React Project

NAME : KAVIN VELAVAN G

DEPT : BTECH AI & DS

ROLL NO : 20I130

SUBJECT : DEVOPS

KARPAGAM
COLLEGE OF ENGINEERING
Rediscover | Refine | Redefine
Coimbatore - 641 032
Autonomous | Affiliated to Anna University, Chennai
Accredited by NAAC with 'A' Grade | NBA (ECE, EEE, CSE, MECH & IT)

AI
Artificial Intelligence
and Data Science

Introduction :

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It's 'V' in MVC. ReactJS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook.

Concept of the game :

The find box is a simple game and easy to play. I have added a theme and levels in the game .In this game there are 10 levels and 2 themes: dark and light themes.In the first level 2 boxes will appear and suddenly disappear within a second you have to find the box correctly if the chosen box is correct it will appear like green color and otherwise it appears red color. If you pick the wrong box then it will redirect to the first level and you have to play the game from the beginning if the player finishes all the 10 levels then the game ends.

React code for the game :

Step 1 : To create a config folder ,inside that create a levels.json and themes.json files

In the levels.theme write a code

```
[

 {

   "cellCount": 3,

   "memoryCount": 3,

   "fieldSize": 300,

   "space": 1

 },
{
  "cellCount": 4,
  "memoryCount": 3,
  "fieldSize": 300,
  "space": 1
},
{
  "cellCount": 4,
  "memoryCount": 4,
  "fieldSize": 300,
  "space": 1
},
{
  "cellCount": 5,
  "memoryCount": 4,
  "fieldSize": 300,
  "space": 1
},
{
  "cellCount": 5,
  "memoryCount": 5,
  "fieldSize": 350,
  "space": 1
},
{
  "cellCount": 5,
  "memoryCount": 6,
  "fieldSize": 350,
  "space": 1
},
{
  "cellCount": 5,
    "memoryCount": 6,
    "fieldSize": 350,
    "space": 1
  },
  {
    "cellCount": 5,
    "memoryCount": 7,
    "fieldSize": 350,
    "space": 1
  },
  {
    "cellCount": 6,
    "memoryCount": 7,
    "fieldSize": 350,
    "space": 1
  },
  {
    "cellCount": 6,
    "memoryCount": 8,
    "fieldSize": 350,
    "space": 1
  },
  {
    "cellCount": 6,
    "memoryCount": 9,
    "fieldSize": 350,
    "space": 1
  },
  {
    "cellCount": 6,
    "memoryCount": 10,
    "fieldSize": 350,
   "space": 1
 }
]
```

Step 2 : Create a file named the themes.json

```
{
 "lightTheme": {
   "header": {
     "height": "400px",
     "background": "blue"
   },
   "body": {
     "bg": "white",
     "color": "red"
   },
   "cell": {
     "bg": "grey",
     "activeBg": "green",
     "failedBg": "red"
   },
   "loader": {
     "bg": "white"
   }
 },
 "darkTheme": {
   "header": {
     "height": "50px",
     "background": "black"
   },
   "body": {
     "bg": "black",
     "color": "blue"
   },
   "cell": {
     "bg": "blue",
     "activeBg": "yellow",
     "failedBg": "orange"
   },
   "loader": {
     "bg": "black"
   }
 }
}
```

Step 3 : Create a game folder inside that create a component folder inside that create the necessary files
Cell.jsx ,gamefield.jsx , styled.jsx and style.js
In the cell.jsx

```
import React, { memo } from "react";
import styled from "styled-components";

import { getFromTheme } from "../../utils";
import { CORRECT_GUESSED_CELL, HIDDEN_CELL } from "../game.utils";

const CellView = styled.div`
 width: ${({ size }) => size}px;
 height: ${({ size }) => size}px;
 background: ${getFromTheme("cell.bg")};
 margin: ${({ space }) => space}px;
 display: flex;
 justify-content: left;
 align-items: left;
`;
```

```
const ActiveCellView = styled.div`
 width: ${({ width }) => width}%;
 height: 100%;
 background: ${getFromTheme("cell.activeBg")};
 transition: width 0.2s ease;
`;

const FailedCellView = styled.div`
 width: ${({ size }) => size}%;
 height: ${({ size }) => size}%;
 background: ${getFromTheme("cell.failedBg")};
 transition: width 0.2s ease, height 0.2s ease;
`;

export const Cell = memo(function Cell(props) {
 const { id, value, forceShowHidden } = props;

 const isActive =
   (forceShowHidden && value === HIDDEN_CELL) ||
   value === CORRECT_GUESSED_CELL;
 const isFailed = !value;

 return (
   <CellView {...props}>
     <ActiveCellView id={id} width={isActive ? 100 : 0} />
     <FailedCellView id={id} size={isFailed ? 100 : 0} />
   </CellView>
 );
});
```

Step 4 : Create a file named gamefield.jsx

```
import React, { memo, useState, useEffect } from "react";
import styled from "styled-components";

import { HIDDEN_CELL_HIDE, HIDDEN_CELL_SHOW } from "../game.reducer";
import { Cell } from "./Cell";
import { WRONG_GUESSED_CELL, CORRECT_GUESSED_CELL } from
"../game.utils";

const FieldView = styled.div`
 width: 100%;
 height: 100%;
 display: flex;
 flex-wrap: wrap;
 justify-content: space-between;
 margin: 20px 0;
 opacity: ${({ animationState }) => animationState};
 transform: scale(${({ animationState }) => animationState});
 transition: opacity 0.2s ease, transform 0.3s ease;
`;

export const Field = memo(function Field({
 fieldSize = 0,
 cellCount = 0,
 space = 0,
 field = [],
 hiddenCells = [],
 level = 0,
 showHidden = false,
 dispatch,
 updateLevel,
 visible
```

```
}) {
 const cellSize = fieldSize / cellCount - space;

 const { gameField, onCellClick } = useGameField(
   field,
   hiddenCells,
   updateLevel
 );

 useEffect(
   () => {
     dispatch({ type: HIDDEN_CELL_SHOW });
     setTimeout(() => dispatch({ type: HIDDEN_CELL_HIDE }), 1500);
   },
   [level]
 );

 return (
   <FieldView
     animationState={visible ? 1 : 0}
     onClick={!showHidden ? onCellClick : null}
   >
     {gameField.map((cellValue, i) => (
       <Cell
         size={cellSize}
         space={space}
         key={i}
         id={i}
         value={cellValue}
         forceShowHidden={showHidden}
       />
     ))}
   </FieldView>
 );
});

function useGameField(field, hiddenCells, updateLevel) {
 const [gameField, setField] = useState(field);
 const [gameHiddenCells, setHidden] = useState(hiddenCells);

 function onCellClick({ target }) {
   const id = Number(target.id);

   if (hiddenCells.includes(id)) {
     const updatedField = gameField.map((e, i) =>
       i === id ? CORRECT_GUESSED_CELL : e
     );
     const updatedHidden = gameHiddenCells.filter(e => e !== id);

     setField(updatedField);
     setHidden(updatedHidden);

     return !updatedHidden.length && setTimeout(updateLevel, 1000);
   }

   const updatedField = gameField.map((e, i) =>
     i === id ? WRONG_GUESSED_CELL : e
   );
   setField(updatedField);

   return setTimeout(updateLevel, 1000, true);
 }

 return { gameField, onCellClick };
```

```
}
```

Step 5 : create a file named the style.js

```
import styled from "styled-components";

export const GameView = styled.div`
 width: 100%;
 height: 100%;
 display: flex;
 justify-content: center;
 align-items: center;
 margin: 5px 0;
`;

export const GameFieldView = styled.div`
 width: ${({ fieldSize, cellCount, space }) =>
   fieldSize + cellCount * space}px;
 height: ${({ fieldSize, cellCount, space }) =>
   fieldSize + cellCount * space}px;
 margin: 20px 0;
`;

export const SwitchView = styled.div`
 display: flex;
 width: 100%;
 justify-content: space-between;
`;
```

Step 6 : In the file named game.reducer.js

```
import { merge } from "ramda";

import levels from "../config/levels";

export const NEW_LEVEL = "level/new";
export const HIDDEN_CELL_HIDE = "hidden/hide";
export const HIDDEN_CELL_SHOW = "hidden/show";
export const FIELD_HIDE = "field/hide";
export const FIELD_SHOW = "field/show";
export const RESET_LEVEL = "level/reset";

const START_LEVEL = 0;

export const initialState = {
 level: START_LEVEL,
 showHidden: true,
 showField: false,
 levelConfig: levels[START_LEVEL]
};

export function GameReducer(state, action) {
 switch (action.type) {
   case NEW_LEVEL:
     return merge(state, {
       level: action.level,
       levelConfig: levels[action.level]
     });
   case HIDDEN_CELL_SHOW:
     return merge(state, { showHidden: true });
   case HIDDEN_CELL_HIDE:
     return merge(state, { showHidden: false });
   case FIELD_HIDE:
     return merge(state, { showField: false });
   case FIELD_SHOW:
     return merge(state, { showField: true });
   case RESET_LEVEL:
     return merge(initialState, { levelConfig: {
...levels[START_LEVEL] } });
   default:
     return state;
 }
}
```

Step 7 : In the file named game.utils.js

```
export function generateGameField(cellCount, memoryCount) {
  const cellsIndexes = [...Array(cellCount * cellCount)].map((_, i) =>
i);
  const field = [...cellsIndexes].fill(1);
  const hiddenCells = [];

  for (let i = 0; i < memoryCount; i++) {
    const rNum = Math.floor(Math.random() * cellsIndexes.length);
    const toChange = cellsIndexes.splice(rNum, 1).pop();

    hiddenCells.push(toChange);
    field[toChange] = 2;
  }

  return {
    field,
    hiddenCells
  };
}
```

```
export const WRONG_GUESSED_CELL = 0;
export const CORRECT_GUESSED_CELL = 3;
export const CELL = 1;
export const HIDDEN_CELL = 2;
```

Step 8 : In the file named index.jsx

```
import React, { memo, useReducer, useMemo, useEffect } from
"react";

import { Field } from "./components/GameField";
import { GameFieldView, GameView, SwitchView } from
"./components/Styled";
import {
 GameReducer,
 initialState,
 NEW_LEVEL,
 FIELD_HIDE,
 FIELD_SHOW,
 RESET_LEVEL
} from "./game.reducer";
import { generateGameField } from "./game.utils";
import Switch from "rc-switch";

import "rc-switch/assets/index.css";

function Game({ toggleTheme }) {
 const [{ level, showHidden, showField, levelConfig }, dispatch] =
useReducer(
   GameReducer,
   initialState
```

```
);

const { cellCount, memoryCount } = levelConfig;

const { field, hiddenCells } = useMemo(
  () => generateGameField(cellCount, memoryCount),
  [levelConfig]
);

useEffect(() => setTimeout(dispatch, 500, { type: FIELD_SHOW }), [
  levelConfig
]);

function updateLevel(shouldReset) {
  dispatch({ type: FIELD_HIDE });
  setTimeout(dispatch, 500, {
    type: shouldReset ? RESET_LEVEL : NEW_LEVEL,
    level: level + 1
  });
}

return (
  <GameView>
    <GameFieldView {...levelConfig}>
      <SwitchView>
        <div>Level: {level}</div>
        <div>
          Theme mode: <Switch onClick={toggleTheme} />
        </div>
      </SwitchView>
      <Field
        {...levelConfig}
```

```
         levelConfig={levelConfig}
         visible={showField}
         key={field}
         level={level}
         field={field}
         hiddenCells={hiddenCells}
         dispatch={dispatch}
         showHidden={showHidden}
         updateLevel={updateLevel}
       />
     </GameFieldView>
   </GameView>
 );
}

export default memo(Game);
```

Step 9 : In the file named App.jsx

```
import React, { useState } from 'react';
import { ThemeProvider, createGlobalStyle } from
'styled-components';
import { getFromTheme } from './utils';
import './index.css';

import Game from './game';
import themes from './config/themes.json';

function App() {
 const [themeName, toggleTheme] = useTheme('darkTheme');

 const GlobalStyle = createGlobalStyle`
```

```
   body {
       background: ${getFromTheme('body.bg')};
       color: ${getFromTheme('body.color')};
       transition: background .3s ease;
   }
 `;

 return (
   <ThemeProvider theme={themes[themeName]}>
     <React.Fragment>
       <GlobalStyle />
       <Game toggleTheme={toggleTheme} />
     </React.Fragment>
   </ThemeProvider>
 );
}

function useTheme(defaultThemeName) {
 const [themeName, setTheme] = useState(defaultThemeName);

 function switchTheme(name) {
   setTheme(themeName === 'darkTheme' ? 'lightTheme' :
'darkTheme');
 }

 return [themeName, switchTheme];
}


export default App;
```

Step 10 : In the file named index.css

```
* {
 box-sizing: border-box;
}

body {
 margin: 0;
 padding: 0;
 font-family: -apple-system, BlinkMacSystemFont, "Segoe UI",
"Roboto", "Oxygen",
   "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica
Neue",
   sans-serif;
 -webkit-font-smoothing: antialiased;
 -moz-osx-font-smoothing: grayscale;
}

.no-select {
 -webkit-touch-callout: none;
 -webkit-user-select: none;
 -khtml-user-select: none;
 -moz-user-select: none;
 -ms-user-select: none;
 user-select: none;
 -webkit-tap-highlight-color: transparent;
}
```

In the index.js

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";
```

```
ReactDOM.render(<App />, document.getElementById("root"));
```

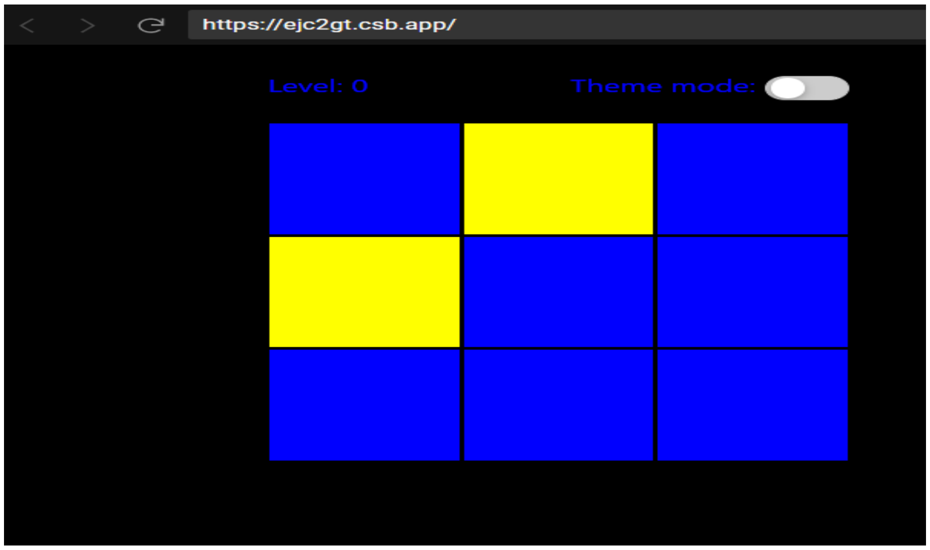
In the utils.js

```
import { path } from "ramda";

export function getFromTheme(themePath = "") {
 return function getFromThemeProps(props = {}) {
   return path(themePath.split("."), props.theme);
 };
}

export function wait(ms) {
 return new Promise(resolve => setTimeout(resolve, ms));
}
```

In the package.json file

```
{
 "name": "new",
 "version": "1.0.0",
 "description": "",
 "keywords": [],
 "main": "src/index.js",
 "dependencies": {
   "ramda": "0.26.1",
   "rc-switch": "1.8.0",
   "react": "16.7.0-alpha.0",
   "react-dom": "16.7.0-alpha.0",
   "react-scripts": "2.0.3",
   "styled-components": "4.1.2"
 },
 "devDependencies": {},
 "scripts": {
   "start": "react-scripts start",
   "build": "react-scripts build",
   "test": "react-scripts test --env=jsdom",
   "eject": "react-scripts eject"
 },
 "browserslist": [">0.2%", "not dead", "not ie <= 11", "not op_mini
all"]
}
```

Output :



Link for game : https://ejc2gt.csb.app/

Result :

Thus the Find Box game is developed and code is executed sucessfully.