How app or web application is build?

* Planning
  + What we are going to build and process of building it
  + Components we can have
    - Header
      * Logo component
      * Nav item component
    - Body
      * Search component
      * Card component i.e food restro cards
        + Cards component
    - Footer
      * Copy right
      * Link
      * Address
      * Contact
* Props in components
  + Something which we can pass to the component 🡪normal arguments to function
  + React will take all the argument into objects and pass as props to child component

Rendering child with props

<RestroCards

        resname="Getha canteen"

        cousin="biryani,South Inidan style"

        rating="2.8"

        deliveryTime="38"

        price="400"

      />

const RestroCards = (props) => {

  return (

    <div className="res-card">

      {props.resname}

      <img

        className="res-logo"

        src="https://media-cdn.tripadvisor.com/media/photo-s/13/04/71/71/chicken-biryani.jpg"

      ></img>

      <h3>Geetha Canteen</h3>

      <h5>biryani,South Inidan style</h5>

      <div className="res-card-footer">

        <h6>3.8</h6>

        <h6>38 minutes</h6>

        <h6>$300 for two</h6>

      </div>

    </div>

  );

};

Config driven ui 🡪

When u open a swiggy there are many offers available based on different location .. the ui is driven by backend .. based on location for example

1.according to data from backend we can display contents

2.the application can work in different location, different areas and based on config the ui will be rendered

Cdn ->cloudnary id .. all images are got from CDN

Note:

The component are just functions example below

const Restro = () => {

  return (

    <div className="restro-container">

      {resObj.map((data, index) => {

        // return <RestroCards resData={data} key={index} />;

        return RestroCards({ resData: data, index });

      })}

    </div>

  );

};

Below code can be written as above code also

const Restro = () => {

  return (

    <div className="restro-container">

      {resObj.map((data, index) => {

        return <RestroCards resData={data} key={index} />;

      })}

    </div>

  );

};

**Each child should have unique key property .. what does it mean?**

* when we loop through element or we have any child inside parent then we should use key
* **why?**
  + We have html and have many child inside it
  + when components are present in same level and when new child comes inside the react cannot identify at which place it need to put the new child
  + **so react cleans all the child and then re-render all the child again even if we need to add only one child**
  + **when we give key .. it already knows there are child element with key 1,2,3 for example and renders only 4 which is not present**
  + it takes big performance hit when we don’t add key
  + can we use index also as key from array ???--> No 🡪 why
    - it is not recommended to use index as key why?

If you use **index as a key**, it can cause **unexpected bugs**, especially when:

* Items are **reordered**
* Items are **inserted or deleted**

What is and <></> and <React.fragment></React.fragment> ?

* Both are same and <> is shorthand for react.fragment
* They are used to group the element together without using tags like div or any other

| **Feature** | **<React.Fragment>** | **<> (shorthand)** |
| --- | --- | --- |
| Can add key prop | ✅ Yes | ❌ No |
| Can add attributes | ✅ Yes | ❌ No |
| Readability | 🟡 Verbose | ✅ Cleaner and shorter |
| Use when? | When key or props are needed | Most common use case |

**❓ What is Virtual DOM?**

The **Virtual DOM (VDOM)** is a lightweight, in-memory **representation of the real DOM** used by React to improve performance.

**What is Reconciliation in React?**

* When the component is rendered react create a virtual copy of DOM elements
* When there is change in props or state or onclick events then new virtual DOM copy is created with changes i.e the virtual DOM is updated with changes from component
* Then the old virtual DOM and new virtual DOM is compared and only the changed element or props is updated in **real DOM** instead of rendering the whole DOM elements each time.
* The process of comparing and updating real dom is called reconciliation
* **After diffing and jsx updated with new data the old virtual DOM gets updated with new virtual DOM ready for next cycle**

Yes — ✅ **the Virtual DOM does get updated in React**, but **not during the diffing itself**.

Let me break it down clearly for you:

### ✅ When does the Virtual DOM get updated?

1. **Initial Render**
   * React builds a Virtual DOM from your JSX.
   * This is stored in memory (not shown in the browser).
2. **When a component updates (state/props change):**
   * React creates a **new Virtual DOM tree** from the updated component.
   * It compares the **new Virtual DOM** to the **previous Virtual DOM**.
   * React applies only the changes (the “diff”) to the **real DOM**.
3. ✅ After the comparison (diffing), **the new Virtual DOM replaces the old one** in memory.  
   So yes, the Virtual DOM **gets updated after diffing** — not during.

### 🔄 Simplified flow:

jsx

CopyEdit

1. Render Component → Create VDOM (v1)

2. State change → Create New VDOM (v2)

3. Diff: Compare v1 and v2

4. Update only the changed parts in real DOM

5. Set v2 as the current Virtual DOM (v1 = v2 now)

### 🧠 TL;DR:

✅ The **Virtual DOM gets updated** after a component re-renders.  
🔍 It’s first used for **comparison**, and then the **new version becomes the current VDOM** for the next update cycle.

**What is React Fiber?**

**React Fiber** is the **reconciliation engine** in React — it’s the **core algorithm** behind how React renders and updates the UI.

It was a **complete rewrite** of React's internal rendering logic introduced in **React 16** to make React **faster**, **smarter**, and more **flexible**.

**What is a Config Driven UI ?**

A **Config-Driven UI** is a **user interface** that is dynamically generated or adjusted based on **configuration settings**, rather than being hardcoded with fixed content and design. The configuration typically comes in the form of **JSON, YAML, or other data structures** that describe the components, layout, styling, and behavior of the UI.