

## **Storybook GPT Documentation**

### **Storybook POC Continuation with Chromatic and Storybook GPT**

The University of Texas at Dallas - Spring 2024

#### **Group 3**

Lillie McMaster (Lead) | Lauren Nguyenphu | Alina Khan

Hamdiya Abdulhafiz | Iman Sheriff | Timothy Naumov

#### **Sponsors - ARGO**

Ponchai Reainthong | Kevin Roa | Raisa Gonzalez

## Table of Contents

1. Train to Standardize Documentation	4
2. Merged Component File Problems	7
Introduction	7
Findings/ Thoughts	7
Conclusion	8
Rename the file to include the name of the component. If the combined files have multiple components, include the names of each component on the file name. While this may seem like a pain, it is what works best with the StorybookGPT. Plus, this is only a problem during training. Daily use of this GPT would not require daily maintenance.	8
3. MUI Theming Development	9
Introduction	9
Findings	9
Conclusion	10
4. API Research	11
Introduction	11
Findings/ Thoughts	11
Conclusion	11
5. Default Prompts	12
Introduction	12
Findings/ Thoughts	12
Conclusion	12
References	13
Appendix A: Bug Experiences	14
List of Figures	
Figure 3.1: Example of Dark theming	9
Figure 3.2: Theming Toolbar.	10

## 1. Train to Standardize Documentation

In the following text, we show how we created GPT to create standardized documentation for a given component.

Me: Can I create a routine that, given a stories file and a typescript of that component, you create standardized documentation?

GPT Builder: I've updated my capabilities to include creating standardized documentation based on your stories file and the TypeScript file of that component. How would you like to proceed with the documentation for your TextField component?

Me: Include the name of the story. Include a brief description of each of the props that the component can use.

GPT Builder: Based on the files for the TextField component and its stories, here's a draft of the standardized documentation including the story name and a brief description of each prop:

---

### TextField Component Documentation

Story Name: Inputs/TextField

This story demonstrates the usage of the `TextField` component with various configurations, showcased in two main stories: Default and CalendarIcon.

#### Props

- `variant ('default' | 'calendar-icon')`: Determines the style of the text field. The `'default'` variant is a standard text field, while the `'calendar-icon'` variant includes a calendar icon adornment.
  - `label (string)`: The label text displayed above the text field.
  - `maxLength (number)`: The maximum number of characters allowed in the text field.
  - `width (number)`: The width of the text field, specified in pixels.
  - `disabled (boolean)`: If `true`, the text field is disabled and cannot be interacted with.
  - `error (boolean)`: If `true`, the text field is displayed with an error state styling.
  - `helperText (string)`: Additional text displayed below the text field to provide context or guidance.
  - `margin ('dense' | 'none' | 'normal')`: The margin applied around the text field. Choices are `'dense'`, `'none'`, or `'normal'`.
  - `multiline (boolean)`: If `true`, the text field will accommodate multiple lines of input.
  - `rows (number)`: Specifies the number of rows to display when `multiline` is `true`. Determines the height of the text field.
-

Now test it. We uploaded files using the Fine-Tuning option of ChatGPT documentation. Fine-Tuning allows you to prompt the

\*\*\*\*\*Add more about fine Tuning [1.1]

Me: Uploaded TextField.tsx, TextField.Stories, Create standardized documentation

GPT: Based on the provided TypeScript component file `TextField.tsx` and its corresponding Storybook stories file `TextField.stories.tsx`, I have created documentation for the `TextField` component. This documentation details the component's props, usage, and story examples.

---

### ### `TextField` Component Documentation

#### #### Overview

The `TextField` component is a customizable text field that can be used in React applications. It is built on top of Material-UI's `TextField` component and supports various configurations, including an optional calendar icon for date inputs.

#### #### Props

- `variant?: 'default' | 'calendar-icon`: Determines the style of the text field. The 'calendar-icon' variant displays a calendar icon at the end of the text field.
- `label?: string`: Label text for the text field.
- `maxLength?: number`: Maximum number of characters that the text field can accept.
- `width?: number`: The width of the text field in pixels. Default is 201px.
- `disabled?: boolean`: If true, the text field will be disabled.
- `error?: boolean`: If true, the text field will be styled to indicate an error.
- `helperText?: string`: Helper text displayed below the text field.
- `margin?: 'dense' | 'none' | 'normal`: The margin of the text field.
- `multiline?: boolean`: If true, the text field will allow multiple lines of input.
- `rows?: string`: The number of rows to display when `multiline` is true.
- `required?: boolean`: If true, the text field will be required to fill out.
- `size?: 'small' | 'medium`: The size of the text field.
- `value?: any`: The input value of the text field.
- `onChange?: (e: any) => void`: Callback function that is called when the input value changes.
- `onClick?: () => void`: Callback function that is called when the calendar icon is clicked.

#### #### Usage

```
```jsx
```

```
import React from 'react';
```

```
import TextField from './TextField';
```

```
const MyComponent = () => (
```

```
  <TextField
```

```
    label="My Label"
```

```
    variant="default"
```

```
    onChange={handleChange}
```

```
    // Other props as needed
```

```
  />
```

```
);
```

...

### ### Storybook Stories

#### #### Default

The default text field with minimal configuration.

```
``jsx
export const Default = Template.bind({});
Default.args = {
  label: 'Label',
  variant: 'default'
};
...

```

#### #### Calendar Icon

A text field with a calendar icon, suitable for date inputs.

```
``jsx
export const CalendarIcon = Template.bind({});
CalendarIcon.args = {
  label: 'Label',
  variant: 'calendar-icon'
};
...

```

---

This documentation provides a comprehensive overview of how to use and customize the `TextField` component within your React applications.

## 2. Merged Component File Problems

UG3-56





Contributors Names: Lauren Nguyenphu

### Introduction

Creating StorybookGPT, it allows us to upload our components .tsx and stories.tsx files. However, it only allows a 20 file limit which presents a problem for our 30+ files. I began to combine all of the .tsx files into a Components.tsx and Components2.tsx file. After completing the merge of all of the components into fewer and larger files, these were uploaded into StorybookGPTs knowledge, but the next problem was exposed. With a large file titled Components.tsx and Components2.tsx, StorybookGPT is unable to easily parse through them to find components asked by the user.

### Findings/ Thoughts

After removing the individual .tsx and uploading the new and combined .tsx files, it was almost like StorybookGPT had been taken apart and was back at the start line. Removing the individual component files, the files it was originally being trained on, completely removed StorybookGPTs knowledge on those components. The original assumption was that because StorybookGPT had a 20 file limitation, combining the components into a larger file will allow us to input all of the components into StorybookGPT. However, after uploading, StorybookGPT is unable to recognize the need to search through the merged .tsx files when a particular component is being asked for. After multiple trials of training it to try and search each file in its knowledge, I noticed that it was searching through its knowledge based off of the file name.

<div><div> You</div><div>what are the props for pagination</div></div> <div><div> Storybook GPT</div><div><p>It seems there isn't a direct reference to a Pagination component in the files provided so far. However, I can assist in creating a standardized documentation and Storybook story for a Pagination component if you provide some details on how you envision this component working, including its props and any specific functionality you'd like it to have.</p><p>If the Pagination component is part of a larger library or if you have specific requirements for it (e.g., page numbers, next/previous buttons, boundary links), please share more details, and I'll be happy to help craft both the code snippet and its corresponding Storybook documentation.</p></div></div>	<div><div> You</div><div>what are the props for pagination</div></div> <div><div> Storybook GPT</div><div><p>The Pagination component does not appear directly in the provided Storybook stories ('RadioButton.stories.tsx', 'Dropdown.stories.tsx', 'CheckBox.stories.tsx', 'Box.stories.tsx', 'Badge.stories.tsx', 'Button.stories.tsx'). These files include components for Radio Buttons, Dropdowns, Checkboxes, Boxes, Badges, and Buttons, each with their own set of props as defined in their respective Storybook configurations.</p><p>Without a specific 'Pagination.stories.tsx' file or a direct reference to a Pagination component within the provided TypeScript files ('Components.tsx', 'Components2.tsx', etc.), I cannot provide the exact props for the Pagination component. Pagination components typically include props for managing the total number of items, items per page, the current page, and callbacks for page change events. If you have a specific file or component in mind that wasn't mentioned or need guidance on defining a pagination component in Storybook, please let me know how I can assist further!</p></div></div>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<p><b>You</b></p> <p>if a component being asked is not directly included in the story files, parse through Components.tsx and Components2.tsx first. If it is not found then just give the props from @mui/material for the component being asked</p> <p><b>GPT Builder</b></p> <p>Alright! If you ask about a component and it's not directly included in the Storybook story files, I'll first look through Components.tsx and Components2.tsx for details. If the component isn't found there, I'll then provide information on the props and their functionalities based on the @mui/material documentation for that component. This ensures you get comprehensive support, whether it's for components explicitly documented in the files you've shared or for standard components from popular libraries like Material-UI. Ready to dive into any component details or have another question? Just let me know!</p>	<p><b>You</b></p> <p>its referenced in Components2.tsx</p> <p><b>Storybook GPT</b></p> <p>The 'Pagination' component in 'Components2.tsx' directly uses the 'PaginationProps' from '@mui/material' for its props. This means the props for the Pagination component are those provided by the Material-UI Pagination component. Common props include:</p> <ul style="list-style-type: none"> <li>'count': The total number of pages.</li> <li>'page': The current page number.</li> <li>'onChange': Callback function fired when the page is changed. Signature: '(event: React.ChangeEvent&lt;unknown&gt;, page: number) =&gt; void'.</li> <li>'color': The color of the component. It supports "standard", "primary", and "secondary" as options.</li> <li>'variant': The variant to use. It supports "text" and "outlined".</li> <li>'shape': The shape of the pagination items. It supports "round" and "rounded".</li> <li>'size': The size of the pagination component. It supports "small", "medium", and "large".</li> </ul>
<p><b>You</b></p> <p>what are the props to left navigation</p> <p><b>Storybook GPT</b></p> <p>The "Left Navigation" component is not explicitly covered in the provided files or the standard @mui/material library components. However, a "Left Navigation" typically refers to a sidebar or a navigation drawer that is positioned on the left side of an application or website, used for navigating between different sections or pages.</p> <p>Since the "Left Navigation" component isn't directly mentioned in the files you've provided (Components.tsx and Components2.tsx), I'll outline a general set of props that such a component might have, based on common patterns in React development. For a custom "Left Navigation" component, the props might include:</p> <ul style="list-style-type: none"> <li>'items': An array of navigation items, where each item could be an object containing information like the title, icon, and target route of each navigation link.</li> <li>'onSelect': A function that gets called when a navigation item is selected, with the selected item's identifier or route as an argument.</li> </ul>	<p><b>You</b></p> <p>what are the props to left navigation</p> <p><b>Storybook GPT</b></p> <p>The props for the Left Navigation component are defined as follows:</p> <ul style="list-style-type: none"> <li>'listItems': An array of 'LNItem', which includes each item's 'label' and 'icon'. This defines the navigation items that will appear in the Left Navigation component.</li> <li>'onFooterClick': An optional callback function that gets called when the footer of the Left Navigation is clicked.</li> <li>'onChange': An optional callback function that gets called when a different navigation item is selected, receiving the index of the newly selected item.</li> </ul> <p>Additionally, there's an internal interface 'LNItem' for the items within the 'listItems' array, which consists of:</p> <ul style="list-style-type: none"> <li>'label': Optional. The text label for the navigation item.</li> </ul>

In the last two conversations shown in the images, the left image shows the response of StorybookGPT when the file name was just Components.tsx, and the right image was the response when the file name was changed to a format such as, componentNameX-componentNameY-componentNameZ.tsx, which is based on what components the file include. This showed a quick solution to allow StorybookGPT to find the correct component.

## Conclusion

Rename the file to include the name of the component. If the combined files have multiple components, include the names of each component on the file name. While this may seem like a pain, it is what works best with the StorybookGPT. Plus, this is only a problem during training. Daily use of this GPT would not require daily maintenance.

### 3. MUI Theming Development

Incorporating Custom Themes, Bundling Fonts, Implementing Dark Mode.

Contributors: Timothy Naumov, Alina Khan, Lillie McMaster

#### Introduction

In this section, we outline our development process of including custom themes into the Storybook repository. Custom themes allow for quicker, out of the box components to work with a specified combination of colors based on a companies branding. We also discuss how we bundled the fonts to optimize the font loading time, and how we implemented dark mode.

#### Findings

The first step in improving the integration of MUI Theming with Storybook [3.1] would be to bundle fonts for performance reasons. In the previous state of the repo, Storybook was pulling the fonts from the Google Fonts CDN. However, bundling fonts with Storybook is better for performance for 3 reasons: fonts will load faster because they are coming from the location of our app, the fonts will be available to load offline, and there are no more inconsistent snapshot tests because fonts will load instantly. Installing the fonts as dependencies was as easy as importing the font source and then importing the CSS files into .storybook/preview.js, which is the entry point for storybook.

After importing the default fonts, we moved forward with creating the Dark mode and other new themes. The themes should be created within the .../src/themes folder in the repository. Ideally, all the .theme files should contain only color objects.

Figure 3.1: Example of Dark theming

```
1  import { createTheme } from "@mui/material";
2  import { blueGrey, cyan, pink } from "@mui/material/colors";
3
4  export const darkTheme = createTheme({
5    palette: {
6      mode: "dark",
7      primary: {
8        main: pink["A200"],
9      },
10     secondary: {
11       main: cyan["A400"],
12     },
13     background: {
14       default: blueGrey["800"],
15       paper: blueGrey["700"],
16     },
17   },
18 });
```

Once multiple different themes have been created, a toolbar to the components is added in on the Storybook front end. This toolbar allows quick changes between themes. This feature provides time



saving capabilities to the developer and to the customer.

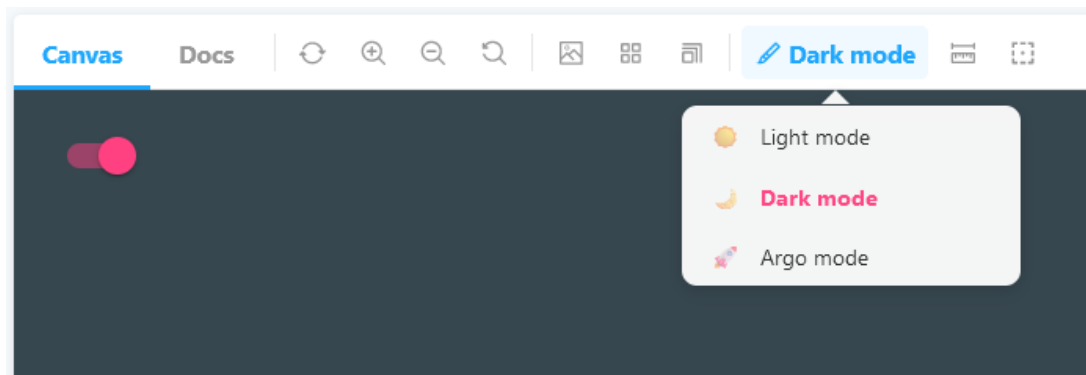


Figure 3.2: Theming Toolbar.

One issue we did run into during theming is the Previous Team's code implementation of components hard coded in the colors. This means that any theming changes within these components would not change the colors of the components. Our solution to this is to re-write the fundamental few of the components that needed to adhere to theming capabilities.

Design Add on and Figma Integration for Button and Switch: Following the Figma documentation to Embed Figma in Storybook, we enabled the Storybook to embed Figma files and prototypes in Storybook [3.2]. For implementing Figma into the two components, we installed the add on designs and registered the addons. Next, we added the Argo Figma URL to a parameter in the design.

## Conclusion

The process of implementing and working with the tooling for MUI theming development had proven to be a significant step forward in enhancing both design and development workflows. All of these theming tasks will streamline the process of creating branded, visually cohesive user interfaces.

## **4. API Research**

UG3-55

Contributors Names: Lauren Nguyenphu

### **Introduction**

AI-driven development tools like ActionsGPT, and design tools like Figma stand out for their potential to streamline workflows. GitHub Actions offer a platform for automating build, test, and deployment pipelines [4.1]. ActionsGPT can provide possibilities for defining custom actions through AI to enhance automation further [4.4]. Figma's API can facilitate integration between design specifications and development.

### **Findings/ Thoughts**

Exploring the capabilities of GitHub Actions reveals its flexibility and power in automating workflows. It can trigger workflows based on specific events, such as pull requests, streamlining the CI/CD process. This not only saves time, but also can reduce human error, ensuring that each code change is tested and deployed consistently. The Chromatic API also offers visual testing of UI components, capturing details down to the pixel.

ActionsGPT can allow developers to define custom actions that the GPT model can execute, opening possibilities for automating complex tasks.

Integrating Figma with development workflows can ensure that components align with design specifications, reducing the back and forth between designers and developers. Having the ability to automatically pull design tokens and specifications into development can ensure consistency and speed up implementation.

### **Conclusion**

The integration of tools like GitHub Actions, ActionsGPT, and Figma's API into the software development process represents a significant leap towards more efficient, error-free, and collaborative workflows. These tools not only save time but also ensure that the final product closely aligns with both the design vision and functional specifications.

## 5. Default Prompts

Task Name: UG3-72

Contributors Names: Iman Sheriff

### Introduction

The Default Prompts for the StorybookGPT are essential as they serve as pre-written cues or starting points for the user and it guides them into generating narratives. The default prompts also ensure consistency and coherence in the generated narratives by providing a common starting point for users. The Storybook GPT has default prompts currently, which include “List props for the Button Component”, “How can I generate a button story”, “What are the Accordion prompts”, “Generate a story for my Dropdown Component”. The current default prompts are adequate, however they can be refined to make them more engaging and versatile for our purposes.

### Findings/ Thoughts

Although the current default prompts also work for the Storybook GPT, they can always be modified to make them more specific and therefore more useful to our developers. The first prompt, “List props for the Button component” should be kept as this is a staple for when users are navigating the StorybookGPT. The “What are the Accordion props” could be taken out as this could be repetitive. Instead a default prompt could be “Generate a story for my Speed Dial component that has copy, save and share as icons”. This default prompt is more specific as it is more action-oriented and directly guides users to generate a particular design.

Another default prompt that could be added is “How can I generate a button story in dark mode”. This default prompt is also more specific and includes a design feature that a user may be wanting to explore. Lastly, a default prompt that could be added into the StorybookGPT is “Craft a story for my Stepper component that is non-linear”. This includes a new component that has been built, and specifies a particular design to give the users more ideas when generating narratives.

### Conclusion

The 4 modified default prompts are:

- “List props for the Button Component”
- “Generate a story for my Speed Dial component that has copy, save and share as icons”
- “How can I generate a button story in Dark Mode”
- “Craft a story for my Stepper component that is non-linear”

All the prompts listed above are specific to our Storybook GPT, and tailored to guide users in exploring and utilizing various components. The modified prompts enhance user experience and inspire more creatively when building out the stories. Also, the modified prompts save time as they are tailored more towards what the user specifically needs to accomplish.

## References

- [1.1] Fine-Tuning: openAI Documentation. Accessed February 29th 2024.  
<https://platform.openai.com/docs/guides/fine-tuning/fine-tuning>
- [3.1] Storybook, Integrate Material UI with Storybook. Accessed March 18th 2024.  
<https://storybook.js.org/recipes/@mui/material>
- [3.2] Figma, Embed Figma in Storybook. Accessed March 18th 2024.  
<https://help.figma.com/hc/en-us/articles/360045003494-Storybook-and-Figma>
- [4.1] GitHub Docs. Understanding GitHub Actions. Accessed March 28th 2024.  
<https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions>
- [4.2] OpenAI Documentation. Actions in GPTs. Accessed March 28th 2024.  
<https://platform.openai.com/docs/actions/introduction>
- [4.3] ActionsGPT. Accessed March 28th 2024. <https://chat.openai.com/g/g-TYElIDU6A-actionsgpt>
- [4.4] ActionsGPT. Create OpenAPI Specifications. Accessed March 11th 2024.  
<https://chat.openai.com/share/e0bd65c5-2727-4e7d-9d3a-ab8a75b8225f>

## Appendix A: Bug Experiences

It created our chat name in Spanish!!

