MATCH CARDS GAME WITH REACT JS

React JS:

React JS is a library of javascript, it is not a framework . There are various libraries like this. Example node js. React is a component based architecture. While HTML and CSS felt bored with dynamic web pages Java Scripts is developed by Brendan Eich . These webpages and its elements run on a browser and the first browser applied in Mozilla Firefox where Brendan Eich was working. Later it is applied in all the browsers. Every Browser has separate javascript engines. Not only in browsers, JS can be used in all displays. After this revolution ECMA script was created in1997.ECMAScript, also known as JavaScript, is a programming language adopted by the European Computer Manufacturer's Association as a standard for performing computations in Web applications.

Match cards game:

Introduction:

The game starts with all the cards face down and players take turns to turn over two cards. If the two cards have the same picture, then they keep the cards, otherwise they turn the cards face down again. The winner is the person with the most cards when all the cards have been taken.

Instructions:

- ➔ Chose a 2 cards one after another
- ➔ If a card is chosen you cannot touch it again it shows you alert
- ➔ If the cards are not same they will turn back
- ➔ If the cards matched they remain as opened
- ➔ Match next pairs of cards
- ➔ The games moves until all the cards are matched

Working Platform: The entire code is coded in ‘CodeSandBox’, an online Javascript compiler with react library inbuilt. It is a live platform where code is reflecting lively updates in a web browser tab.

File structure :

- ➢ Public
  - Image
  - Index.html
- ➢ Src
  - Components
    - Card.js
    - Cards.js
  - App.js
  - Index.js
  - Index.css
  - Style.css
- ➢ package. json

Index.html:

Index contains the basic display the error if javascript is off when the javascript fails. This is the based file for our game where all the elements and components play their roles over this file . This a default file created while using react js

Img folder:

This folder contains 8 phots which will be displayed as cards in our game

Source folder:

This folder contains our javascript files and its components files. A separate folder is created for components to avoid confusions and make a proper file structure. Other files are stored and linked with components. These file are rendered to display on the webpage

Components :

Component is encapsulation of DOM elements. In this game i have created 2 components ‘cards.js’ and ‘card.js’. The main use of javascript is to make changes in a particular element without making any changes to the entire page. Those particular elements are the components. In this game we are only going to make changes with the cards so the components are card. Here the use of 2 files is one

for storage of images and operations whereas other one is for rendering into Apps.js

Cards.js

Here in this part of the script I have used the ‘useState’ and ‘useRef’ of the hook concept of react. I have created a function named cards() and created an array of all my photos from the ‘img’ folder in public. This array is randomized using the math function to shuffle and display the cards.

Two variables created in the name of previousCardState, setPreviousCardState and value is set as -1 via setState which is the index value. A const named previous index is created and set as -1 for indexing the reference of previous one

A new function named current card is created and assigned to a constant name ‘matchCheck’ and assigned using the arrow function. There is an if else condition inside is made. The if condition checks if the index position of the currently selected card is equal to the previously selected card if it so then sets both the card state as ‘active’ which means the card will display the image permanently. In the else part the useStare of the previous card is set to -1 again and both cards are set to unmatched. So the cards are flipped again.

Another new function named index card is created and assigned to a constant name ‘clickHandler’ and assigned using the arrow function. This contains an nested if else statement to pop errors.

- If the index is not equal to the previous card
    - If the selected card is selected again which is flipped and stated as active an alert pops “already matched”
    - It moves to the ‘matchcheeck’ function
- Else pops “the card is selected currently” when you click the same card twice

Finaly a return statement is given for the cards function with mapping index and card array of useState along with the attributes of card,index,key,clickhandler to card.js

useState: The useState hook is used for storing variables that are part of your application's state and will change as the user interacts with your website. The useEffect hook allows components to react to lifecycle events such as mounting to the DOM, re-rendering, and unmounting**.**

useRef:useState returns 2 properties or an array. One is the value or state and the other is the function to update the state. In contrast, useRef returns only one value which is the actual data stored. When the reference value is changed, it is updated without the need to refresh or re-render.

```
import { useState, useRef } from "react";
import Card from "./Card.js";
var x= 0 ;
export default function Cards() {
 const [cards, setCards] = useState(
   [
     { id: 0, name: "akash0", status: "", img: "/img/01.jpg" },
     { id: 0, name: "akash0", status: "", img: "/img/01.jpg" },
     { id: 1, name: "akash1", status: "", img: "/img/02.jpg" },
     { id: 1, name: "akash1", status: "", img: "/img/02.jpg" },
     { id: 2, name: "akash2", status: "", img: "/img/03.jpg" },
     { id: 2, name: "akash2", status: "", img: "/img/03.jpg" },
     { id: 3, name: "akash3", status: "", img: "/img/04.jpg" },
     { id: 3, name: "akash3", status: "", img: "/img/04.jpg" },
     { id: 4, name: "akash4", status: "", img: "/img/05.jpg" },
     { id: 4, name: "akash4", status: "", img: "/img/05.jpg" },
     { id: 5, name: "akash5", status: "", img: "/img/06.jpg" },
     { id: 5, name: "akash5", status: "", img: "/img/06.jpg" },
     { id: 6, name: "akash6", status: "", img: "/img/07.jpg" },
     { id: 6, name: "akash6", status: "", img: "/img/07.jpg" },
     { id: 7, name: "akash7", status: "", img: "/img/08.jpg" },
     { id: 7, name: "akash7", status: "", img: "/img/08.jpg" }
   ].sort(() => Math.random() - 0.5)
 );
 const [previousCardState, setPreviousCardState] = useState(-1);
 const previousIndex = useRef(-1);
```

```
const matchCheck = (currentCard) => {
  if (cards[currentCard].id === cards[previousCardState].id) {
    cards[previousCardState].status = "active matched";
    cards[currentCard].status = "active matched";
    setPreviousCardState(-1);
    <p>right match</p>
    x=x++;
  } else {
    cards[currentCard].status = "active";
    setCards([...cards]);
    setTimeout(() => {
      setPreviousCardState(-1);
      cards[currentCard].status = "unmatch";
      cards[previousCardState].status = "unmatch";
      setCards([...cards]);
    }, 1000);
  }
};

const clickhandler = (index) => {
  if (index !== previousIndex.current) {
    if (cards[index].status === "active matched") {
      alert("already matched");
    } else {
      if (previousCardState === -1) {
        previousIndex.current = index;
        cards[index].status = "active";
        setCards([...cards]);
        setPreviousCardState(index);
      } else {
        matchCheck(index);
        previousIndex.current = -1;
```

```
      }
    }
  } else {
    alert("card currently seleted");
  }
};

return (
  <div className="container">
    {cards.map((card, index) => {
      return (
        <Card
          x={x}
          card={card}
          key={index}
          index={index}
          clickhandler={clickhandler}
        />
      );
    })}
  </div>
);
}
```

Card.js

Here the attributes are passed for the cards function in cards.js file and checks the card status as active or unmatched.

```
import { useState } from "react";

export default function Card({ x,card, index, clickhandler }) {
 const cardClassName = card.status ? "active" : "";
```

```
 return (
   <div className={`card ${card.status}`} onClick={() =>
clickhandler(index)}>
     <img src={card.img} alt={card.name} />

   </div>
 );
}
```

Apps.js:

This the file where all the components and functions are integrated and pass one function to render in the web page. ‘App’ function is created and the card function from card.js is called and I created a button here to reload the page when you want to restart the page or to a new game when you're done.By default the ‘App’ function exported to the index.js

```
mport "./styles.css";
import Cards from "./components/Cards.js";
import "./components/Card.js";

function App() {
 return (
   <div
     className="centered-div"
     style={{
       alignItems: "center",
       justifyContent: "center",
       height: "100vh"
     }}
   >
     <h1>JOOKER MATCH CARDS</h1>
     <Cards />
```

```
     <button className='button' onClick={() =>
window.location.reload()}>New Game</button>;
     {/* <div> <button
onClick=location.reload();>restart<button></div> */}
   </div>
 );
}

export default App;
```

Index.js:

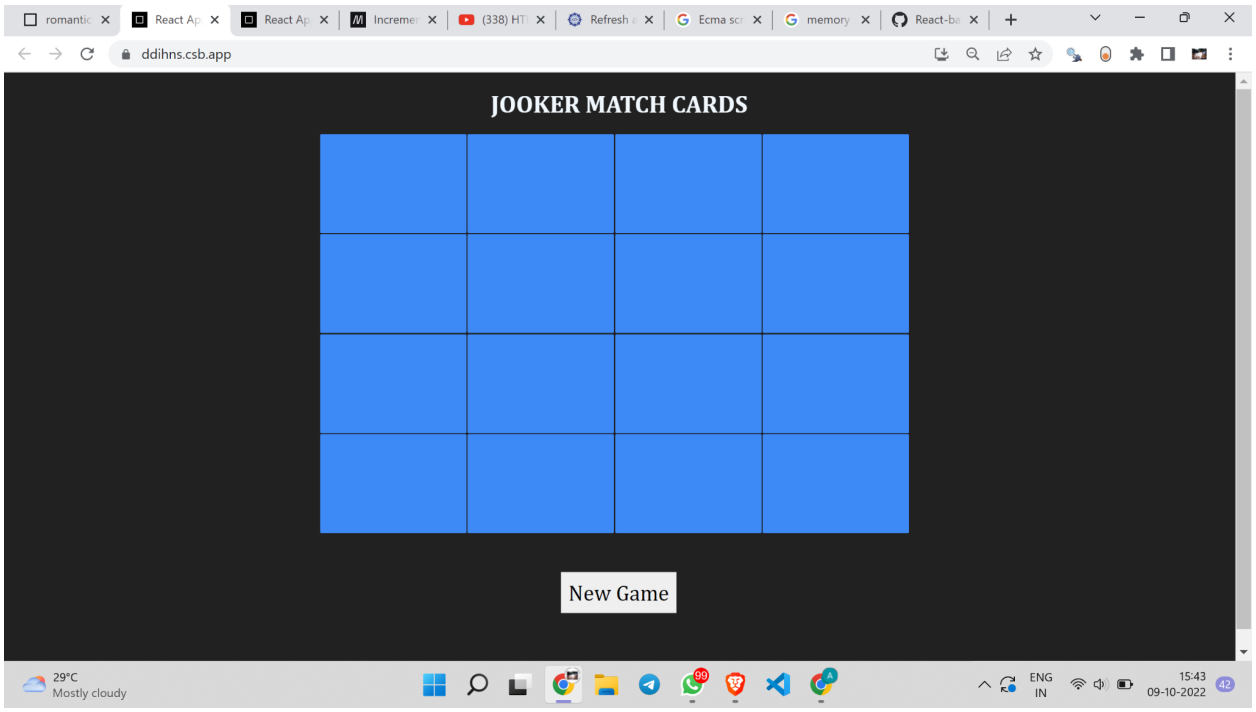
This is the file which is rendered on the webpage. The html file is accessed using the function getElementById and the app function is imported into the root to render

```
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import "./index.css";
import App from "./App";

const rootElement = document.getElementById("root");
const root = createRoot(rootElement);

root.render(
 <StrictMode>
   <App />
 </StrictMode>
);
```

Output:


ddihns.csb.app
JOOKER MATCH CARDS
New Game
29°C
Mostly cloudy
ENG
IN
15:43
09-10-2022