**Practical - 1**

|  |  |
| --- | --- |
| Practical 1 | Create a new react application which shows the implementation of Table |
| Code: | Create a new react app:  create-react-app myapp  cd myapp  npm start  Next, install the bootstrap and react-bootstrap library using below command,  Npm install –save boostrap react-bootstrap  Next, open *App.css* (src/App.css) and remove all CSS classes.  Next, create a simple table component, *SimpleTable* (src/SimpleTable.js) and  Render a table as shown below –  **App.js**  import { Table } from 'react-bootstrap';  function SimpleTable() {  return (  <Table striped bordered hover>  <thead>  <tr>  <th>#</th>  <th>Name</th>  <th>Age</th>  <th>Email</th>  </tr>  </thead>  <tbody>  <tr>  <td>1</td>  <td>John</td>  <td>25</td>  <td>john.example@tutorialspoint.com</td>  </tr>  <tr>  <td>2</td>  <td>Peter</td>  <td>15</td>  <td>peter.example@tutorialspoint.com</td>  </tr>  <tr>  <td>3</td>  <td>Olivia</td>  <td>23</td>  <td>olivia.example@tutorialspoint.com</td>  </tr>  </tbody>  </Table>  );  }  export default SimpleTable;  **index.js**  import './App.css';  import "bootstrap/dist/css/bootstrap.min.css";  import SimpleTable from './SimpleTable';  function App() {  return (  <div className="container">  <div style={{ padding: "10px" }}>  <div>  <SimpleTable />  </div>  </div>  </div>  );  }  export default App; |
| output |  |

Practical -**2**

|  |  |
| --- | --- |
| Practical 2 | Create a new react application which shows the implementation of Counter. |
| Code | **App.js**  import React, { useState } from 'react';  function Counter() {  const [count, setCount] = useState(0); // Initialize count state to 0  const increment = () => {  setCount(count + 1); // Update count state  };  const decrement = () => {  setCount(count - 1); // Update count state  };  return (  <div>  <h2>Count: {count}</h2>  <button onClick={increment}>Increment</button>  <button onClick={decrement}>Decrement</button>  </div>  );  }  export default Counter;  **index.js**  import React from 'react';  import ReactDOM from 'react-dom/client';  import './index.css';  import Counter from './App';  import reportWebVitals from './reportWebVitals';  const root = ReactDOM.createRoot(document.getElementById('root'));  root.render(  <Counter />  );  reportWebVitals(); |
| Output |  |

**Practical - 3**

|  |  |
| --- | --- |
| Practical 3 | Create a new react application which shows the implementation of List. |
| Code | **App.js**  import React from 'react';  function ShoppingList() {  const products = [  { id: 1, title: 'Apples' },  { id: 2, title: 'Bananas' },  { id: 3, title: 'Milk' },  ];  return (  <div>  <h2>Shopping List</h2>  <ul>  {products.map(product => (  <li key={product.id}>{product.title}</li>  ))}  </ul>  </div>  );  }  export default ShoppingList;  **index.js**  import React from 'react';  import ReactDOM from 'react-dom/client';  import './index.css';  import ShoppingList from './App';  import reportWebVitals from './reportWebVitals';  const root = ReactDOM.createRoot(document.getElementById('root'));  root.render(  <ShoppingList />  );  reportWebVitals(); |
| Output |  |

**Practical - 4**

|  |  |
| --- | --- |
| Practical 4 | Create a new React application which shows the implementation of a Router which helps to route to pages. |
| Code | To add React Router to your application, run this in the terminal from the root directory of the application:  npm install react-router-dom  **Create React App using following command:**  npx create-react-app routingeg  Within the src folder, we'll create a folder named pages with several files:  src\:   * Layout.js * Home.js * Blogs.js * Contact.js * NoPage.js   Each file will contain a very basic React component.  Use React Router to route to pages based on URL:  **App.js:**  import { Route, Routes, BrowserRouter } from "react-router-dom";  import "./App.css";  import Home from "./Home";  import About from "./About";  import Contact from "./Contact";  import Navigate from "./Navigate";  function App() {  return (  <div className="App">  <BrowserRouter>  <Routes>  <Route path="/" element={<Navigate />}>  <Route index element={<Home />} />  <Route path="about" element={<About />} />  <Route path="contact" element={<Contact />} />  </Route>  </Routes>  </BrowserRouter>  </div>  );  }  export default App;  **Layout.js**  import { Outlet, Link } from "react-router-dom";  const Layout = () => {  return (  <>  <nav>  <ul>  <li>  <Link to="/">Home</Link>  </li>  <li>  <Link to="/blogs">Blogs</Link>  </li>  <li>  <Link to="/contact">Contact</Link>  </li>  </ul>  </nav>  <Outlet />  </>  );  };  export default Layout;  **Home.js**  const Home = () => {  return <h1>Home</h1>;  };  export default Home;  **Blogs.js**  const Blogs = () => {  return <h1>Blog Articles</h1>;  };  export default Blogs;  **contact.js**  const Contact = () => {  return <h1>Contact Me</h1>;  };  export default Contact;  **NoPage.js**  const NoPage = () => {  return <h1>404</h1>;  };  export default NoPage;  **index.js**  import React from 'react';  import ReactDOM from 'react-dom/client';  import './index.css';  import App from './App';  import reportWebVitals from './reportWebVitals';  const root = ReactDOM.createRoot(document.getElementById('root'));  root.render(  <React.StrictMode>  <App />  </React.StrictMode>  );  reportWebVitals(); |
| Output |  |

**Practical - 5**

|  |  |
| --- | --- |
| Practical 5 | Create and validate the user form in React. |
| Code | **Userform.js**  import React, { useState } from 'react';  function UserForm() {  const [formData, setFormData] = useState({  username: '',  email: '',  password: '',  });  const [errors, setErrors] = useState({});  const handleChange = (e) => {  const { name, value } = e.target;  setFormData((prevData) => ({ ...prevData, [name]: value }));  };  const validateForm = () => {  let newErrors = {};  if (!formData.username) {  newErrors.username = 'Username is required';  }  if (!formData.email) {  newErrors.email = 'Email is required';  } else if (!/\S+@\S+\.\S+/.test(formData.email)) {  newErrors.email = 'Email address is invalid';  }  // Add more validation rules for password, etc.  setErrors(newErrors);  return Object.keys(newErrors).length === 0; // Return true if no errors  };  const handleSubmit = (e) => {  e.preventDefault(); // Prevent default form submission  if (validateForm()) {  // Form is valid, proceed with submission (e.g., API call)  console.log('Form submitted:', formData);  } else {  console.log('Form has errors');  }  };  return (  <form onSubmit={handleSubmit}>  <div>  <label htmlFor="username">Username:</label>  <input  type="text"  id="username"  name="username"  value={formData.username}  onChange={handleChange}  />  {errors.username && <p style={{ color: 'red' }}>{errors.username}</p>}  </div>  <div>  <label htmlFor="email">Email:</label>  <input  type="email"  id="email"  name="email"  value={formData.email}  onChange={handleChange}  />  {errors.email && <p style={{ color: 'red' }}>{errors.email}</p>}  </div>  {/\* Add password field and other inputs similarly \*/}  <button type="submit">Register</button>  </form>  );  }  export default UserForm;  **index.js**  import React from 'react';  import ReactDOM from 'react-dom/client';  import './index.css';  import reportWebVitals from './reportWebVitals';  import UserForm from './UserForm';  const root = ReactDOM.createRoot(document.getElementById('root'));  root.render(  <UserForm />  );  reportWebVitals(); |
| Output |  |

**Practical – 6**

|  |  |
| --- | --- |
| Practical 6 | Create Chat application by using Socket.IO |
| code | Server.js  // server/server.js      const express = require('express');      const http = require('http');      const socketIo = require('socket.io');      const cors = require('cors');      const app = express();      const server = http.createServer(app);      const io = socketIo(server, {          cors: {              origin: '\*', // Allow all origins for development, restrict in production              methods: ['GET', 'POST']          }      });      app.use(cors()); // Use CORS middleware for Express routes if needed      io.on('connection', (socket) => {          console.log('A user connected:', socket.id);          socket.on('message', (message) => {              console.log('Message received:', message);              io.emit('message', message); // Broadcast message to all connected clients          });          socket.on('disconnect', () => {              console.log('User disconnected:', socket.id);          });      });      const PORT = process.env.PORT || 5000;      server.listen(PORT, () => {          console.log(`Server running on port ${PORT}`);      });  App.js(client)     // client/src/App.js      import React, { useState, useEffect } from 'react';      import io from 'socket.io-client';      const socket = io('http://localhost:5000'); // Connect to your server      function App() {          const [message, setMessage] = useState('');          const [messages, setMessages] = useState([]);          useEffect(() => {              socket.on('message', (msg) => {                  setMessages((prevMessages) => [...prevMessages, msg]);              });              return () => {                  socket.off('message');              };          }, []);          const sendMessage = (e) => {              e.preventDefault();              if (message.trim()) {                  socket.emit('message', message);                  setMessage('');              }          };          return (              <div>                  <h1>Real-time Chat</h1>                  <div>                      {messages.map((msg, index) => (                          <p key={index}>{msg}</p>                      ))}                  </div>                  <form onSubmit={sendMessage}>                      <input                          type="text"                          value={message}                          onChange={(e) => setMessage(e.target.value)}                          placeholder="Type a message..."                      />                      <button type="submit">Send</button>                  </form>              </div>          );      }      export default App; |
| Output |  |

**Practical - 7**

|  |  |
| --- | --- |
| Practical 7 | Fetch the API data and render the list on React app. |
| Code | import React, { useEffect, useState } from 'react';  // import './App.css';  function FetchData() {  const [records, setRecords] = useState([]);  useEffect(() => {  fetch('https://jsonplaceholder.typicode.com/users')  .then(response => response.json())  .then(data => setRecords(data))  .catch(err => console.log(err));  }, []);  return (  <div>  <ul>  {records.map((list, index) => (  <li key={index}>  {list.id} | {list.name}  </li>  ))}  </ul>  </div>  );  }  export default FetchData; |
| Output |  |

**Practical - 8**

|  |  |
| --- | --- |
| Practical 8 | In Angular create TODO list with different components. |
| Code | **Angulartodo.html**  <!DOCTYPE html>  <html lang="en" ng-app="todoApp">  <head>  <meta charset="UTF-8">  <title>AngularJS To-Do List</title>  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>  <script src="todoang.js"></script>  </head>  <body>  <div ng-controller="TodoController">  <h2>My To-Do List</h2>  <input type="text" ng-model="newTodoText" placeholder="Add a new task">  <button ng-click="addTodo()">Add Task</button>  <ul>  <li ng-repeat="todo in todos">  <input type="checkbox" ng-model="todo.done">  <span ng-class="{ 'done': todo.done }">{{ todo.text }}</span>  <button ng-click="removeTodo(todo)">Remove</button>  </li>  </ul>  <p>Remaining tasks: {{ remaining() }}</p>  </div>  <script defer src="https://static.cloudflareinsights.com/beacon.min.js/vcd15cbe77  72f49c399c6a5babf22c1241717689176015" integrity="sha512-ZpsOmlRQV6y907TI0dKBHq9Md29nnaEIPlkf84rnaERnq6zv  WvPUqr2ft8M1aS28oN72PdrCzSjY4U6VaAw1EQ==" data-cf-beacon='{"version":"2024.11.0","token":"499e684b7b1043878  977050a0a606794","r":1,"server\_timing":{"name"  :{"cfCacheStatus":true, "cfEdge": true,"cfExtPri": true,"cfL4":true,"cfOrigin":true,"  cfSpeedBrain":true},"location\_startswith":null}}' crossorigin="anonymous"></script>  </body>  </html>  **Todoang.js**  angular.module('todoApp', [])  .controller('TodoController', function($scope) {  $scope.todos = [  { text: 'Learn AngularJS', done: false },  { text: 'Build a To-Do list', done: true }  ];  $scope.addTodo = function() {  if ($scope.newTodoText) {  $scope.todos.push({ text: $scope.newTodoText, done: false });  $scope.newTodoText = ''; // Clear input  }  };  $scope.removeTodo = function(todoToRemove) {  $scope.todos = $scope.todos.filter(function(todo) {  return todo !== todoToRemove;  });  };  $scope.remaining = function() {  var count = 0;  angular.forEach($scope.todos, function(todo) {  count += todo.done ? 0 : 1;  });  return count;  };  }); |
| Output |  |

**Practical - 9**

|  |  |
| --- | --- |
| Practical 9 | Fetch the API data and render the list on the Angular app. |
| Code |  |
| Output |  |

s