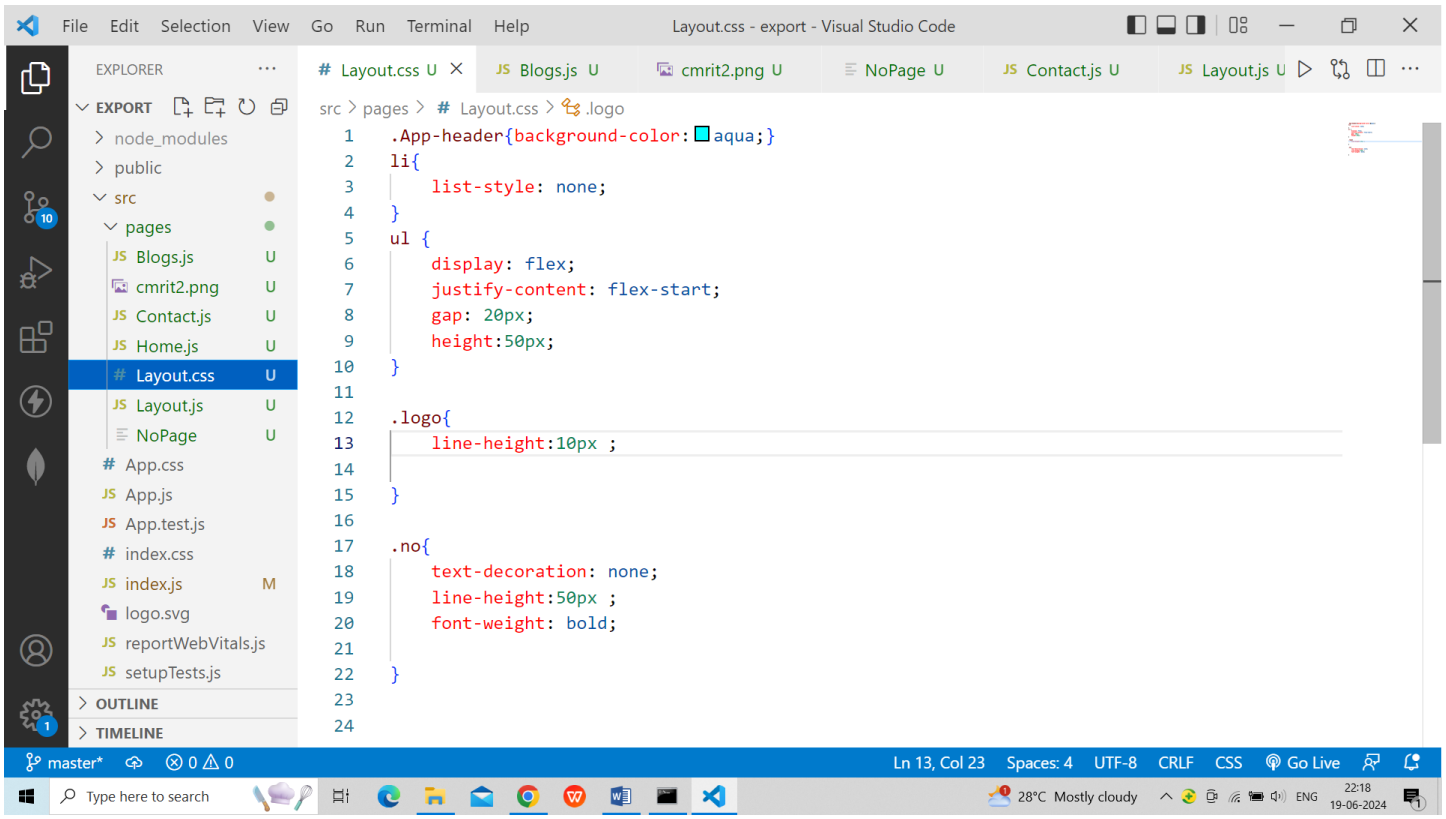
Exp-11
Design Single Page Application with different menu items using react.



open react project week11and open src folder Create one folder name pages
create this files **Layout.js,Layout.css,Home.js,Contact.js, Blogs.js and NoPage.js**

- **Layout.js**

```
import './Layout.css';
import { Outlet, Link } from "react-router-dom";

const Layout = () => {
  return (
    <>

      <nav className="App-header">
        <ul>
        <li className="logo">
          <h1>MyWorld</h1>
        </li>
          <li>
            <Link to="/" className="no">Home</Link>
          </li>
          <li>
            <Link to="/blogs" className="no">Blogs</Link>
          </li>
          <li>
            <Link to="/contact" className="no">Contact</Link>
          </li>
        </ul>
      </nav>

      <Outlet />

```

```
    </>
  )
};

export default Layout;
```

- **Layout.css**

```
.App-header{background-color:aqua;}
li{
    list-style: none;
}
ul {
    display: flex;
    justify-content: flex-start;
    gap: 20px;
    height:50px;
}

.logo{
    line-height:10px ;

}

.no{
    text-decoration: none;
    line-height:50px ;
    font-weight: bold;

}

```

- **Home.js**

```
const Home = () => {
    return <h1>Home</h1>;
  };

  export default Home;
```

- **Contact.js**

```
const Contact = () => {
  return <h1>Contact Me</h1>;
};

export default Contact;
```

- **Blogs.js**

```
const Blogs = () => {
  return <h1>Blog Articles</h1>;
};

export default Blogs;
```

- **NoPage.js**

```
const NoPage = () => {
  return <h1>404</h1>;
};

export default NoPage;
```

Open src folder write code inside index.js

- **index.js**

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';

import reportWebVitals from './reportWebVitals';
import { BrowserRouter, Routes, Route } from "react-router-dom";
import Layout from "./pages/Layout";
import Home from "./pages/Home";
import Blogs from "./pages/Blogs";
import Contact from "./pages/Contact";
import NoPage from "./pages/NoPage";

const root = ReactDOM.createRoot(document.getElementById('root'));
export default function App1() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Layout />}>
          <Route index element={<Home />} />
          <Route path="blogs" element={<Blogs />} />
          <Route path="contact" element={<Contact />} />
          <Route path="*" element={<NoPage />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

root.render(
  <React.StrictMode>
    <App1 />

  </React.StrictMode>
);

reportWebVitals();
```

Output:-

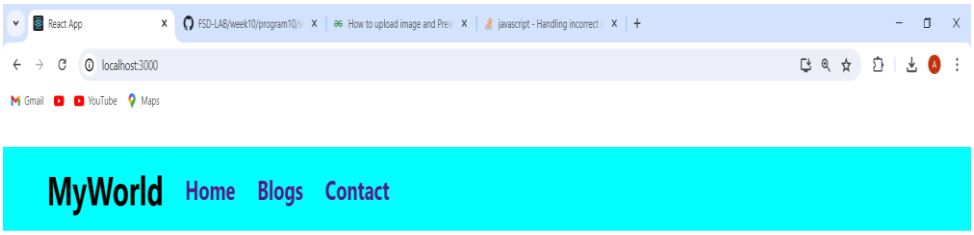


Fig.1.Home page

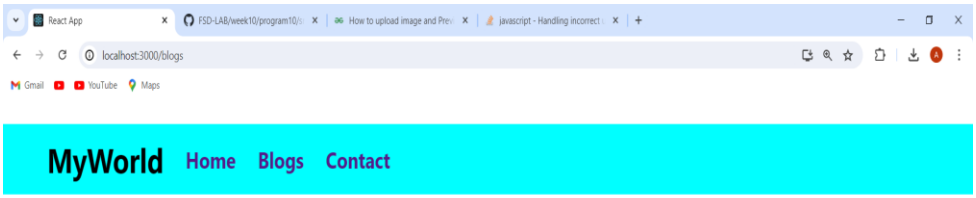


Fig.2.Blogs page

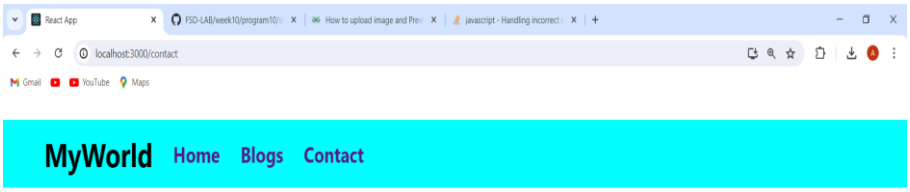


Fig.3.Contact page

VIVA QUESTIONS:

1.How does react work?

Ans: One of the key innovations of React is the Virtual DOM. Instead of directly manipulating the browser's Document Object Model (DOM), which can be slow, React applications work with a lightweight in-memory representation of the DOM called the Virtual DOM

React Hooks:

Introduced in React 16.8, hooks allow functional components to use state and other React features without writing a class. The most commonly used hooks are useState for managing component state and useEffect for performing side effects in functional components.

2.What are props in react ?

Ans:

```
<MyComponent myProp="value" />
class MyComponent extends React.Component {
  render() {
    return <p>{this.props.myProp}</p>;
  }
}
function MyComponent(props) {
  return <p>{props.myProp}</p>;
}
function MyComponent({ myProp }) {
  return <p>{myProp}</p>;
}
function ParentComponent(props) {
  return <div>{props.children}</div>;
}

// Usage
<ParentComponent>
  <ChildComponent />
</ParentComponent>
data.map((item) => <Component key={item.id} {...item} />)
```

3.Difference between VirtualDOM and RealDOM ?

Real DOM
Definition: The Real DOM is the browser's programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content.
Manipulation: Changes to the Real DOM are slow. The Real DOM is stateful; any change in the DOM can cause page re-renders, which are expensive in terms of performance.
Updates: Direct manipulation or updates to the Real DOM can be inefficient, especially with frequent updates, because each change directly affects the elements on the web page, leading to potential performance issues.
Operations Cost: Operations on the Real DOM are expensive. The process of re-rendering or repainting the page due to DOM manipulations can be heavy, especially for complex applications.
Direct Interaction: JavaScript directly manipulates the Real DOM for adding or removing elements, changing content, etc.

Virtual DOM
Definition: The Virtual DOM is a lightweight copy or abstraction of the Real DOM. It is a concept implemented by libraries like React to improve web application performance.
Manipulation: Changes to the Virtual DOM are fast. The Virtual DOM can undergo frequent updates without the performance costs associated with the Real DOM because these changes do not immediately affect the actual DOM on the web page.
Updates: The Virtual DOM allows for the batching of updates. Instead of updating the Real DOM after every single change, the Virtual DOM can aggregate several changes and apply them simultaneously in a single update cycle.
Operations Cost: Operations on the Virtual DOM are inexpensive. Since the Virtual DOM is just a lightweight copy,

manipulating it doesn’t have the same performance implications as manipulating the Real DOM.
Reconciliation Algorithm: Libraries that use the Virtual DOM, like React, implement a "diffing" algorithm. This algorithm compares the current state of the Virtual DOM with the previous state and calculates the most efficient way to update the Real DOM to match the Virtual DOM. This selective rendering minimizes direct manipulation of the Real DOM, thereby improving performance.

4**. Describe MVC in reference to angular ?**

Ans: The Model-View-Controller (MVC) architecture is a design pattern that separates an application into three interconnected components: the model, the view, and the controller. This separation helps manage complex applications by dividing them into smaller, more manageable parts, each responsible for specific aspects of the application. Although Angular is often associated with a slightly different architectural pattern (Model-View-ViewModel or MVVM), it still embodies the core principles of MVC in how it structures applications.

5. **Define stateful component ?**

Ans: A stateful component, in the context of web development frameworks such as React, Angular, or Vue, is a component that maintains its own state. The state of a component is an object that holds some information that may change over the lifetime of the component. Stateful components are aware of changes in this information and can react to those changes by updating the UI accordingly. These components keep track of changes in data that affect the render output, making them crucial for creating dynamic and interactive user interfaces.