File structures:

1. Visual.ts: Main file where code for making the visual and its manipulation is stored.
2. Data.ts: Stores the code for the function transform which parses through the report data to store part number (or whatever column field is given as input to the visual) in data.ts
3. Settings.ts: For custom formatting options for the visual (give user ability to change color, size of visual etc.)
4. Visual.less: Stores formatting code for the HTML DOM similar to CSS
5. Capabilities.Json: Stores the properties of the visual.
6. Node\_Modules: Stores all the modules for formatting or beautifying the visual. Note: PowerBI-API's are also stored in this folder which can be queried/called if the visual needs to filter data.
7. Pbiviz.Json: File stores the information about the visual and its programmer in Json format. Please change author name, github URL, email etc to your own especially if visual needs to be published to Microsoft Appstore.

**Visual.ts:** Mainly broken down into 2 parts; import statements and export class. The export class is further broken down into two main segments, the constructor function and the update function.

The constructor function is executed only once at the start when the PowerBI report loads in. The function handles the code for generating the visual I.e The Textbox, buttons etc. All code related to giving output, clearing the visual with the clear button etc. Is written in this

The update function does what its name suggests I.e updates the visual anytime there’s new input or button clicked etc. This is the function wherein the transform function from data.ts can be called in to access part numbers stored in variable data.values as an array.

The invalid\_search function takes user input(text)I.e string and part numbers(part\_nos) I.e array as arguments. The function first splits the user input(text) by commas and stores each substring into a new array I.e NumArr

The function then loops through part numbers to check if each user input number is present in the part number or not. Note: The flag variable is set to 1 if number is present in part number otherwise flag is set to 0 and that user input is pushed/appended into a new array invalid\_nos which stores all the invalid user inputs. Therefore to get an output of invalid\_nos simply call this function and give user input and part\_nos as argument.

Lastly, inside the constructor function, the variable target which is declared as an HTML Element is used to store the HTML DOM code using .innerhtml property. This variable stores all of the visual frontend code which includes the text box, what happens when clear button is pressed etc.

Note: Unfortunately, PowerBI does not allow .alert() prompts from Javascript so the only reliable method left for getting an output is to use HTML properties.

Note 2: The class of the buttons I.e C-Glyph search is basically the CSS code for the buttons in Visual.less .Right now the buttons are not ‘beautified’. For future visual development once the output box is developed, adding in features such as changing cursor to click animation when hovering over the buttons will enhance user experience which can be done by coding in the visual.less file.

Note 3: The console.log() function outputs the backend code to the developer tab which can be accessed using ctr+shift+I or going into more tools on chrome settings and clicking developer tools then going under console tab. This feature is useful if you want to extract data types other than matrix(data type/visual in which part numbers is stored) in which the data.ts file can be edited accordingly.

Note 4: The capabilites.json file stores properties of the visual I.e how many inputs to take(which I've restricted to 1 under max, general property). Also for future version, filter property is set to true so the function can also act as filter for any future versions. The datareduction algorithm stores how many maximum input rows to take which is inherently capped to somewhere around 10,000 by Microsoft but that dosen’t affect our visual properties in any way.

I’ve also added comments on every code snippet so programmer can understand its function.