**Creating Flexible Components**

Your task is to build a **highly re-usable**, custom Button component that can be used in all the following ways *(also see the code in the App.js file)*:

**"Filled" mode (default):**

1. <Button>Default</Button>

or

1. <Button mode="filled">Filled</Button>

should yield buttons that looks like this:

A screenshot of a computer

Description automatically generated

**"Outline" mode:**

1. <Button mode="outline">Outline</Button>

should yield a button that looks like this:

A blue rectangle with text

Description automatically generated

**"Text-only" mode:**

1. <Button mode="text">Text</Button>

should yield a button that looks like this:

A close-up of a text

Description automatically generated

**With Icon:**

1. <Button Icon={HomeIcon}>Home</Button>

or

1. <Button Icon={PlusIcon} mode="text">
2. Add
3. </Button>

should yield buttons that look like this:

A blue and black home button

Description automatically generated

*Hint: To make sure the icon becomes visible (if passed correctly to the component & used in there), wrap the icon component in the button with a <span> that has the class "button-icon" on it.*

*Also wrap the children prop with a <span>!*

You find all the styles (CSS classes) that are required to build a button that supports these different "modes" in the provided index.css file!

All buttons need a button CSS class - and then, depending on their mode, additional classes.

In addition, the custom Button component **must accept** all **standard props** that could be set on the built-in <button>. These props should be **forwarded** to the default <button> element that will be used in the custom Button component.

Your task therefore is to work on the Button component provided in the Button.js file. **Don't** add multiple custom components, instead work on that **one** provided component and make sure that it supports all these different modes & features. Also make sure, that if no mode is set, the *"filled"* mode is assumed as a default.