Assumptions

1. The user wants to trigger a filter only when a new value from values dropdown is selected.
    a. This is because as possible operators change based on selected property, and values are populated based on the selected operator, I thought it would be a better option to give the user the chance to select the new value and then trigger a filter instead of changing the state unnecessarily based on a default value
    b. For example, if the user previously selected (“Product Name”, “Equals”, “Headphones”) and they change the property to be “Weight” instead of “Product Name”, they have to select a new value to trigger a change.

2. We will always get valid results from getProperties and getOperators method from the datastore and the list of valid operators for each property remains constant.

Approach

First I tried to find components inside the UI that can be separated out in a component of its own. For example, we can see that the page had 3 Dropdown menus so instead of having 3 separate Dropdown menu in the main component I decided to separate it and make reusable DropDown components to reduce code duplication and redundancy.

Then I looked into how I would go about designing the main component which implements the filtering logic. The filtering process was to be carried out on three parameters:-

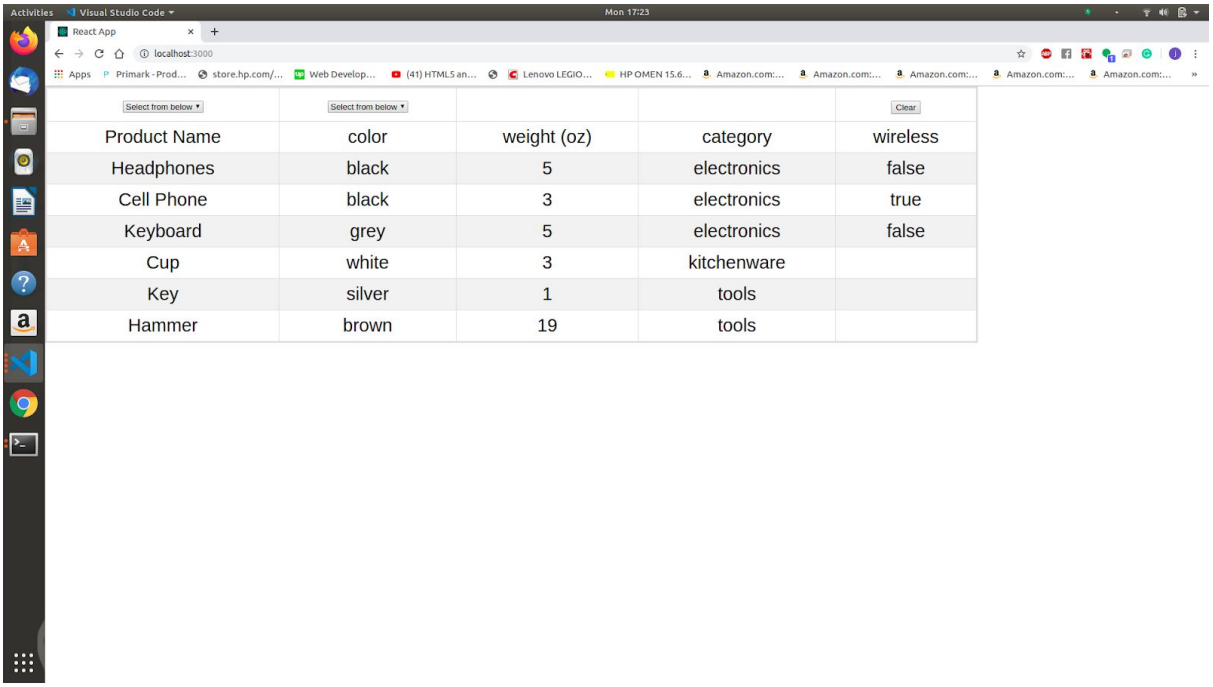
1. The Id of the selected property.
2. The Id of the selected operator.
3. The selected values by the user.

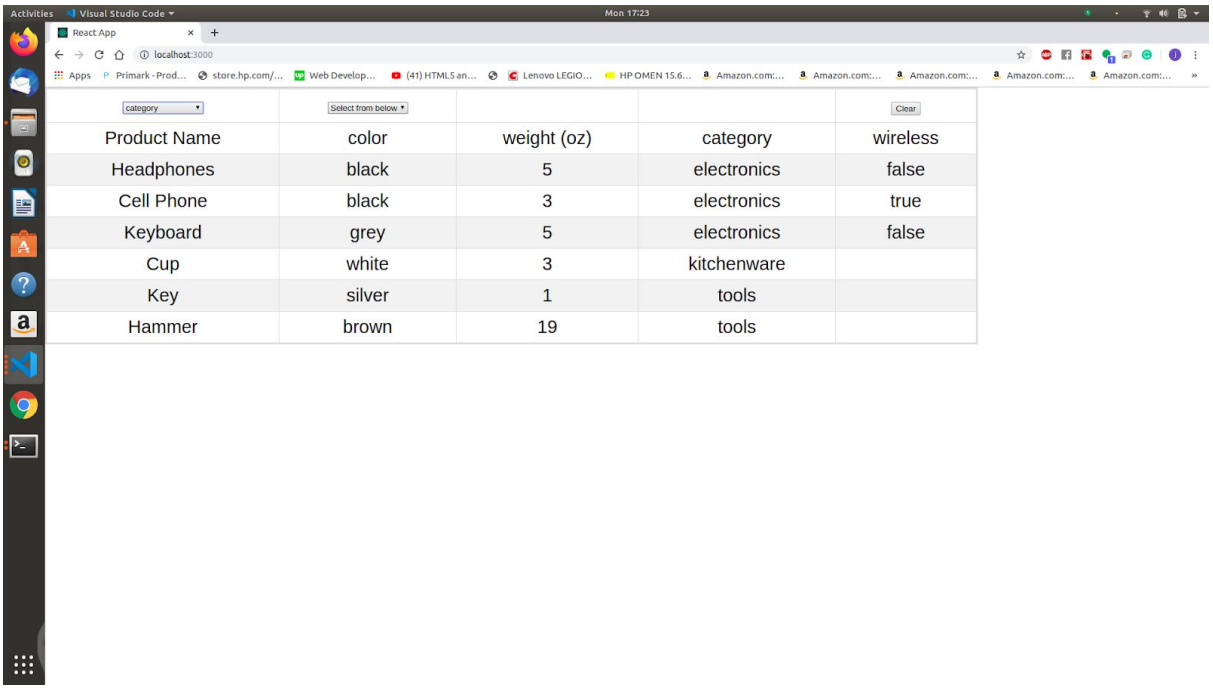
I wanted to figure out what information should be shown in the Dropdown menu i.e. dynamically read properties from datastore and load them up in dropdown menus. Once I configured this, I decided to implement logic which loads appropriate operator based on which property is selected for filtering.
I had to decide when should I compute all the possible values which are possible for any particular property based on the data given in datastore. I generated a list of all unique values for a given property and rendered that list on the basis of selected property and operator.
I decided to compute these values initially when the system loads the entire data instead of computing it every time filter is selected for one particular property. This saves repetitive computation for the system.
Now I decided to implement the logic of actual filtering of data for each operator at a time.

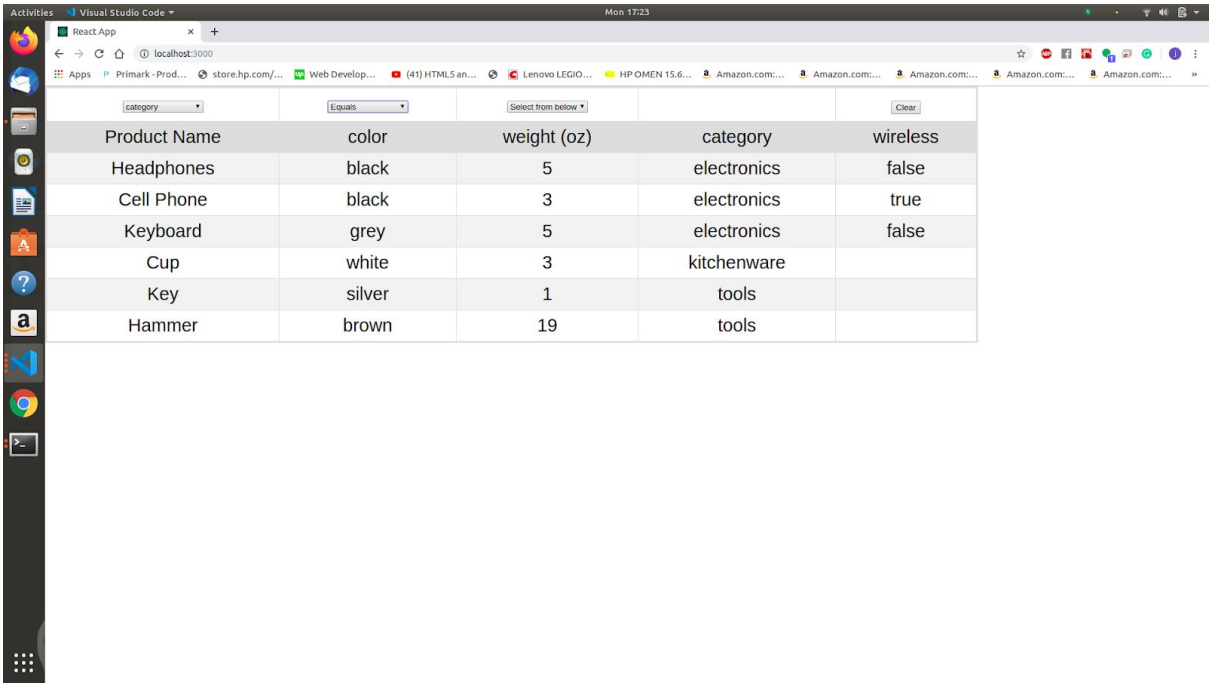
Because the only difference in the filtering process was the operator selected by the user, I made one FilterSet method and wrote different cases of filtering based on the selected operators instead of writing different methods for filtering.
After filtering the data, I rendered the visible dataset into the table.

The overall time spent in this assignment was around 20 hours(+- 2 hours). I sometimes got stuck on some issues like how to reduce code complexity when calculating the same thing for different parameters. While some things passed by very smoothly. I could have improved some aspects which I have included below in the “Scope for improvement” section.

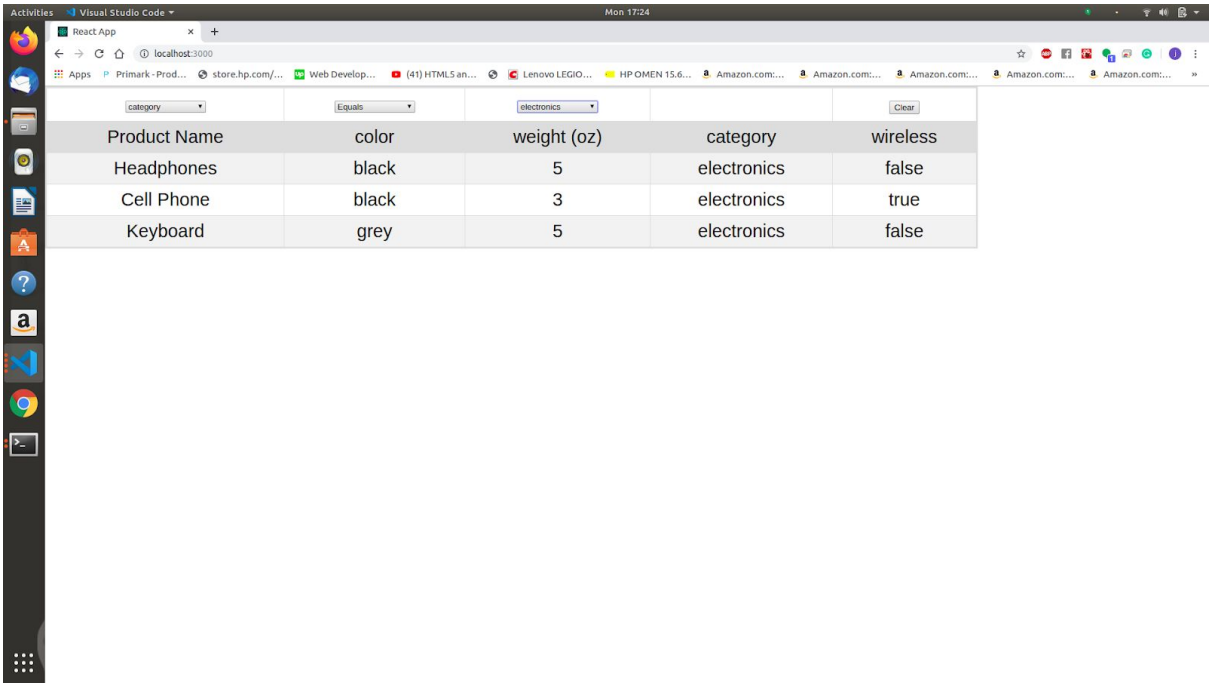
Results

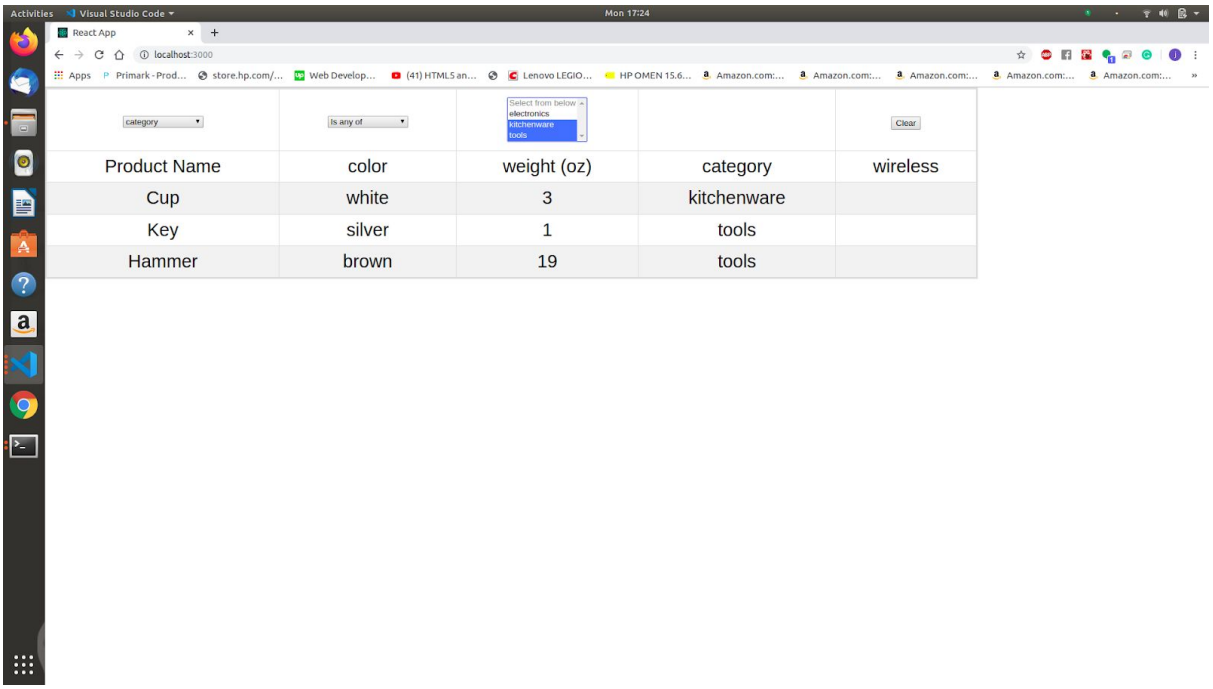

Activities
Visual Studio Code
Mon 17:23
React App
localhost:3000
Apps
Primark - Prod...
store.hp.com/...
Web Develop...
(41) HTML5 an...
Lenovo LEGIO...
HP OMEN 15.6...
Amazon.com:...
Select from below
Select from below
Clear
Product Name
color
weight (oz)
category
wireless
Headphones
black
5
electronics
false
Cell Phone
black
3
electronics
true
Keyboard
grey
5
electronics
false
Cup
white
3
kitchenware
Key
silver
1
tools
Hammer
brown
19
tools

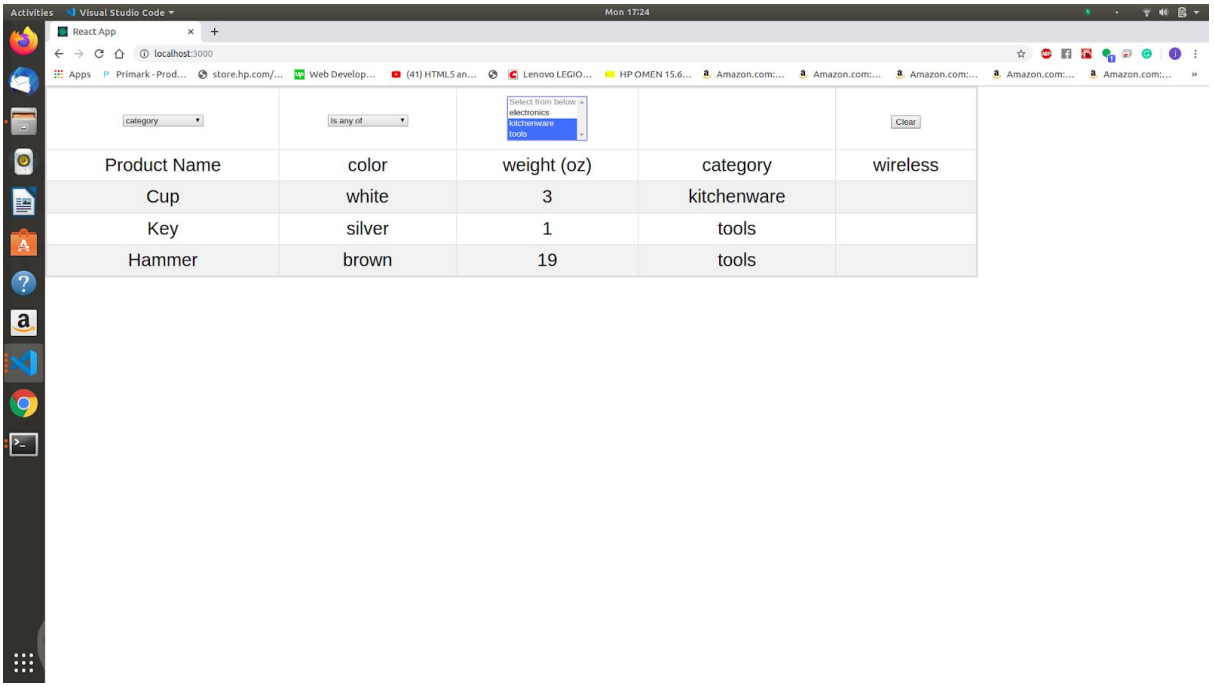

Activities
Visual Studio Code
Mon 17:23
React App
localhost:3000
Apps
Primark - Prod...
store.hp.com/...
Web Develop...
(41) HTML5 an...
Lenovo LEGIO...
HP OMEN 15.6...
Amazon.com:...
category
Select from below
Clear
Product Name
color
weight (oz)
category
wireless
Headphones
black
5
electronics
false
Cell Phone
black
3
electronics
true
Keyboard
grey
5
electronics
false
Cup
white
3
kitchenware
Key
silver
1
tools
Hammer
brown
19
tools

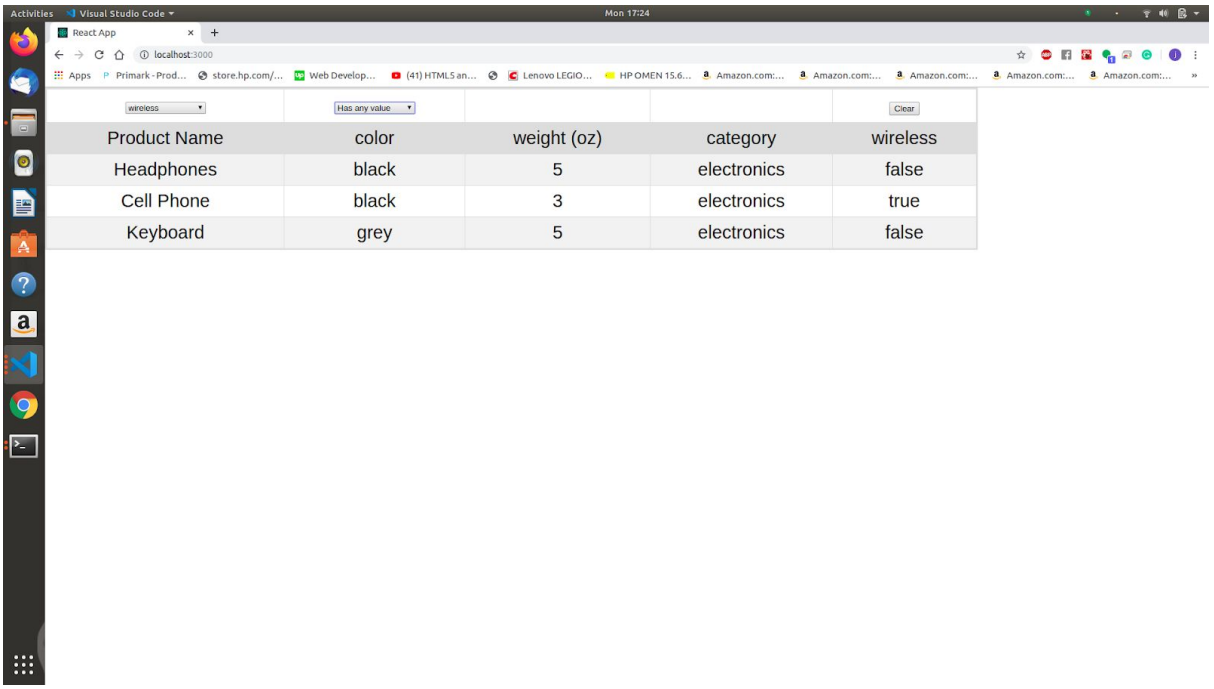


Product Name	color	weight (oz)	category	wireless
Headphones	black	5	electronics	false
Cell Phone	black	3	electronics	true
Keyboard	grey	5	electronics	false
Cup	white	3	kitchenware	
Key	silver	1	tools	
Hammer	brown	19	tools	


Activities
Visual Studio Code
Mon 17:24
React App
localhost:3000
Apps
Primark - Prod...
store.hp.com/...
Web Develop...
(41) HTML5 an...
Lenovo LEGIO...
HP OMEN 15.6...
Amazon.com:...
category
Equals
electronics
Clear
Product Name
color
weight (oz)
category
wireless
Headphones
black
5
electronics
false
Cell Phone
black
3
electronics
true
Keyboard
grey
5
electronics
false


Activities
Visual Studio Code
Mon 17:24
React App
localhost:3000
Apps
Primark - Prod...
store.hp.com/...
Web Develop...
(41) HTML5 an...
Lenovo LEGIO...
HP OMEN 15.6...
Amazon.com:...
category
Is any of
Select from below
electronics
kitchenware
tools
Clear
Product Name
color
weight (oz)
category
wireless
Cup
white
3
kitchenware
Key
silver
1
tools
Hammer
brown
19
tools





Dependencies/ How to run the project

1. Make sure your system has node installed. (https://nodejs.org/en/download/)
2. Go to React Project directory and run `npm install` command.
3. Now run `npm start` command.

This should open up a tab in your browser where you can play around with the application.

Scope of improvement

1. We can divide the independent components in different files to reduce code complexity.
2. Instead of populating the possible operators for each property statically, we can try to figure a way out for dynamic loading,
3. Instead of writing an if-else ladder we can have different filter components for each condition while keeping code duplication in check.