it to your computer's buffer using *Ctrl* + *C*:

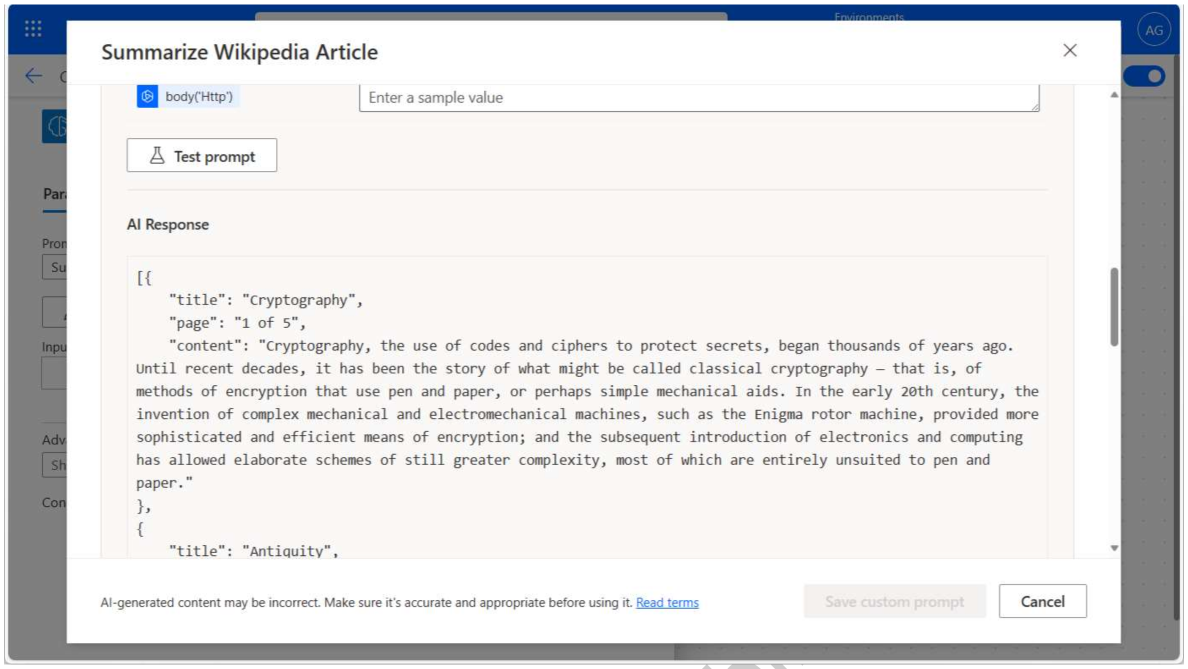
3. Switch back to the browser tab containing the customized AI prompt. In the **Test your prompt** area, paste the copied Wikipedia API output, as shown in *Figure 7.22*:


Summarize Wikipedia Article
Your custom prompt has been saved. You can access it from the models page or use it in a flow or app. Learn more
Add dynamic value
Test your prompt
Use sample values to see how well your prompt works.
body('Http')
AI-generated content may be incorrect. Make sure it's accurate and appropriate before using it. Read terms
Save custom prompt
Cancel

```
{
  "batchcomplete": "",
  "warnings": [
    {
      "code": "textextracts-malformed-html",
      "module": "query+extracts"
    }
  ],
  "query": {
    "normalized": [
      {
        "from": "history_of_cryptography",
        "to": "History of cryptography"
```

Figure 7.22 – Loading data into the Test your prompt area

4. Scroll to the bottom of the **Test your prompt** area and click **Test prompt**.
5. Wait while the AI Builder prompt generates a response. Review the response to make sure it adheres to your requirements:



Summarize Wikipedia Article

body('Http') Enter a sample value

Test prompt

AI Response

```
[{
    "title": "Cryptography",
    "page": "1 of 5",
    "content": "Cryptography, the use of codes and ciphers to protect secrets, began thousands of years ago.
Until recent decades, it has been the story of what might be called classical cryptography — that is, of
methods of encryption that use pen and paper, or perhaps simple mechanical aids. In the early 20th century, the
invention of complex mechanical and electromechanical machines, such as the Enigma rotor machine, provided more
sophisticated and efficient means of encryption; and the subsequent introduction of electronics and computing
has allowed elaborate schemes of still greater complexity, most of which are entirely unsuited to pen and
paper."
},
{
    "title": "Antiquity",
```

AI-generated content may be incorrect. Make sure it's accurate and appropriate before using it. Read terms

Save custom prompt Cancel

Figure 7.23 – Reviewing the AI-generated content

6. Click inside the **Input Body('Http')** field, select the dynamic content icon, and then select the **Body** object of the **Parse JSON** action:

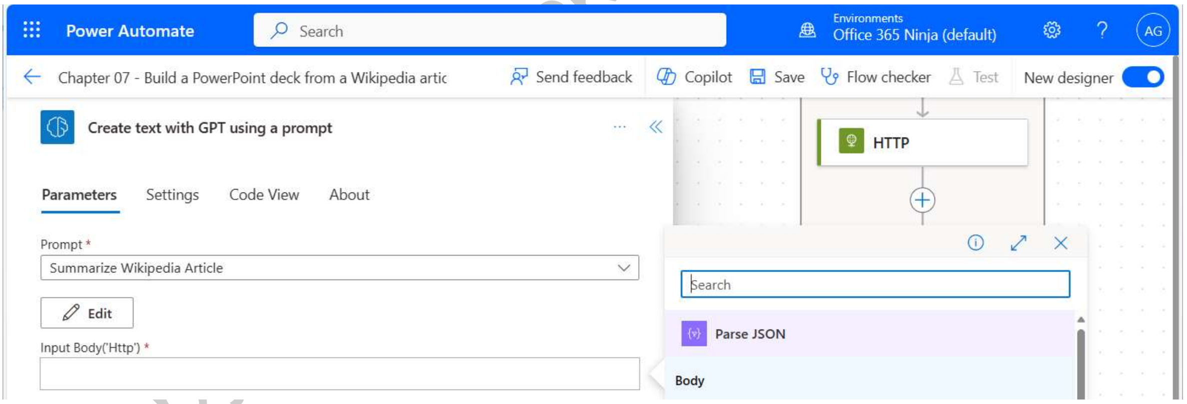
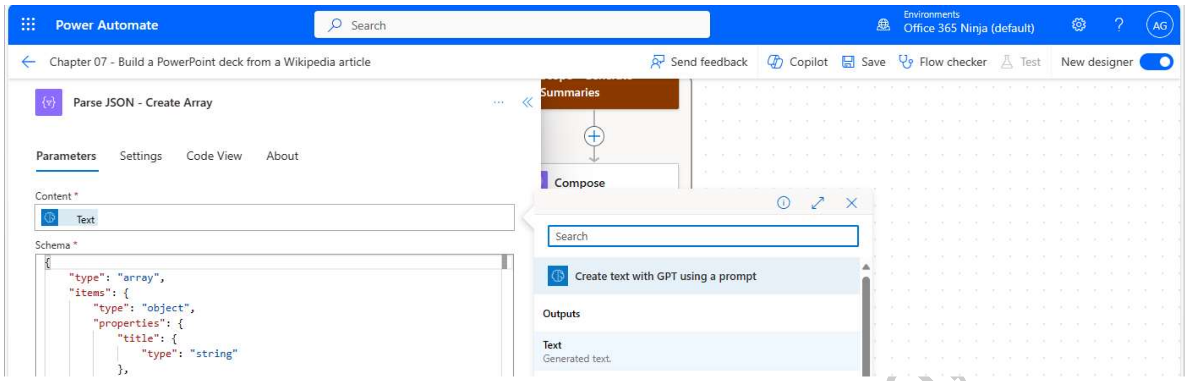


Figure 7.24 – Adding the Body object to the Input Body('Http') field

7. Click the + icon after the **Create text with GPT using a prompt** action and select **Add an action**.
8. Select the **Parse JSON** action. This action will convert the GPT output into an array object that can be iterated through later.
9. On the **Parse JSON** action flyout, rename the action **Parse JSON – Create Array**.

10. On the **Parameters** tab, click inside the **Content** field. Select the dynamic content icon and then choose the **Text** token under the **Create text with GPT using a prompt** action. See *Figure 7.25*:



```
{
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "title": {
                "type": "string"
            },
```

Figure 7.25 – Adding the text dynamic content token

11. In the **Schema** area, copy and paste the following content:

```
{
    "type": "array",
    "items": {
        "type": "object",
        "properties": {
            "title": {
                "type": "string"
            },
            "page": {
                "type": "string"
            },
            "content": {
                "type": "string"
            }
        },
        "required": [
            "title",
            "page",
            "content"
        ]
    }
}
```

This schema definition can be obtained by running the flow up to this point, copying the output, and then pasting it in the **Use sample payload to generate schema** popup on the **Parse JSON** flyout.

12. Click **Save.** Do not close the Power Automate flow canvas.

So far, you've created a flow that can take a Wikipedia article URL and summarize each H2 element into its own JSON object. Next, we'll work on sending that data to the PowerPoint template.

Creating the Generate Slides scope

In this section, we'll take the JSON object array and turn it into a PowerPoint slide deck. Follow these steps to process the AI-generated content:

1. Scroll to the bottom of the flow on the canvas and select the + icon outside the **Scope – Generate Summaries** control:

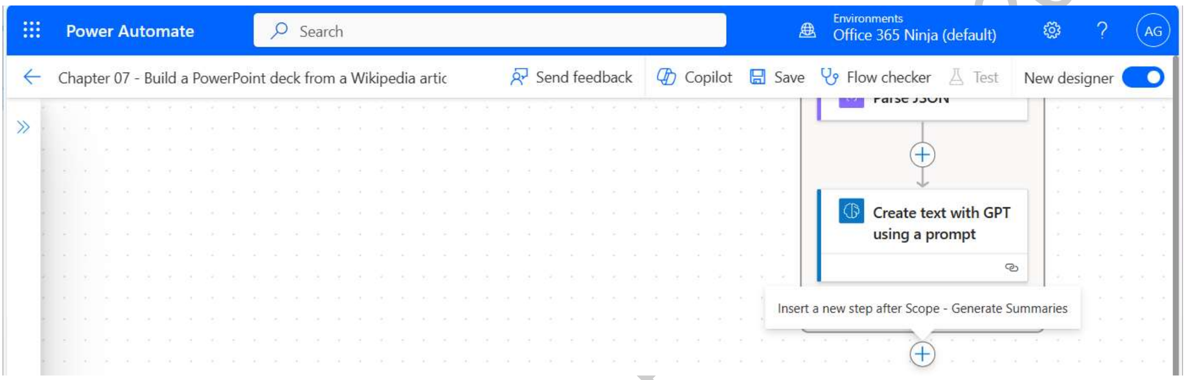

Power Automate
Search
Environments
Office 365 Ninja (default)
AG
Chapter 07 - Build a PowerPoint deck from a Wikipedia artic
Send feedback
Copilot
Save
Flow checker
Test
New designer
Parse JSON
Create text with GPT using a prompt
Insert a new step after Scope - Generate Summaries

Figure 7.25 – Adding a new step

2. Add a new scope control and rename it `Scope - Generate Slides`.
3. Click the + icon and choose **Add an action** inside the **Scope - Generate Slides** card.
4. Choose the **Get file content using path** action for either SharePoint Online or OneDrive for Business, depending on where you saved your template file:

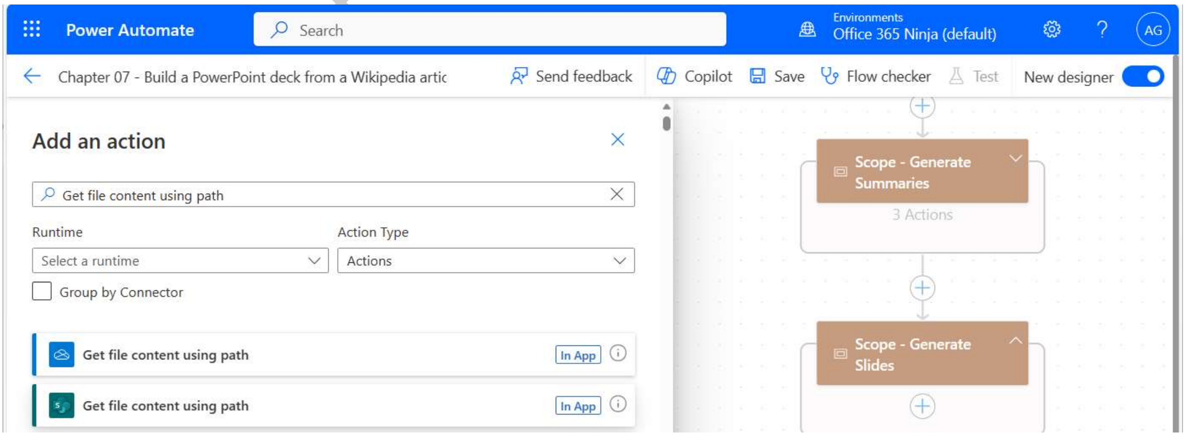
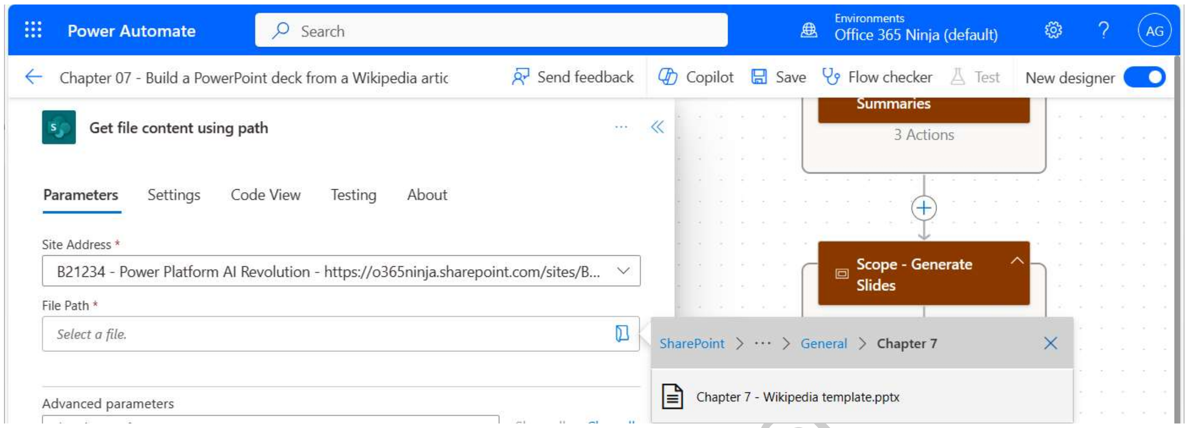


Figure 7.26 – Adding the Get file content using path action

5. On the **Get file content using path** flyout, locate the file. If you're using the OneDrive for Business action, select the **File Path** field using the folder browser. If you're using the SharePoint Online action, select the **Site Address** value where the file is located and then use the folder browser in the **File Path** field to select the file:

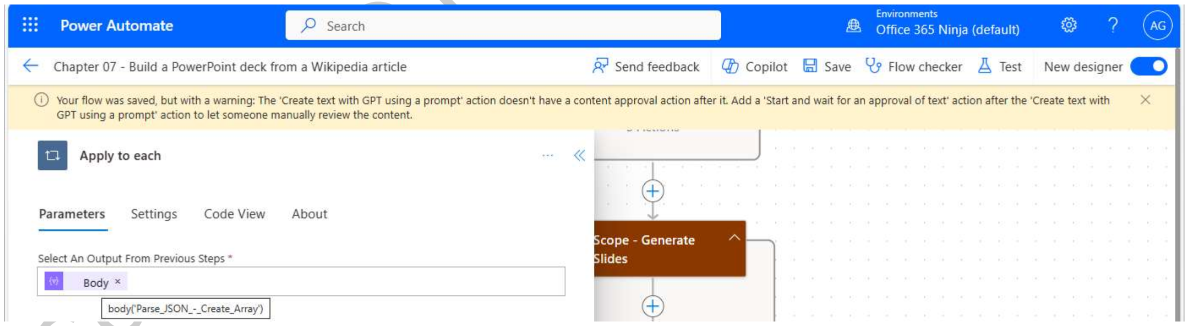


Site Address *
B21234 - Power Platform AI Revolution - https://o365ninja.sharepoint.com/sites/B...
File Path *
Select a file.

SharePoint > ··· > General > Chapter 7
Chapter 7 - Wikipedia template.pptx

Figure 7.27 – Selecting the PowerPoint template file

6. Inside the **Scope – Generate Slides** card, click the + icon and select **Add an action**.
7. Add the **Apply to each** control.
8. On the **Apply to each** control flyout, select the **Parameters** tab.
9. In the **Select an Output From Previous Steps** field, add the **Body** output from the **Parse JSON – Create Array** action you created earlier:



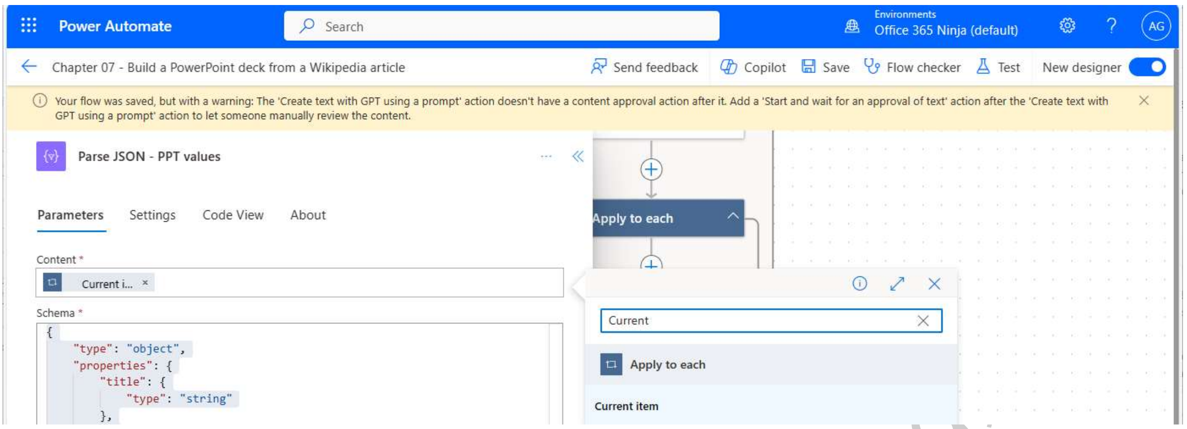
Your flow was saved, but with a warning: The 'Create text with GPT using a prompt' action doesn't have a content approval action after it. Add a 'Start and wait for an approval of text' action after the 'Create text with GPT using a prompt' action to let someone manually review the content.

Select An Output From Previous Steps *
Body
body('Parse_JSON_-_Create_Array')

Figure 7.28 – Adding the Body output from the Parse JSON – Create Array action

10. Inside the **Apply to each** control card, click + and select **Add an action**.
11. Select the **Parse JSON** action.
12. Rename the **Parse JSON** action `Parse JSON – PPT Values`.
13. On the **Parameters** tab of the **Parse JSON – PPT values** flyout, click inside the **Content** field.

14. Select the **Current item** dynamic content token. See *Figure 7.29*:



Power Automate | Search | Environments Office 365 Ninja (default) | AG

Chapter 07 - Build a PowerPoint deck from a Wikipedia article | Send feedback | Copilot | Save | Flow checker | Test | New designer

Your flow was saved, but with a warning: The 'Create text with GPT using a prompt' action doesn't have a content approval action after it. Add a 'Start and wait for an approval of text' action after the 'Create text with GPT using a prompt' action to let someone manually review the content.

Parse JSON - PPT values

Parameters | Settings | Code View | About

Content *
Current i... ×

Schema *
```
{
    "type": "object",
    "properties": {
        "title": {
            "type": "string"
        },
```

Apply to each

Current

Apply to each

Current item

Figure 7.29 – Selecting the Current item dynamic content token

15. In the **Schema** area, copy and paste the following value:

```
{
    "type": "object",
    "properties": {
        "title": {
            "type": "string"
        },
        "page": {
            "type": "string"
        },
        "content": {
            "type": "string"
        }
    }
}
```

This content can be derived by running the flow up to this point, viewing the **Run history** area, selecting the output of the **Create text with GPT using a prompt** action, and then pasting it into the **Use sample payload to generate schema** popup.

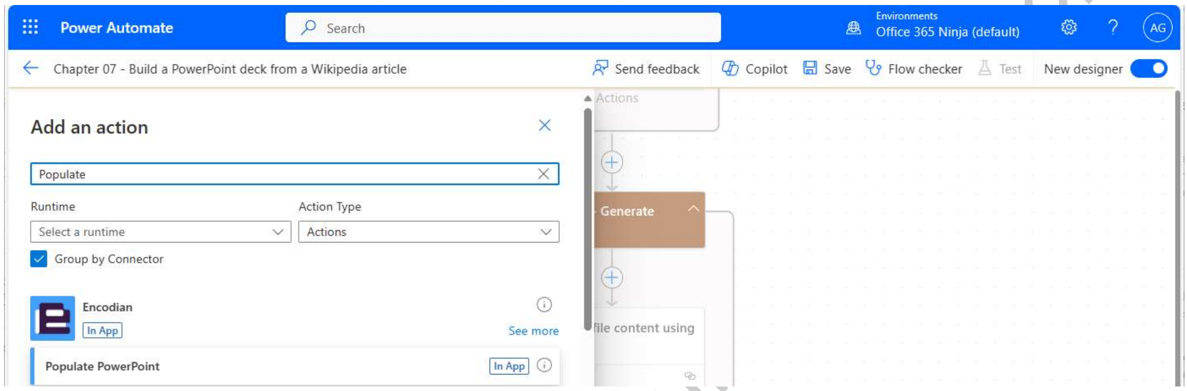
16. Click **Save**. Do not close the Power Automate flow canvas.

Next, we'll start sending data to the Encodian connector.

Working with Encodian Flowr

In this section, you'll start interfacing with the Flowr connector. It can be tricky to get it to work, so do the following:

1. Inside the **Apply to each** card, click the + icon following **Parse JSON – PPT values** and select **Add an action**.
2. Add the **Populate PowerPoint** action:



Add an action

Populate

Runtime: Select a runtime

Action Type: Actions

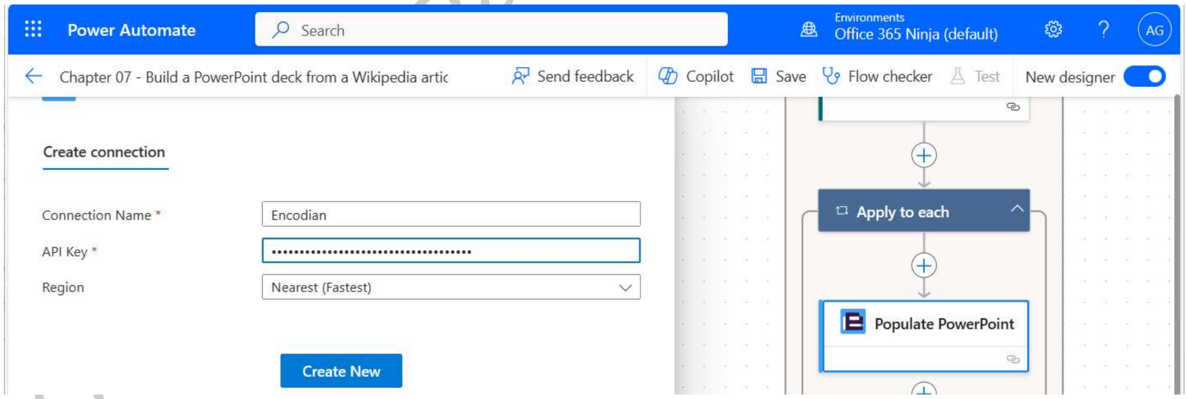
✓ Group by Connector

Encodian In App — See more

Populate PowerPoint In App

Figure 7.30 – Adding the Populate PowerPoint action

3. On the **Populate PowerPoint** flyout, enter a **Connection Name** value for the Encodian connection and add your **API Key**. Click **Create New** to finish the setup:



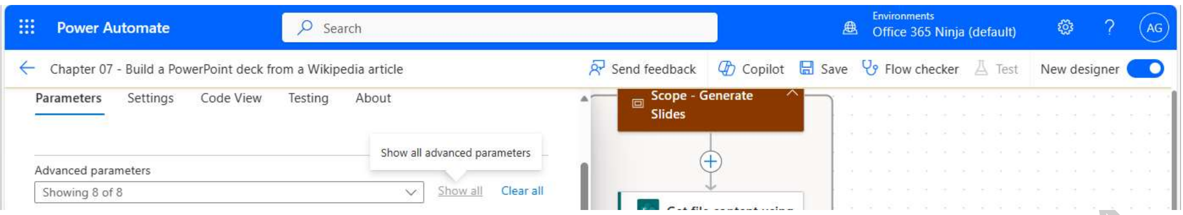
Create connection

Connection Name *	Encodian
API Key *	••••••••••••••••••••••••••••••••••••
Region	Nearest (Fastest)

Create New

Figure 7.31 – Configuring the Encodian connection

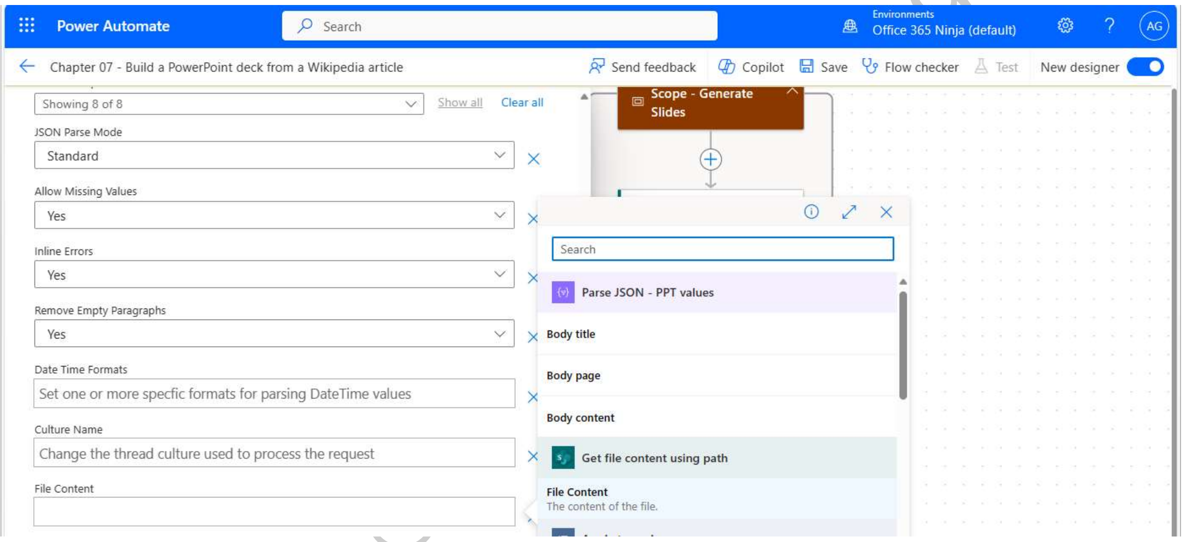
4. On the **Populate PowerPoint** flyout, click **Show all** next to the **Advanced parameters** dropdown:



Advanced parameters
Showing 8 of 8

Figure 7.32 – Expanding all of the available parameters

5. Click inside the **File Content** field, select the dynamic content icon, and then choose the **File Content** token for the configured storage location (either **SharePoint Online** or **OneDrive for Business**):



Showing 8 of 8

JSON Parse Mode
Standard

Allow Missing Values
Yes

Inline Errors
Yes

Remove Empty Paragraphs
Yes

Date Time Formats
Set one or more specfic formats for parsing DateTime values

Culture Name
Change the thread culture used to process the request

File Content

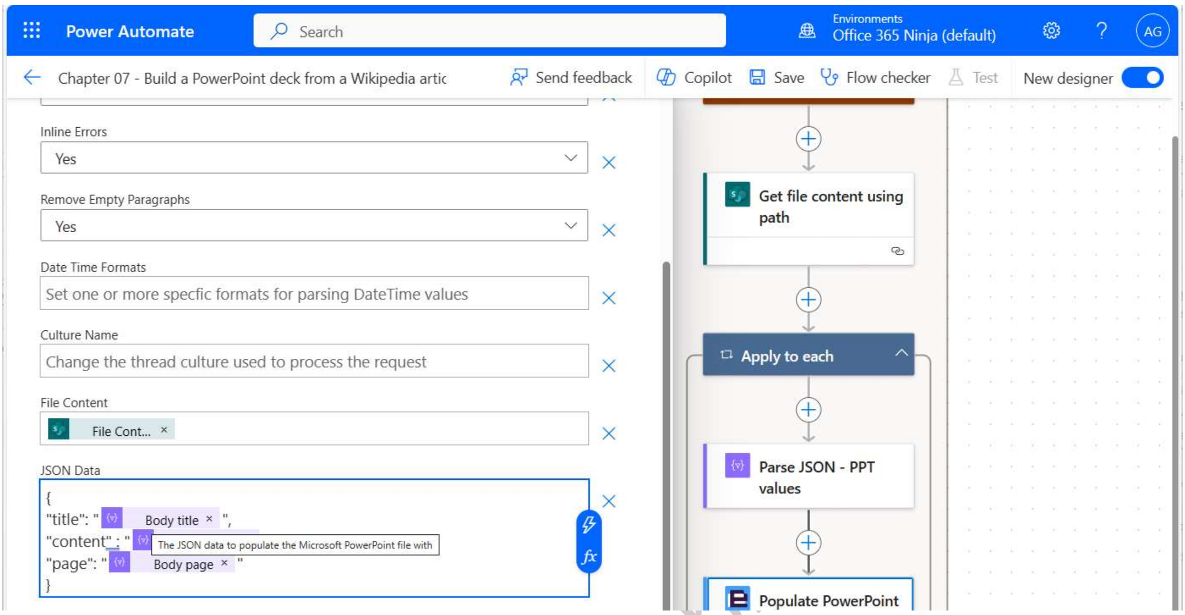
File Content
The content of the file.

Figure 7.33 – Adding the File Content token

6. In the **JSON Data** field, define the JSON structure that the connector will use to replace the content tokens in your template file. For example, this exercise utilizes the **title**, **content**, and **page** tokens. The JSON definition should look similar to the following sample:

```
{
"title": "@{body('Parse_JSON_-_PPT_values')?['title']}",
"content" : "@{body('Parse_JSON_-_PPT_values')?['content']}",
"page": "@{body('Parse_JSON_-_PPT_values')?['page']}"
}
```

It should look like this:



Inline Errors: Yes

Remove Empty Paragraphs: Yes

Date Time Formats: Set one or more specfic formats for parsing DateTime values

Culture Name: Change the thread culture used to process the request

File Content: File Cont...

JSON Data:

```
{
"title": " Body title ",
"content": " The JSON data to populate the Microsoft PowerPoint file with
"page": " Body page "
}
```

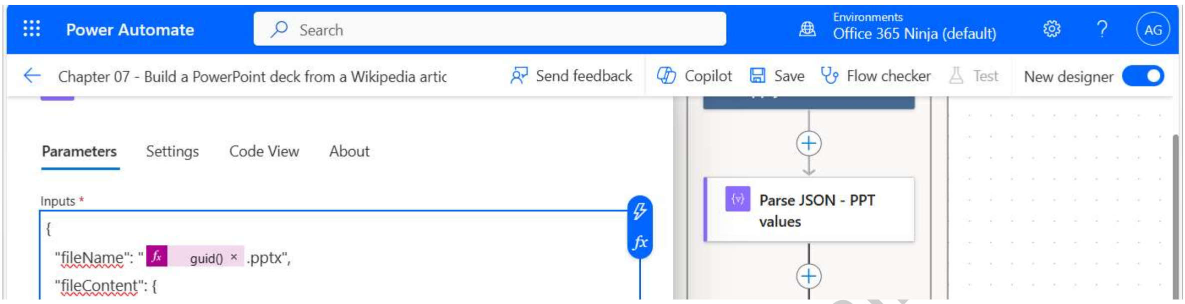
Figure 7.34 – Configuring the JSON data property

You can type the definitions (such as `"title" : " "`) and then choose the corresponding parameter values from the dynamic content token list. The outputs should be selected from the **Parse JSON – PPT Values** action. Alternatively, if you have renamed the actions to follow the exercise, you can copy and paste the entire JSON definition.

1. Inside the **Apply to each** card, click the + icon after the **Populate PowerPoint** action and select **Add an action**.
2. Select the **Compose** action.
3. On the **Compose** flyout, rename the action `Compose PowerPoint Slides`.
4. On the **Parameters** tab of the **Compose PowerPoint slides** flyout, click inside the **Inputs** field and paste the following content:

```
{
"fileName": ".pptx",
"fileContent": {
"$content-type": "application/vnd.openxmlformats-officedocument.
presentationml.presentation",
"$content":
}
}
```

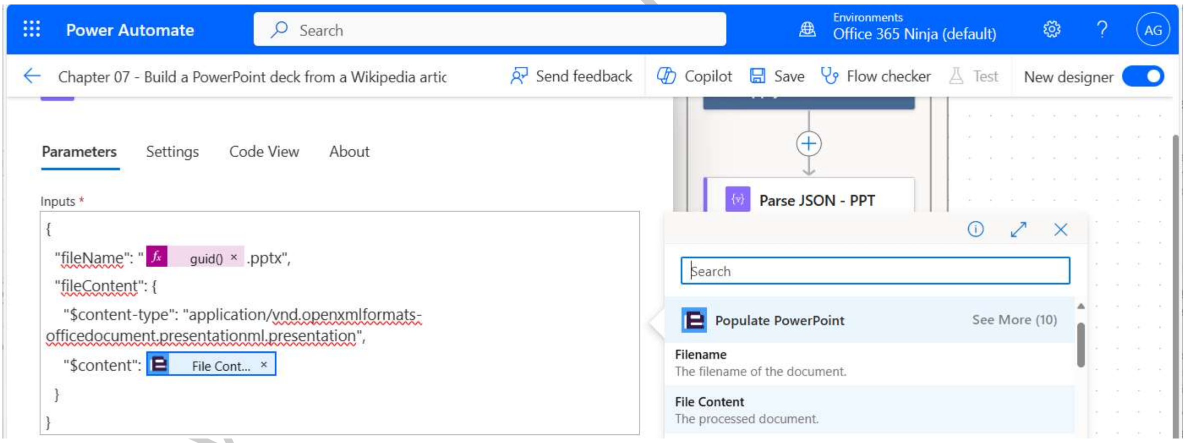
5. In the **Inputs** area, place the cursor before the `.` character on the **fileName** definition. Click the function icon and add the **guid()** function. The **Merge Presentation** action requires a **fileName** property later – and **guid()** is the simplest function to allow you to create random filenames that won't conflict with existing files anywhere. See *Figure 7.35* to refer to placement. Note that both the key and value parameters are enclosed with double quotation marks (`"`). Without the quotes, you will receive a JSON format error:



```
{
 "fileName": " guid() .pptx",
 "fileContent": {
```

Figure 7.35 – Adding the guid() function to the filename

6. In the **Inputs** area, place the cursor after `"$content":` and click the dynamic content icon. Choose the **File Content** dynamic content token. See *Figure 7.36*:

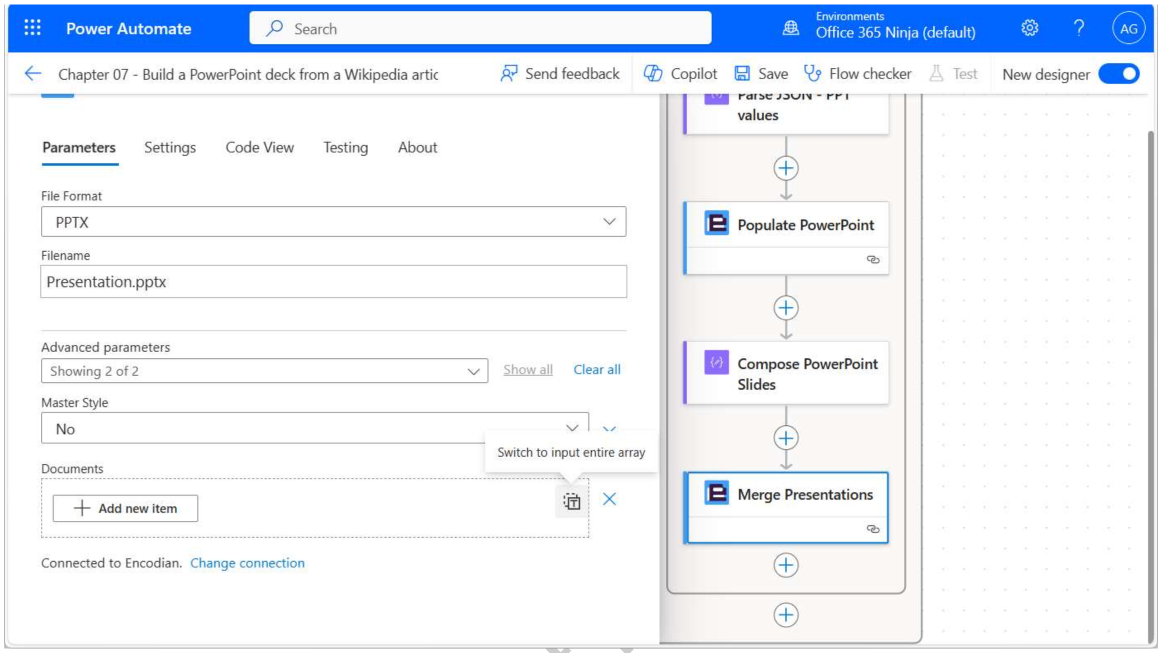


```
{
 "fileName": " guid() .pptx",
 "fileContent": {
  "$content-type": "application/vnd.openxmlformats-officedocument.presentationml.presentation",
  "$content": File Cont...
 }
}
```

Figure 7.36 – Adding the File Content dynamic content token

7. Inside the **Apply each card**, click the + icon below the **Compose PowerPoint Slides** action and select **Add an action**.
8. Choose the **Merge Presentations** action.
9. Enter a **Filename** value if desired (the default filename is `Presentation.pptx`).
10. Under **Advanced parameters**, select **Show all**.

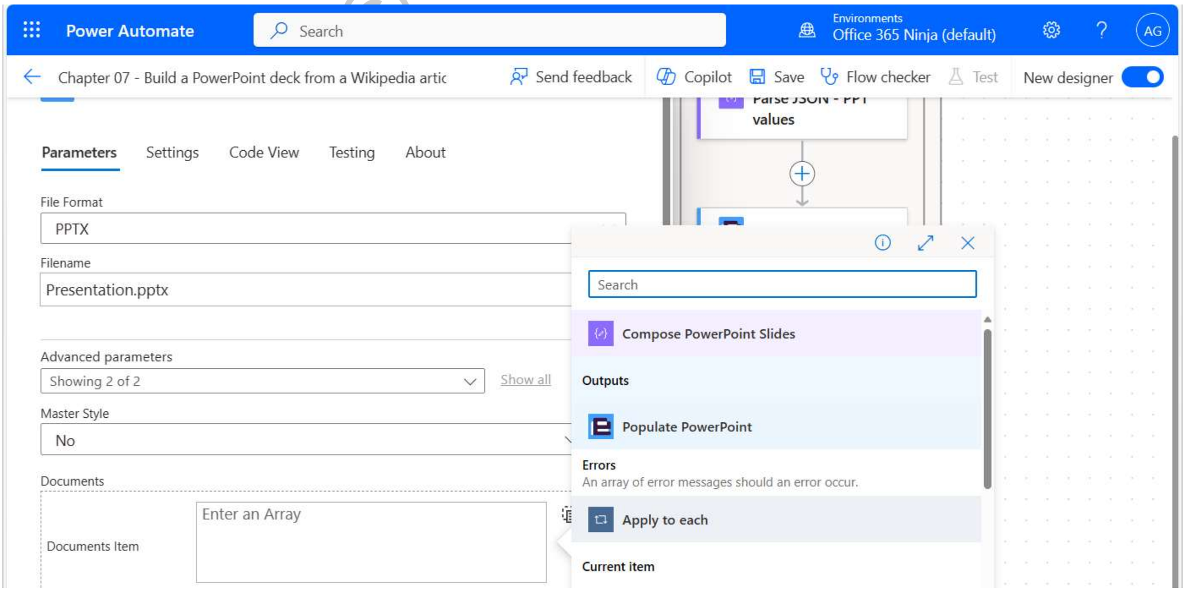
11. In the **Documents** section, click the **T** icon (**Switch to input entire array**). This will allow you to select the output of the **Compose PowerPoint Slides** action – the collection of files that were assigned `<guid>.PPTX` names:



File Format: PPTX
Filename: Presentation.pptx
Advanced parameters: Showing 2 of 2
Master Style: No
Documents
Connected to Encodian. Change connection

Figure 7.37 – Changing the input type

12. Click inside the **Documents Item** field and select the dynamic content token icon. Under **Compose PowerPoint Slides**, choose **Outputs**:



File Format: PPTX
Filename: Presentation.pptx
Advanced parameters: Showing 2 of 2
Master Style: No
Documents
Documents Item: Enter an Array

Figure 7.38 – Updating the Document Item field

13. Collapse the **Merge Presentations** flyout.
14. Drag the **Merge Presentations** card outside of the **Apply to each** and **Scope – Generate Slides** cards and place it on the + icon below the scope card. See *Figure 7.39*:

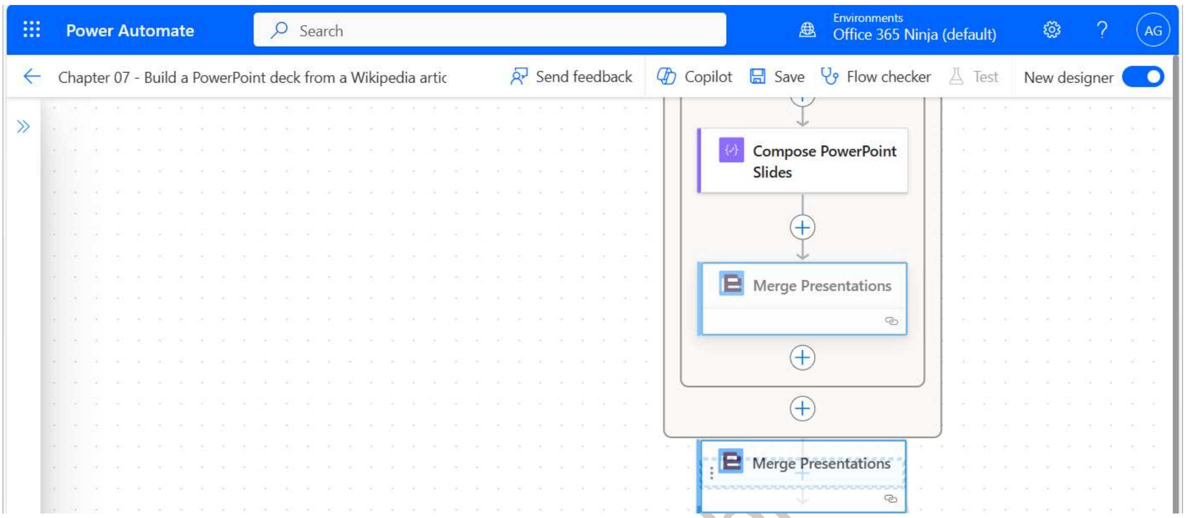
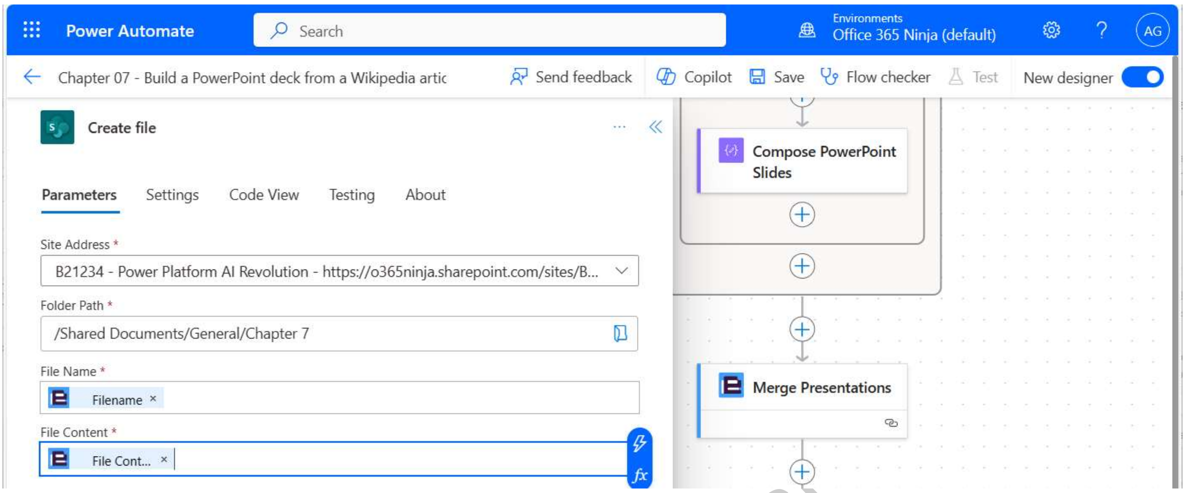


Figure 7.39 – Moving the Merge Presentations card

This order of steps is necessary to ensure that the **Compose PowerPoint Slides** output can be chosen from the dynamic content menu.

15. Click the + icon after **Merge Presentations** and then select **Add a step** to add it to the end of the flow.
16. Select the **Create file** action (either SharePoint or OneDrive for Business), depending on where you want the output. It does not have to be in the same location where the template file is stored.
17. On the **Create file** flyout, select the location (either a **Folder Path** location if you chose OneDrive for Business or a **Site Address** and **Folder Path** location if you chose SharePoint Online).
18. In the **File Name** field, add the **Filename** dynamic content token from the **Merge Presentations** action.

19. In the **File Content** field, add the **File Content** dynamic content token from the **Merge Presentations** action, as shown in *Figure 7.40*:



Site Address *
B21234 - Power Platform AI Revolution - https://o365ninja.sharepoint.com/sites/B...

Folder Path *
/Shared Documents/General/Chapter 7

File Name *
Filename

File Content *
File Cont...

Figure 7.40 – Selecting the File Content dynamic content token

20. Click **Save**.

Click the **Flow checker** area to ensure your flow contains no obvious errors. When you're finished with that, it's time to test the flow!

Testing the flow

To test the flow, follow these steps:

1. Open a new browser tab. Navigate to Wikipedia (`https://en.wikipedia.org`) and search for an article of your choosing. For this example, you might try searching for `Sailing` or `History of cryptography`.
2. After the article loads, click the URL bar and copy the URL value using *Ctrl* + *C*.
3. Switch back to the browser tab that's running the Power Automate canvas.
4. Click the **beaker** icon labeled **Test**:

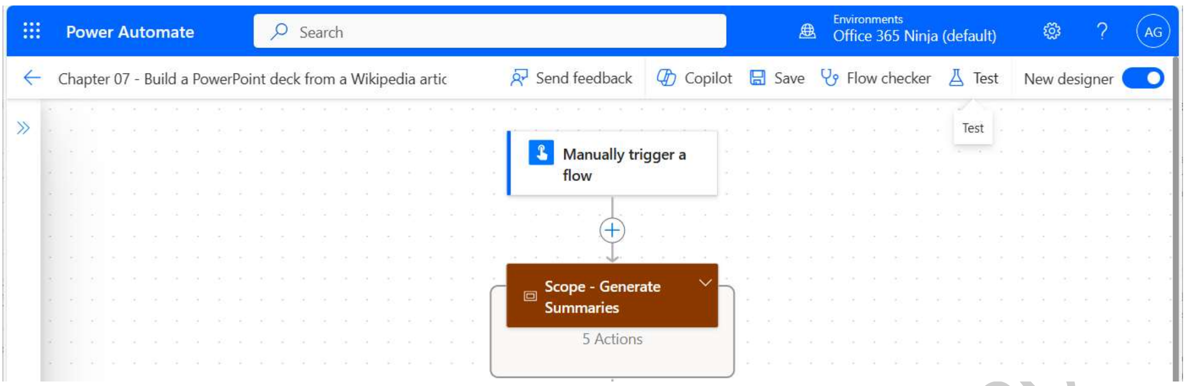
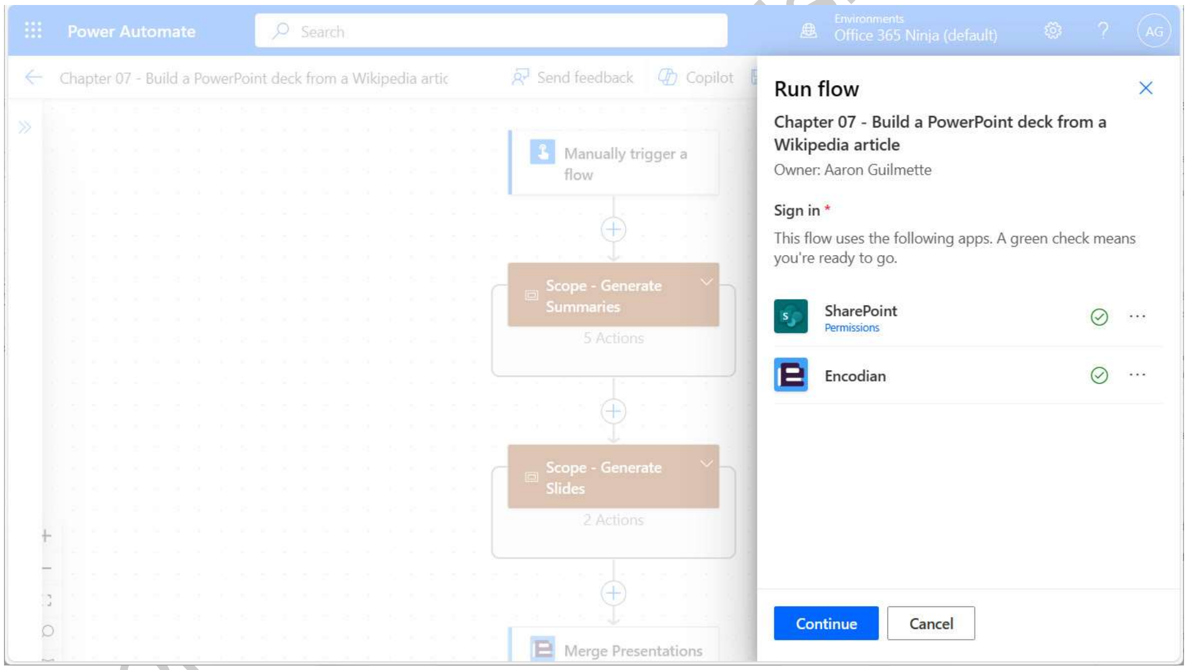


Figure 7.41 – Preparing to test the flow

5. On the **Test Flow** flyout, select the **Manually** radio button and click **Test**.
6. If prompted, confirm any permissions and click **Continue**:


Run flow

Chapter 07 - Build a PowerPoint deck from a Wikipedia article

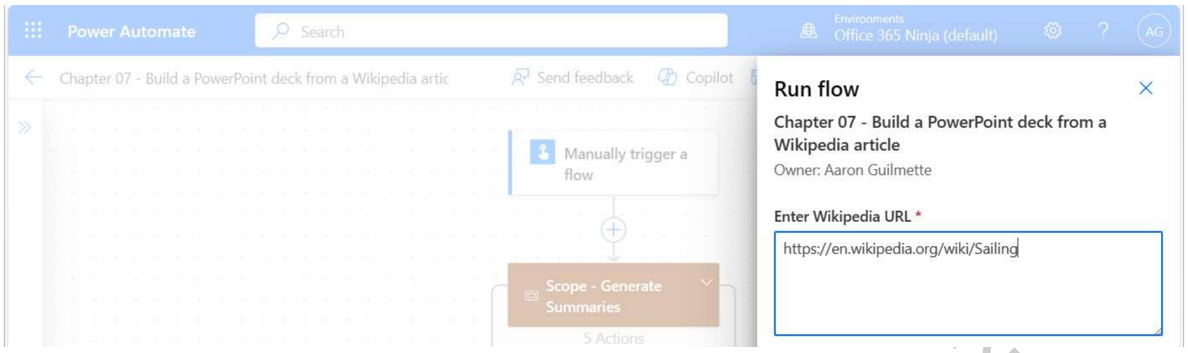
Owner: Aaron Guilmette

Sign in *

This flow uses the following apps. A green check means you're ready to go.

Figure 7.42 – Confirming permissions for the flow

7. Paste the copied Wikipedia URL into the prompt area and click **Run flow**:


Power Automate
Search
Environments
Office 365 Ninja (default)
AG
Chapter 07 - Build a PowerPoint deck from a Wikipedia artic
Send feedback
Copilot
Manually trigger a flow
Scope - Generate Summaries
5 Actions
Run flow

Chapter 07 - Build a PowerPoint deck from a Wikipedia article

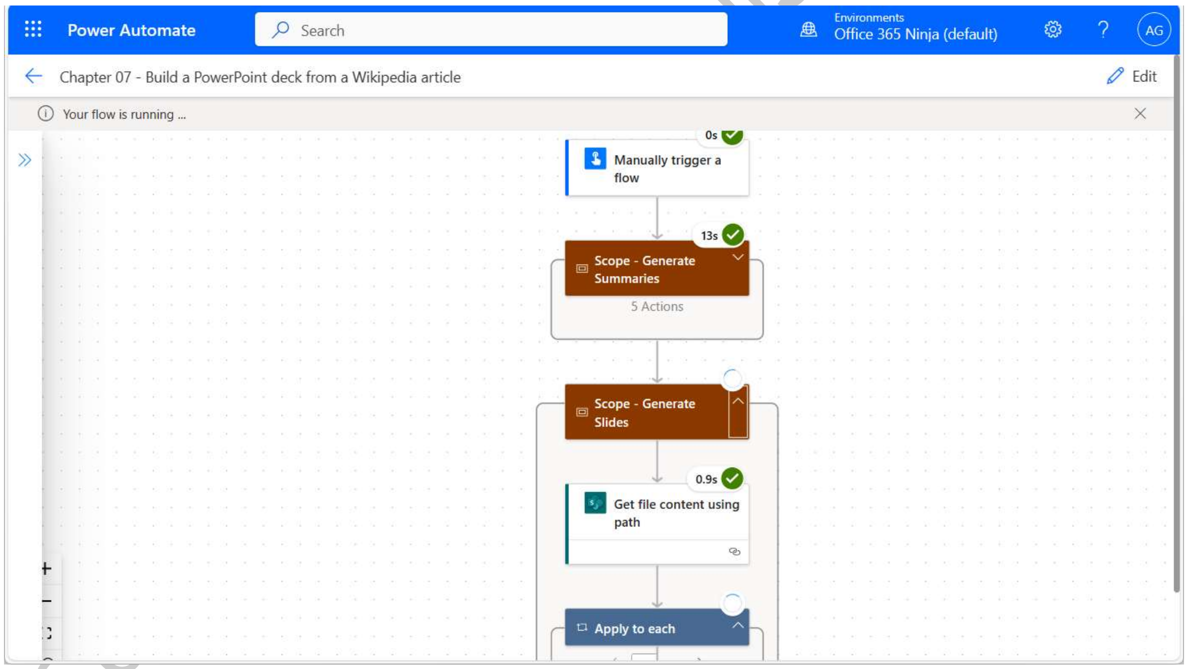
Owner: Aaron Guilmette

Enter Wikipedia URL *

https://en.wikipedia.org/wiki/Sailing

Figure 7.43 – Entering a URL

8. Click **Done** to return to the canvas and watch the flow run execute.
9. Wait while the flow executes:

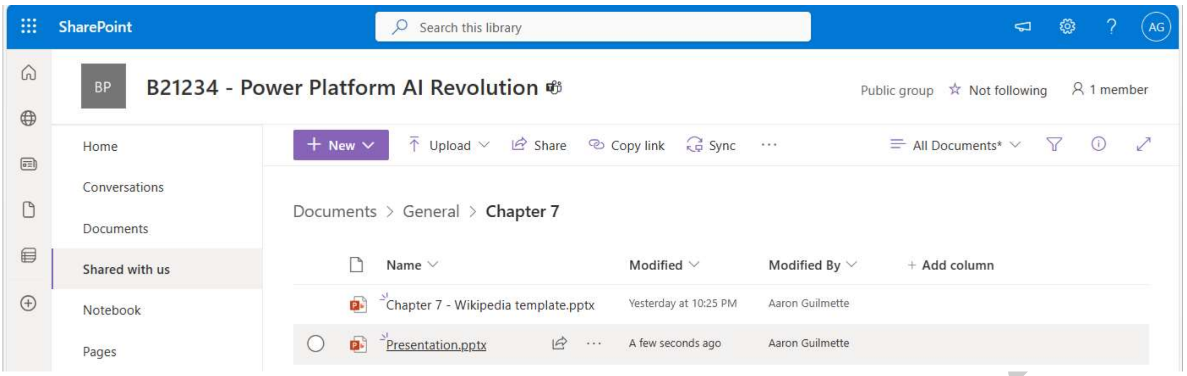

Power Automate
Search
Environments
Office 365 Ninja (default)
AG
Chapter 07 - Build a PowerPoint deck from a Wikipedia article
Edit
0s
Manually trigger a flow
13s
Scope - Generate Summaries
5 Actions
Scope - Generate Slides
0.9s
Get file content using path
Apply to each

Your flow is running ...

Figure 7.44 – Waiting for the flow to complete

10. After the flow has been completed, navigate to the location where you specified the completed presentation should be saved.

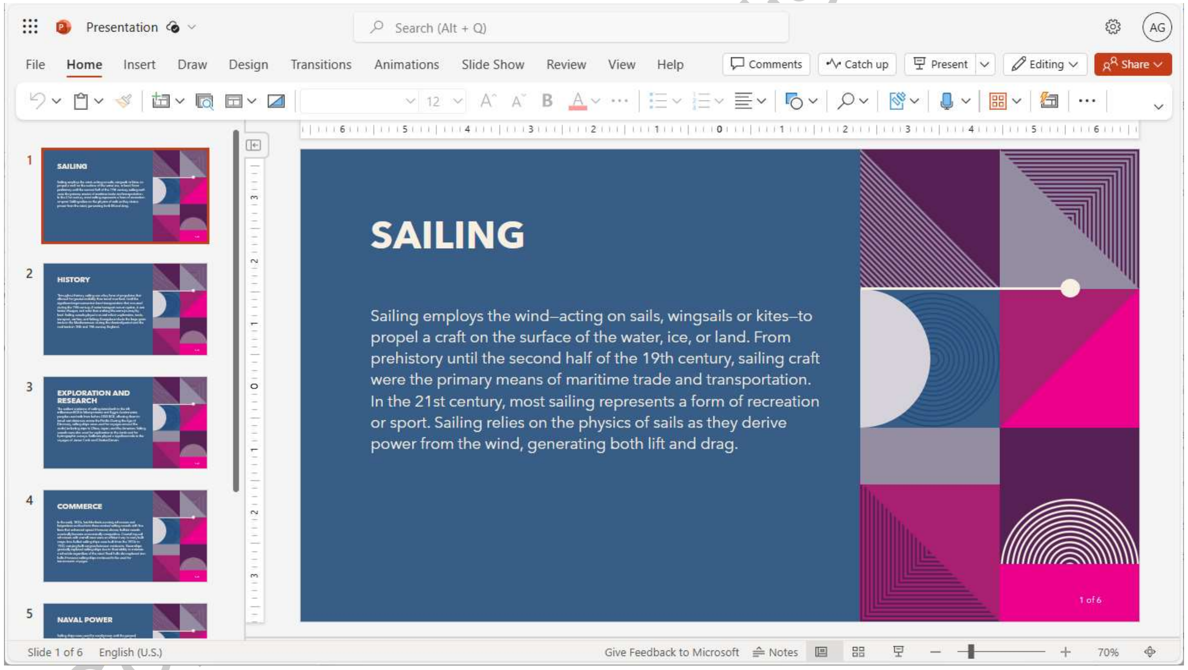
11. Open the presentation:



Name	Modified	Modified By
Chapter 7 - Wikipedia template.pptx	Yesterday at 10:25 PM	Aaron Guilmette
Presentation.pptx	A few seconds ago	Aaron Guilmette

Figure 7.45 – Launching the presentation

12. Review the completed presentation, taking note of how the content tokens in the slide were replaced with the corresponding JSON values that were passed to the Encodian Flowr connector:



SAILING

Sailing employs the wind–acting on sails, wingsails or kites–to propel a craft on the surface of the water, ice, or land. From prehistory until the second half of the 19th century, sailing craft were the primary means of maritime trade and transportation. In the 21st century, most sailing represents a form of recreation or sport. Sailing relies on the physics of sails as they derive power from the wind, generating both lift and drag.

Figure 7.46 – Reviewing the completed presentation

That's it! You've now created an entire PowerPoint presentation just by scraping website content.

Further exploration

Think about all the ways you can further expand, reuse, or repurpose this type of solution to build slide decks from different types of content sources:

- Quarterly reports
- Executive summaries
- Sales reports

By combining the power of Generative AI with automation for common Microsoft 365 Apps document creation tools, you can create presentations in a snap!

Summary

This chapter demonstrated some of the incredible capabilities that are available by harnessing the power of Generative AI. By using the native AI Builder ChatGPT connector, you were able to ingest web content from the world's largest online encyclopedia, summarize it, and turn it into a PowerPoint presentation (with the help of the Encodian Flowr connector).

Using the skills you learned here, you could adapt this flow to pull together reports and presentations from content stored in SharePoint, OneDrive, or other websites and documents.

In the next chapter, we'll learn about using the AI Builder ID reader model.