**13.Reactjs\_Hol**

**1. Explain Various Ways of Conditional Rendering**

Conditional rendering in React means displaying components or elements based on conditions such as state or props. It can be implemented using if-else, ternary operators, logical AND (&&), or element variables.

**2. Explain How to Render Multiple Components**

Multiple components can be rendered together by returning them inside a common parent like <div> or <React.Fragment>. This helps build complex UIs from smaller reusable blocks.

**3. Define List Component**

A list component is used to display a set of similar items dynamically using array methods like map(). It helps present data like users, flights, or products in a clean, structured way.

**4. Explain About Keys in React Applications**

Keys are unique identifiers used by React to track list elements during updates. They help React optimize rendering by identifying which elements changed, were added, or removed.

**5. Explain How to Extract Components with Keys**

When rendering lists, each item can be extracted into a separate component and passed a unique key prop. This improves code readability and React’s ability to efficiently update the UI.

**6. Explain React Map and map() Function**

The map() function in React is used to iterate over arrays and return JSX elements. It enables dynamic rendering of lists based on data and should include a key for each item.

Lab:

Create a React App named “bloggerapp” in with 3 components.

1. Book Details
2. Blog Details
3. Course Details

Implement this with as many ways possible of Conditional Rendering.

**Book.js**

import React from 'react';

function BookDetails() {

  return (

    <div>

      <h2>📚 Book Details</h2>

      <ul>

        <li>Title: React Essentials</li>

        <li>Author: Dan Abramov</li>

        <li>Pages: 250</li>

      </ul>

    </div>

  );

}

export default BookDetails;

**BlogDetails.js**

import React from 'react';

function BlogDetails() {

  return (

    <div>

      <h2>📝 Blog Details</h2>

      <ul>

        <li>Title: React vs Angular</li>

        <li>Author: Sarah</li>

        <li>Likes: 120</li>

      </ul>

    </div>

  );

}

export default BlogDetails;

**CourseDetails.js**

import React from 'react';

function CourseDetails() {

  return (

    <div>

      <h2>🎓 Course Details</h2>

      <ul>

        <li>Course: Full Stack Web Development</li>

        <li>Duration: 3 Months</li>

        <li>Instructor: John Doe</li>

      </ul>

    </div>

  );

}

export default CourseDetails;

**App.js**

import React, { useState } from 'react';

import BookDetails from './BookDetails';

import BlogDetails from './BlogDetails';

import CourseDetails from './CourseDetails';

function App() {

  const [active, setActive] = useState("book");

  // Using element variable

  let content;

  if (active === "book") {

    content = <BookDetails />;

  } else if (active === "blog") {

    content = <BlogDetails />;

  } else {

    content = <CourseDetails />;

  }

  return (

    <div style={{ textAlign: 'center', padding: '20px' }}>

      <h1>📖 Blogger App</h1>

      {/\* Buttons to change component \*/}

      <div style={{ marginBottom: '20px' }}>

        <button onClick={() => setActive("book")}>Book</button>{" "}

        <button onClick={() => setActive("blog")}>Blog</button>{" "}

        <button onClick={() => setActive("course")}>Course</button>

      </div>

      {/\* 1. Using if-else via element variable \*/}

      {content}

      {/\* 2. Using Ternary Operator \*/}

      {/\* Uncomment below to test \*/}

      {/\*

      {active === "book" ? <BookDetails /> : <CourseDetails />}

      \*/}

      {/\* 3. Using Logical && \*/}

      {/\* Uncomment below to test \*/}

      {/\*

      {active === "blog" && <BlogDetails />}

      \*/}

    </div>

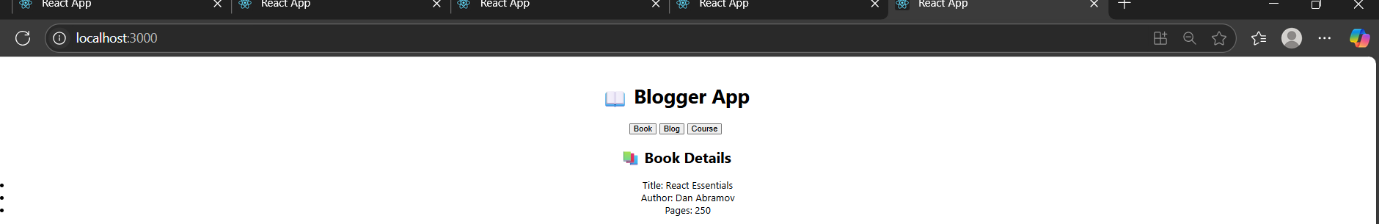
  );

}

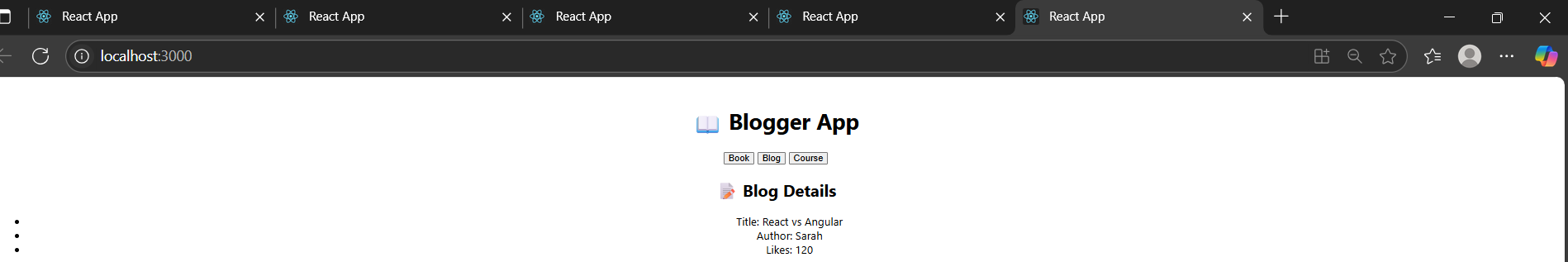
export default App;

Output:

BookDetails



Blog details



Course Details