**Explain various ways of conditional rendering**

Conditional rendering in React allows you to show different components or elements based on certain conditions or application states. Instead of rendering everything at once, you can control what appears on the screen. The most common methods include:

* **if-else Statements**: A standard JavaScript if statement can be used within a component's render logic to return different JSX blocks. This is useful for more complex conditional logic.
* **Ternary Operator (condition ? true : false)**: This is a concise, inline way to decide between two rendering options directly within your JSX.
* **Logical && Operator (condition && expression)**: This is perfect for rendering an element only if a condition is true. If the condition is false, it renders nothing.

**Explain how to render multiple components**

Rendering multiple components is a core feature of React's component-based architecture. You can create a "parent" or "container" component that imports and arranges several "child" components. This allows you to build complex UIs by combining smaller, reusable pieces.

For example, a main

App component can render a Course Details component, a Book Details component, and a Blog Details component all at once to form a complete page layout

// Inside a parent component's return statement

<div>

<CourseDetails />

<BookDetails />

<BlogDetails />

</div>

**Define list component**

**list component** is a component specifically designed to render a collection or list of similar items. It typically takes an array of data (like a list of books or courses) as a prop and uses a loop, most commonly the

map() function, to generate a UI element for each item in the array

**Explain about keys in React applications**

**Keys** are a special string attribute you need to include when creating lists of elements in React. They serve as a stable identity for each element in the list.

* **Purpose**: Keys help React identify which items have changed, been added, or been removed. This allows React to efficiently update the user interface by re-rendering only the necessary elements instead of the entire list.
* **Requirement**: Keys must be **unique** among sibling elements in a list. It is best practice to use a stable and unique identifier from your data, like a product ID or a database primary key (e.g.,

book.id)

**Explain how to extract components with keys**

When you extract the logic for rendering a list item into its own separate component, the

key must still be applied to the component's tag inside the map() loop. The key should not be placed inside the child component itself.

**Correct Usage:**

JavaScript

// The key is on the child component in the map loop.

function BookList({ books }) {

return (

<ul>

{books.map((book) => (

<BookListItem key={book.id} book={book} />

))}

</ul>

);

}

**Explain React Map, map() function**

The map() function is a standard JavaScript array method that is fundamental to rendering lists in React. It transforms an array of data into an array of JSX elements.

The function iterates over every item in an array, performs an operation on that item (like wrapping it in JSX), and returns a new array containing the results. In React, this new array of JSX elements is then rendered to the DOM.

**Example:** The following code uses

map() to transform an array of books into an array of <div> elements.

JavaScript

const bookdet = (

<ul>

{props.books.map((book) => (

<div key={book.id}>

<h3>{book.bname}</h3>

<h4>{book.price}</h4>

</div>

))}

</ul>

);

**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**.

**Components**

**BookDetails.js**

import React from 'react';

const BookDetails = ({ books }) => {

  return (

    <div>

      <h2>Book Details</h2>

      {books.map((book) => (

        <div key={book.id}>

          <h3>{book.bname}</h3>

          <h4>{book.price}</h4>

        </div>

      ))}

    </div>

  );

};

export default BookDetails;

**BlogDetails.js**

import React from 'react';

const BlogDetails = ({ blogs }) => {

  return (

    <div>

      <h2>Blog Details</h2>

      {blogs.map((blog) => (

        <div key={blog.id}>

          <h3>{blog.title}</h3>

          <p>{blog.author}</p>

          <p>{blog.description}</p>

        </div>

      ))}

    </div>

  );

};

export default BlogDetails;

**CourseDetails.js**

import React from 'react';

const CourseDetails = ({ courses }) => {

  return (

    <div>

      <h2>Course Details</h2>

      {courses.map((course) => (

        <div key={course.id}>

          <h3>{course.name}</h3>

          <p>{course.date}</p>

        </div>

      ))}

    </div>

  );

};

export default CourseDetails;

**Data.js**

export const books = [

  { id: 101, bname: 'Master React', price: 670 },

  { id: 102, bname: 'Deep Dive into Angular 11', price: 800 },

  { id: 103, bname: 'Mongo Essentials', price: 450 },

];

export const courses = [

    {id: 1, name: 'Angular', date: '4/5/2021'},

    {id: 2, name: 'React', date: '6/3/20201'}

];

export const blogs = [

    {id: 1, title: 'React Learning', author: 'Stephen Biz', description: 'Welcome to learning React!'},

    {id: 2, title: 'Installation', author: 'Schewzdenier', description: 'You can install React from npm.'}

];

**App.js**

import React, { useState } from 'react';

import BookDetails from './components/BookDetails';

import BlogDetails from './components/BlogDetails';

import CourseDetails from './components/CourseDetails';

import { books, blogs, courses } from './Data';

import './App.css';

function App() {

  const [showBooks, setShowBooks] = useState(true);

  const [showCourses, setShowCourses] = useState(true);

  const [showBlogs, setShowBlogs] = useState(true);

  return (

    <div className="App">

      <header className="App-header">

        <h1>Blogger App</h1>

        <div className="buttons-container">

          <button onClick={() => setShowCourses(!showCourses)}>

            {showCourses ? 'Hide' : 'Show'} Course Details

          </button>

          <button onClick={() => setShowBooks(!showBooks)}>

            {showBooks ? 'Hide' : 'Show'} Book Details

          </button>

          <button onClick={() => setShowBlogs(!showBlogs)}>

            {showBlogs ? 'Hide' : 'Show'} Blog Details

          </button>

        </div>

      </header>

      <div className="main-content">

        {/\* Method 1: Ternary Operator \*/}

        {showCourses ? (

          <div className="section">

            <CourseDetails courses={courses} />

          </div>

        ) : null}

        {/\* Method 2: Logical && Operator \*/}

        {showBooks && (

          <div className="section">

            <BookDetails books={books} />

          </div>

        )}

        {/\* Method 3: Using an if condition (via a function) \*/}

        {(() => {

          if (showBlogs) {

            return (

              <div className="section">

                <BlogDetails blogs={blogs} />

              </div>

            );

          }

        })()}

      </div>

    </div>

  );

}

export default App;

**App.css**

.App {

  text-align: center;

}

.App-header {

  background-color: #f0f0f0;

  padding: 20px;

  margin-bottom: 20px;

}

.buttons-container button {

  margin: 0 10px;

  padding: 10px 15px;

  font-size: 16px;

  cursor: pointer;

}

.main-content {

  display: flex;

  justify-content: space-around;

  text-align: left;

}

.section {

  padding: 0 20px;

}

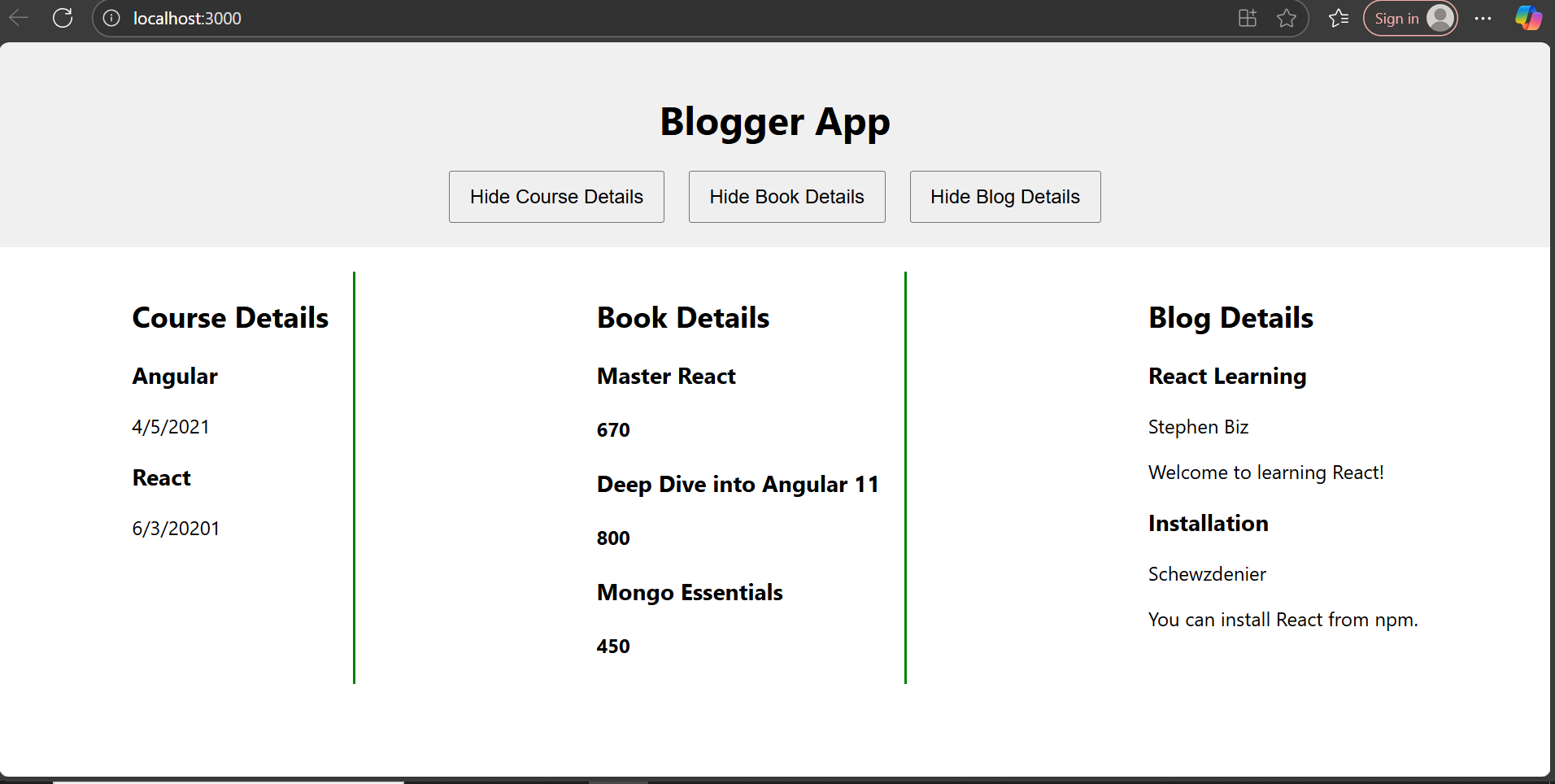
/\* Vertical lines \*/

.section:not(:last-child) {

  border-right: 2px solid green;

}

**Output:**

****

