**CSS Properties:**

object-fit: contain; // used for images to maintain proportion

// to make the items in a parent element evenly spaced out on the same level

//following is for parent element

display: flex;

align-items: center; //on the same level

justify-content: space-between; //evenly spaced

//to make a child element occupy all the remaining space in a line inside a flex //parent

flex: 1;

**to make an element stick to top:**

position: sticky;

top: 0;

**to specify the position order like on front or behind of elements when stacked one upon other**

z-index: -1; // behind one element

z-index: 100; // on front of 100 elements, so basically never hidden

**whenever using weights using ‘flex’ to takeup some % of screen, make sure all weights add up to 1. Otherwise, there will be weird spacing in between.**

.sidebar {

flex: 0.2;}

.recommendedVideos {

flex: 0.8;

}

**\*flex can be used for assigning weights of elements iff the parent, ‘display: flex’**

**We can also pass components as props in react!**

import HomeIcon from ‘materialui/blah/blah’;

<SidebarRow Icon={HomeIcon} title=”Home” />

//HomeIcon is a component

//\*\*Remember when passing the component, prop name must start with **uppercase,** as in **Icon.**

When rendering the passed component we write ‘<props.Icon />’

**Apply styles when 2 or more classes are present for an element**

.sidebarRow.selected {

Background-color: lightgray;

}

**Only when both ‘sidebarRow’ and ‘selected’ classes are present apply bg-color to lightgray**

display: flex;

flex-wrap: wrap;

//**child elements accupy the row, when row is filled wrap up to next row.**

**Router:** <Router> </Router> is responsible for determining what URL loads what COMPONENT.

Whenever we make change in the app, we have to do ‘npm run build’ again.

npm run build : removes all the extra development tools and makes the app production ready.

Final Deploy :

https://clone-d4116.web.app/