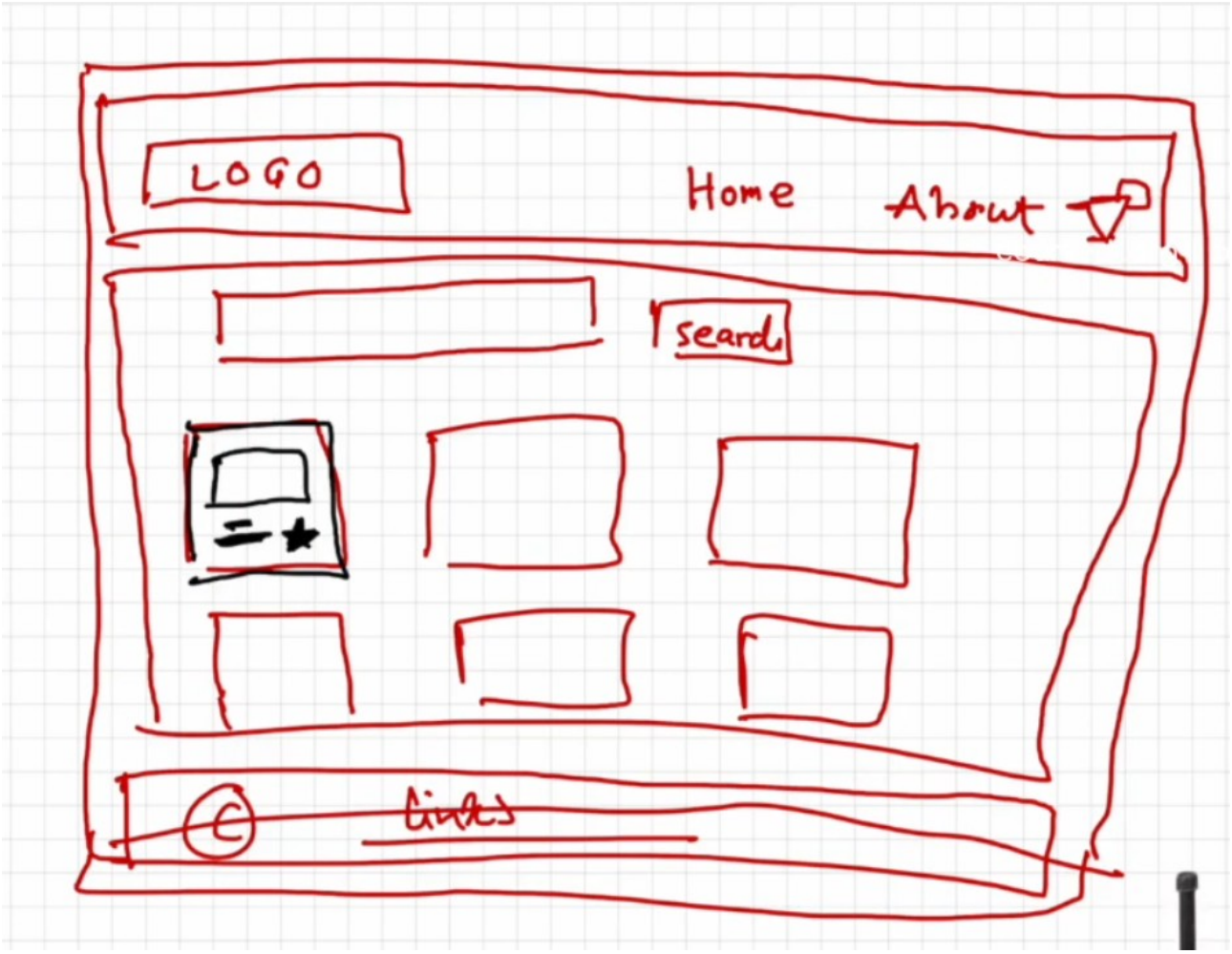
EPISODE 4 :TALK IS CHEAP SHOW ME THE CODE

PART 01:

In this episode we are going to design a food app. Lets make a plan of the app.



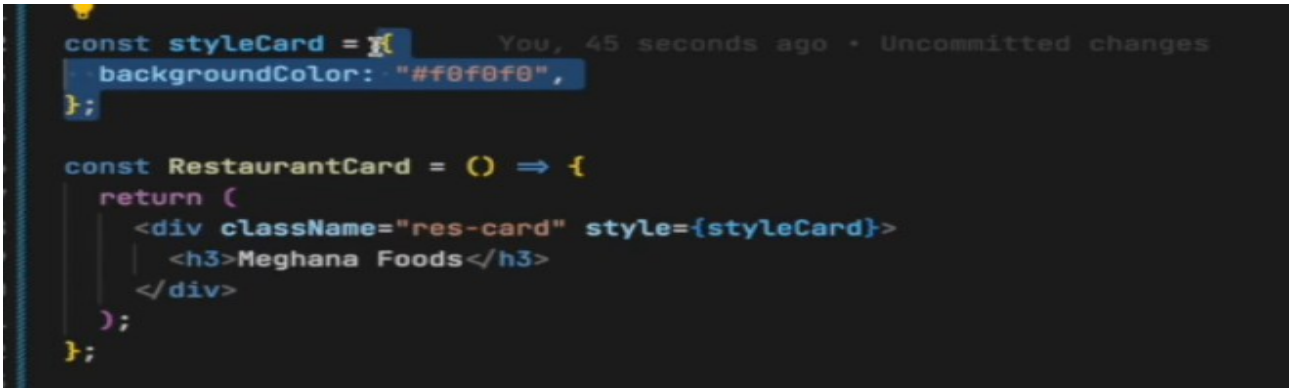
Overview of an app:

- HEADER
  - logo
  - nav-items

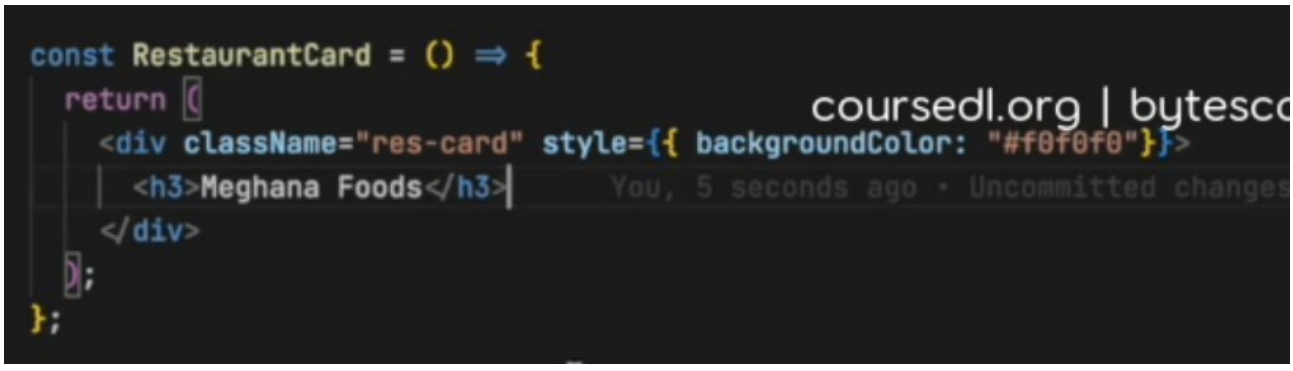
- BODY
  - search bar
  - Restaurant container
    - restaurant-cards
      - (img,name,star.cusines)
- FOOTER
  - copyrights.

WAYS OF WRITING CSS

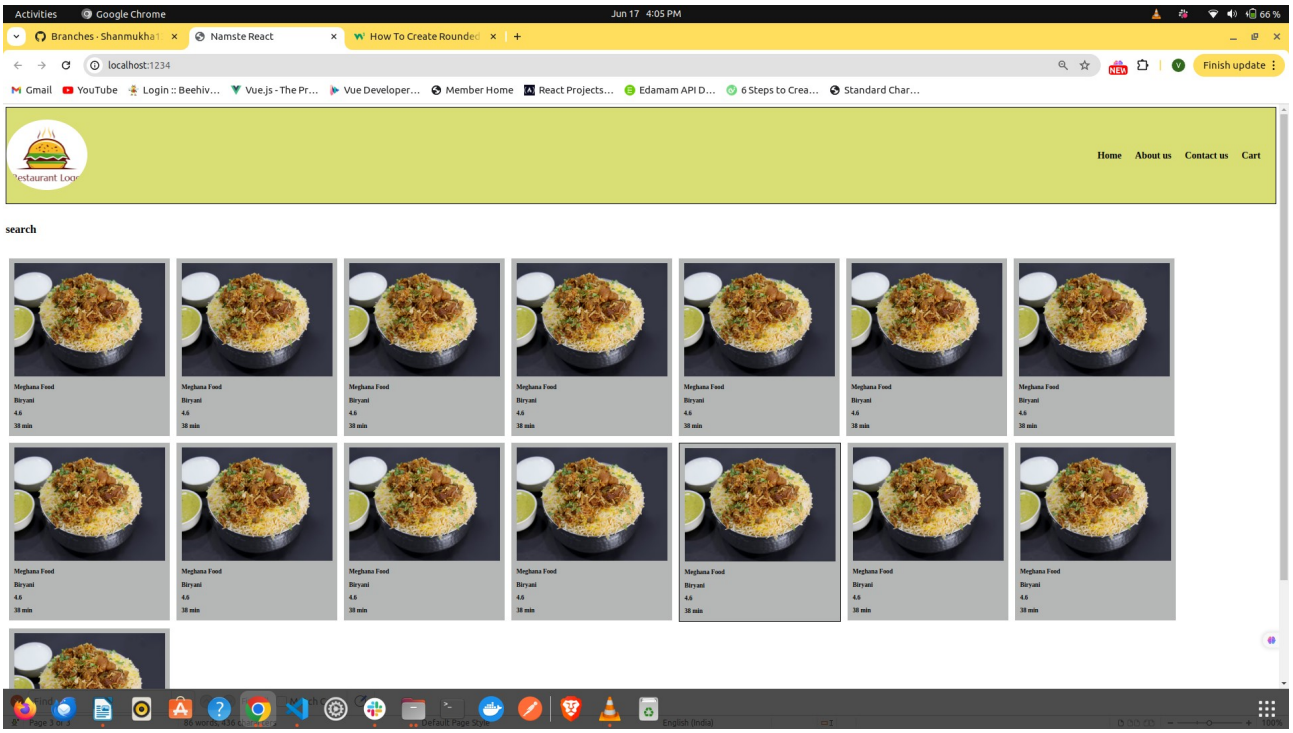
I.CSS as a javascript object.



II.CSS inline styling.



In this episode we part 1 we have created a header body and cards.something like below.



In this code we saw the text is hard coded.For each card the data is same

PART 02:

In this part we will see props .

Props are nothing but parameters to a function.

Ways of using props:

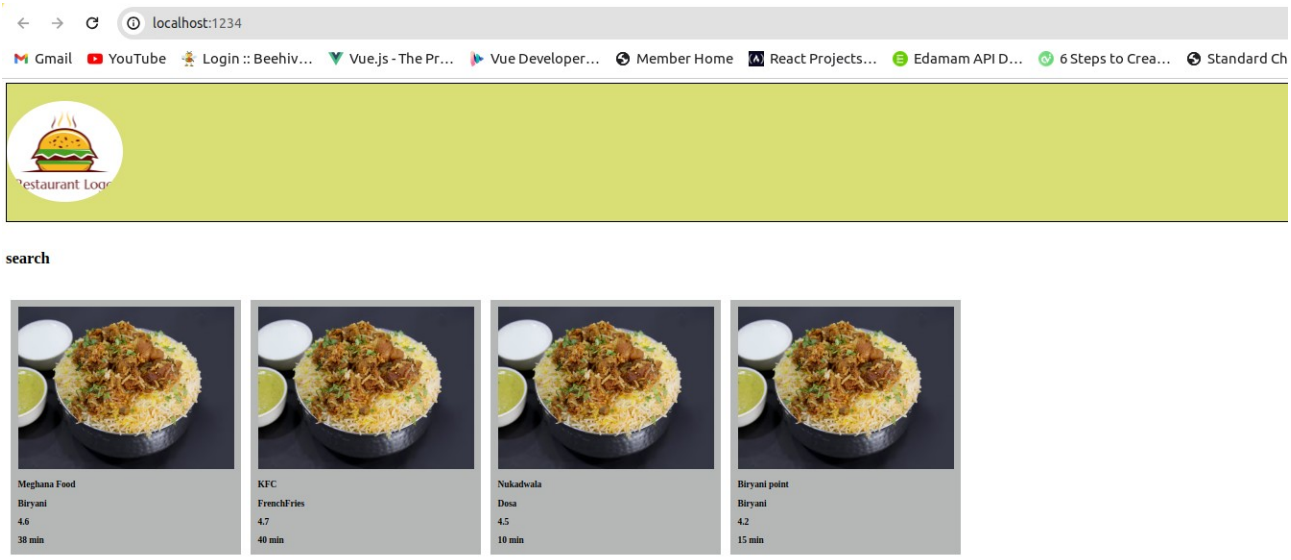
```jsx
const ResCard = (props) => {      You, 1 second ago • Uncommitted changes
    return(

        <div className="res-card">

            <img className="res-logo" src="https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/inntwq
            <h3>{props.resName}</h3>
            <h3>{props.resFood}</h3>
            <h3>{props.foodRatings}</h3>
            <h3>{props.time}</h3>
        </div>
    )
}
const Body = () => {
    return(
        <div className="body-container">
            <div className="search">
                <h5>search</h5>
            </div>
            <div className="res-container">
                <ResCard resName="Meghana Food" resFood="Biryani" foodRatings="4.6" time="38 min"/>
                <ResCard resName="KFC" resFood="FrenchFries" foodRatings="4.7" time="40 min"/>
            </div>
        </div>
    )
}

const AppLayout = () =>{
```

Way2 :(destructuring)

```jsx
const ResCard = (props) => {
    const {resName,resFood,foodRatings,time} = props;
    return(

        <div className="res-card">

            <img className="res-logo" src="https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/inntwq
            <h3>{resName}</h3>
            <h3>{resFood}</h3>
            <h3>{foodRatings}</h3>
            <h3>{time}</h3>
        </div>
    )
}
const Body = () => {
    return(
        <div className="body-container">
            <div className="search">
                <h5>search</h5>
            </div>
            <div className="res-container">
                <ResCard resName="Meghana Food" resFood="Biry (property) foodRatings: string min"/>
                <ResCard resName="KFC" resFood="FrenchFries" foodRatings="4.7" time="40 min"/>
                <ResCard resName="Nukadwala" resFood="Dosa" foodRatings="4.5" time="10 min"/>                You, 1 second ago
            </div>
        </div>
    )
}
```

Way 3: (destructuring)

```jsx
const ResCard = ({resName,resFood,foodRatings,time}) => {
    return(

        <div className="res-card">

            <img className="res-logo" src="https://media-assets.swiggy.com/swiggy/image/upload/fl_lossy,f_auto,q_auto,w_660/inntwq
            <h3>{resName}</h3>
            <h3>{resFood}</h3>
            <h3>{foodRatings}</h3>
            <h3>{time}</h3>
        </div>
    )
}
const Body = () => {
    return(
        <div className="body-container">
            <div className="search">
                <h5>search</h5>
            </div>
            <div className="res-container">
                <ResCard resName="Meghana Food" resFood="Biryani" foodRatings="4.6" time="38 min"/>
                <ResCard resName="KFC" resFood="FrenchFries" foodRatings="4.7" time="40 min"/>
                <ResCard resName="Nukadwala" resFood="Dosa" foodRatings="4.5" time="10 min"/>
                <ResCard resName="Biryani point" resFood="Biryani" foodRatings="4.2" time="15 min"/>                You, 1 second a
            </div>
        </div>
    )
}
```



but always hardcoding props is also not a good idea , so we take data from apis in json.
To view json in a proper format install ***json viewer chrome extension.***

Config driven UI:

Config-driven UI is a technique that allows you to create user interfaces based on a configuration file, such as JSON, or a

TypeScript file that defines the layout and content of the UI components. This can be useful for creating dynamic and customizable UIs without hard coding them.That is why we have different cards ,different corousels for based on different locations in same food order app.

Keys in map:

- It is always a good idea to use keys while using map function.
- Ex:Lets take an example ,we have a large data of array which contains list of objects in it.the data in that object include name of item,unique key,delivery time which will be used in cards of food item.
- You can execute all this array items data in broswer by map() function .
- If new object is added the all the initial cards will rerender in browser because react fall in the ambiguity to where to keep that card.so all the cards will be reredered .it is not a good practice.
- So if you use key the react will identify the new card and it only renders it all the previous cards will not re -rendered which improves performance.
- So,using keys is must and should while working with map() function.

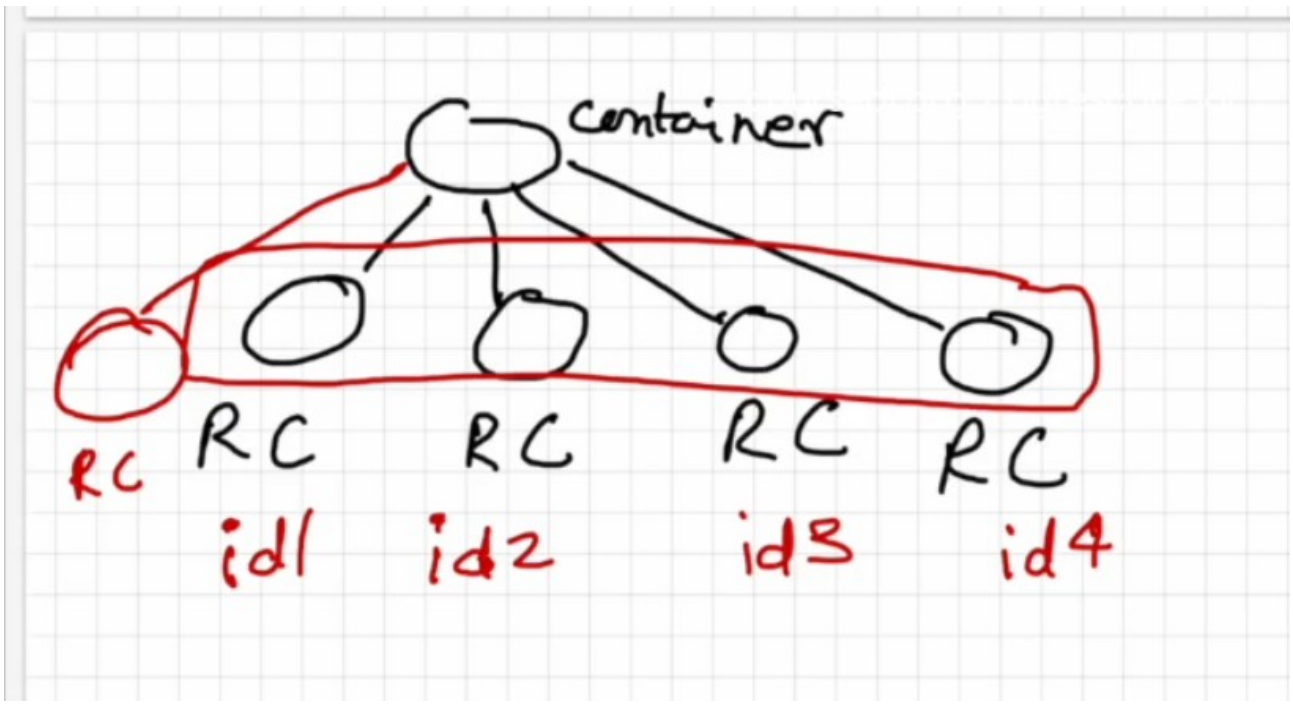
Priorty:

no key(not acceptable)<<<<index as akey<<<<<<unique id as akey

Using an index as a key in React components can lead to problems, including:

- **Performance issues**

When items are added, removed, or re-ordered, the index can change and React may not be able to reconcile the order. This can lead to unnecessary re-renders, inconsistencies in the application's state, and inefficient reconciliation of the virtual DOM, especially for large lists.


container
RC
RC
RC
RC
RC
id1
id2
id3
id4