Steps for adding a new widget

1. create the required widget as a separate react component
    - place the index.js file into the ***src/components*** folder
    - name the component according to the other widgets (Widget..... .js)
2. point to the widget in ***WidgetGroup/index.js***
    - import the react component that needs to be displayed in the widget here
    - add a case in the switch statement and display this

```
      case "svg": {
        return (
          <CardBody
            style={{
              height: widgetProperties.widgetHeight * 130 - 5
            }}
            data-testid="Widget-svg"
          >
            <WidgetSvg
              widgetIndex={widgetIndex}
              widgetId={widgetId}
              chartType={chartType}
              widgetData={widgetData}
              widgetName={widgetName}
              pointData={props.pointData}
              widgetProperties={widgetProperties}
            />
          </CardBody>
        );
      }
```

3. declare the widget in the ***SwapData.js***

defines the option that will be shown in the dialog when the user wants to replace the current widget with a different widget

- include the case in the object

```
svg: {
  name: stringConstants.SVG,
  img: svg,
  widgetName: "svg",
  id: "21",
  key: "svg",
  height: 3,
  width: 4,
  responsive: false,
},
```

make sure to update the *id* value with the incremented value of the highest *id.* Also include a unique key value

4. Set the default values in the ***defaultData.js***

   this defines what the widget will show when the widget is loaded for the first time. the data loaded is picked up from this file

   - declare the widget name as an object in the widgetSwap object

```
markdown: {
  defaultData: {
    value: [],
    xAxisData: [],
    markDown: `## Markdown Content with Handlebars`,
    legend: [],
    propertyField: ["svg"],
    yAxis: [],
    startDate: "",
    ...commonData,
```

```
    },
  },
```

- also include the required values in the defaultData that is required for the widget

5. declare case in ***initialValuesHandler.js*** *(optional)*

   used to point to the initial values that need to loaded for different widgets

   can use pre-existing inputs to handle the inputs

   - point to the correct value here

```
    case "markDown":
      initialValues = {
        ...initialCommonValues,
        markDown: widget.widgetData.displayData[0].markDown,
      };
      break;
```

6. declare a case in ***PropertyForm/index.js*** *(optional)*

   used to show the required input fields in the properties panel for the widget

   can use pre-existing inputs to handle the inputs

   - add a case for the required input in the case
   - also add the required functions if required for the input handling
   - might also require modifying the ***PropertyForm/uploadProperty.js*** to manage the inputs differently

```
case "markDown":
  return (
    <MarkDown
      key={key}
      handleOnBlur={handleOnBlur}
      handleOnChange={handleOnChange}
      handleDropdownToggle={handleDropdownToggle}
```

```
      handleDropdownOpen={handleDropdownOpen}
      handleDropdownClose={handleDropdownClose}
      dropdownToggle={dropdownToggle}
      filteredOptions={filteredOptions}
      handleOptionSelect={handleOptionSelect}
      handleOnFocus={handleOnFocus}
      formik={formik}
      widgetName={widgetName}
      inputFieldCount={inputFieldCount}
      handleRemoveField={handleRemoveField}
      handleAddField={handleAddField}
      index={index}
    />
  );
```

7. declare and import required string constants in the ***stringConstants.js***
   - declare the string constants here and import them in the required places to use it

```
IMAGE_DETAILS: "imageDetails",
SVG_DETAILS: "svgDetails",
```

8. update the values in the ***Dashboard.js***

   this is required to pass the props to the widget component

   will be required for loading of the initial data

   - if required, update a new key here to use the required property fields that will be sent to the widget

```
      case "svgDetails":
        widgetProperties.svgUrl = value["svgUrl"];
        widgetProperties.fileName = value["fileName"];
        break;
```