**Day5(05-Sept) Reactjs**

**=============================================================**

**Session Agenda:**

RWD using bootstrap

Web API (RESTful Services)

Create REST using json-server

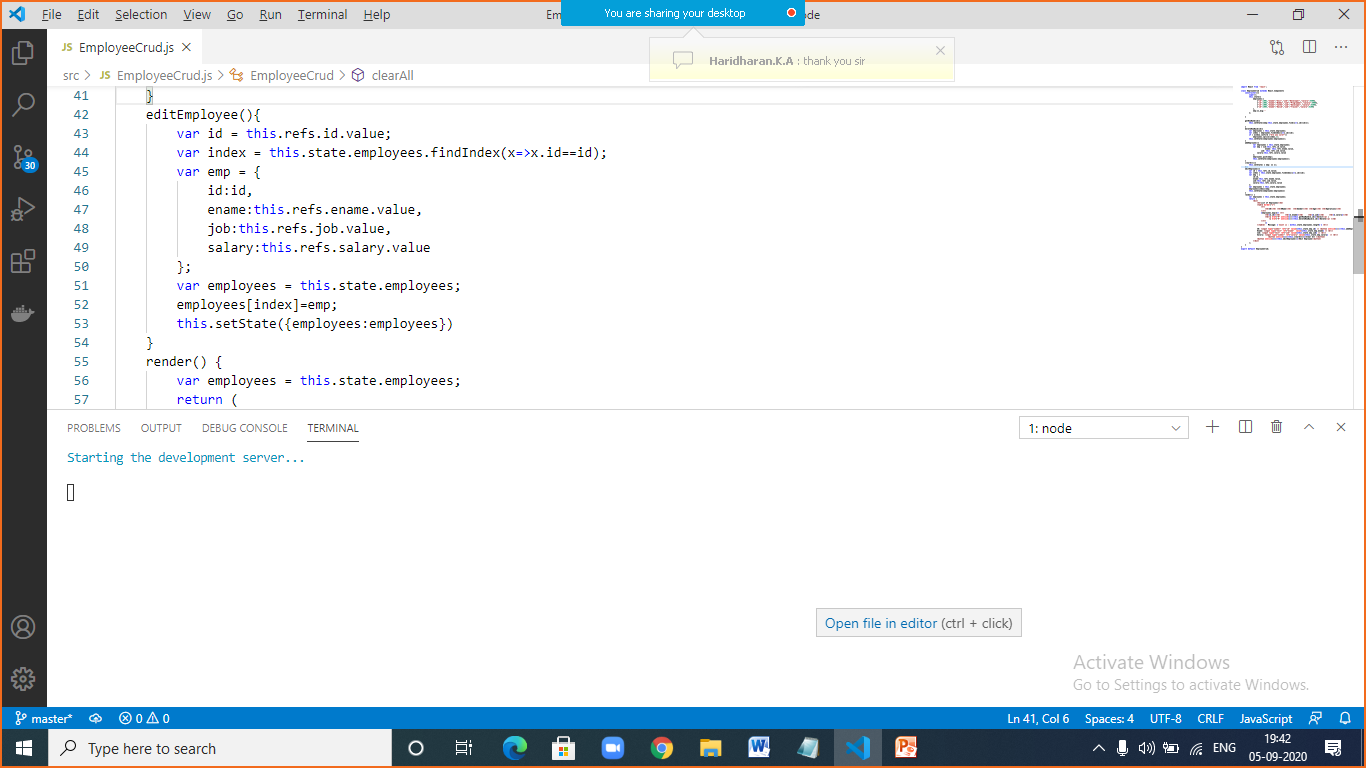
REST Client

Consume REST using React.js

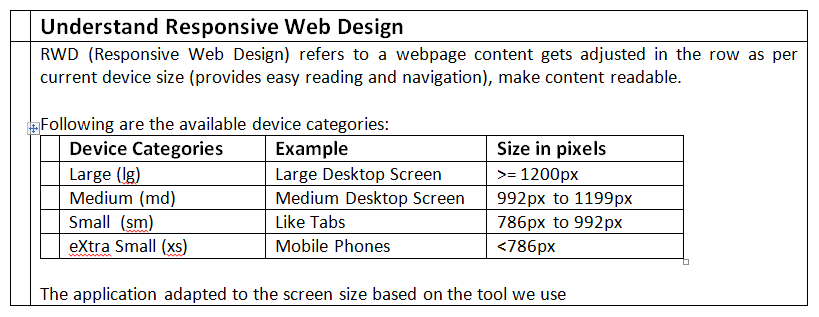
Using fetch() method and axios

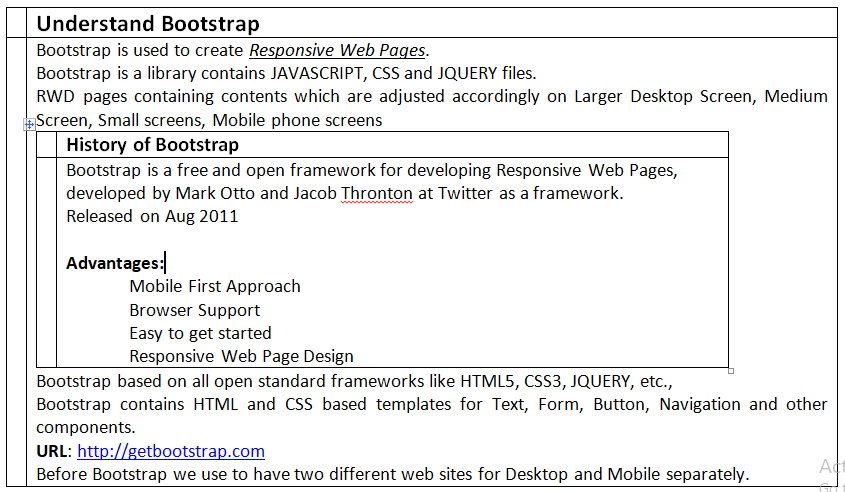
Routing

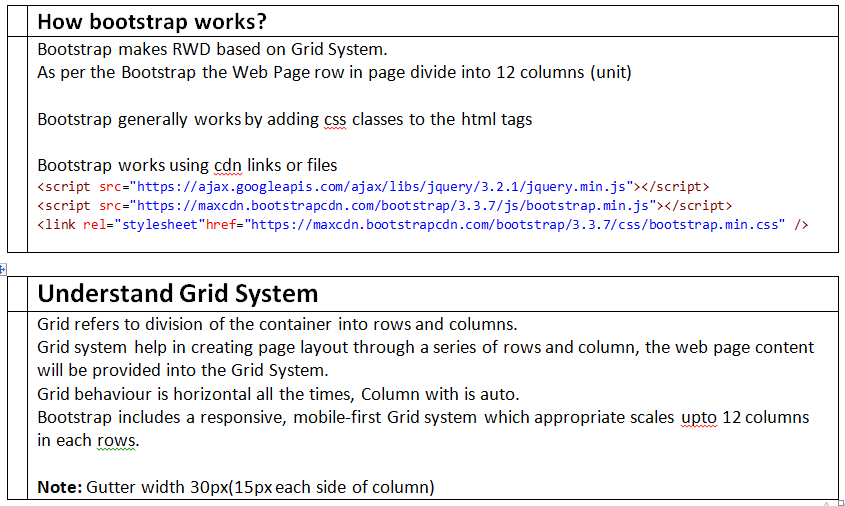
Edit employee source in json data:

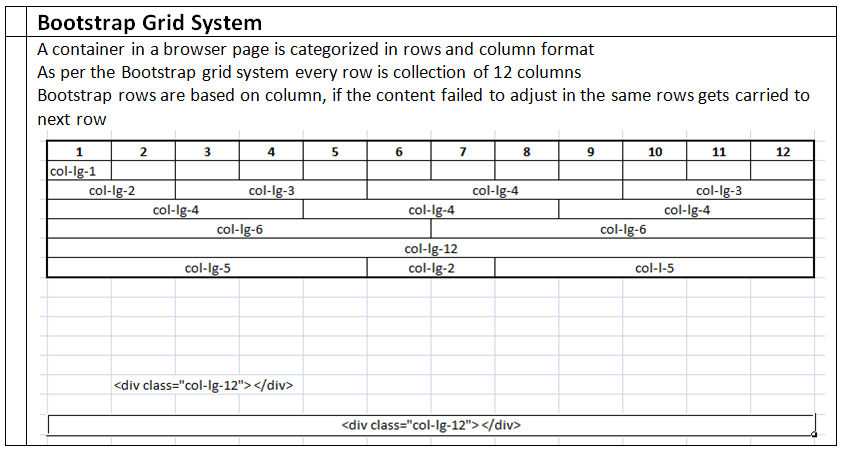


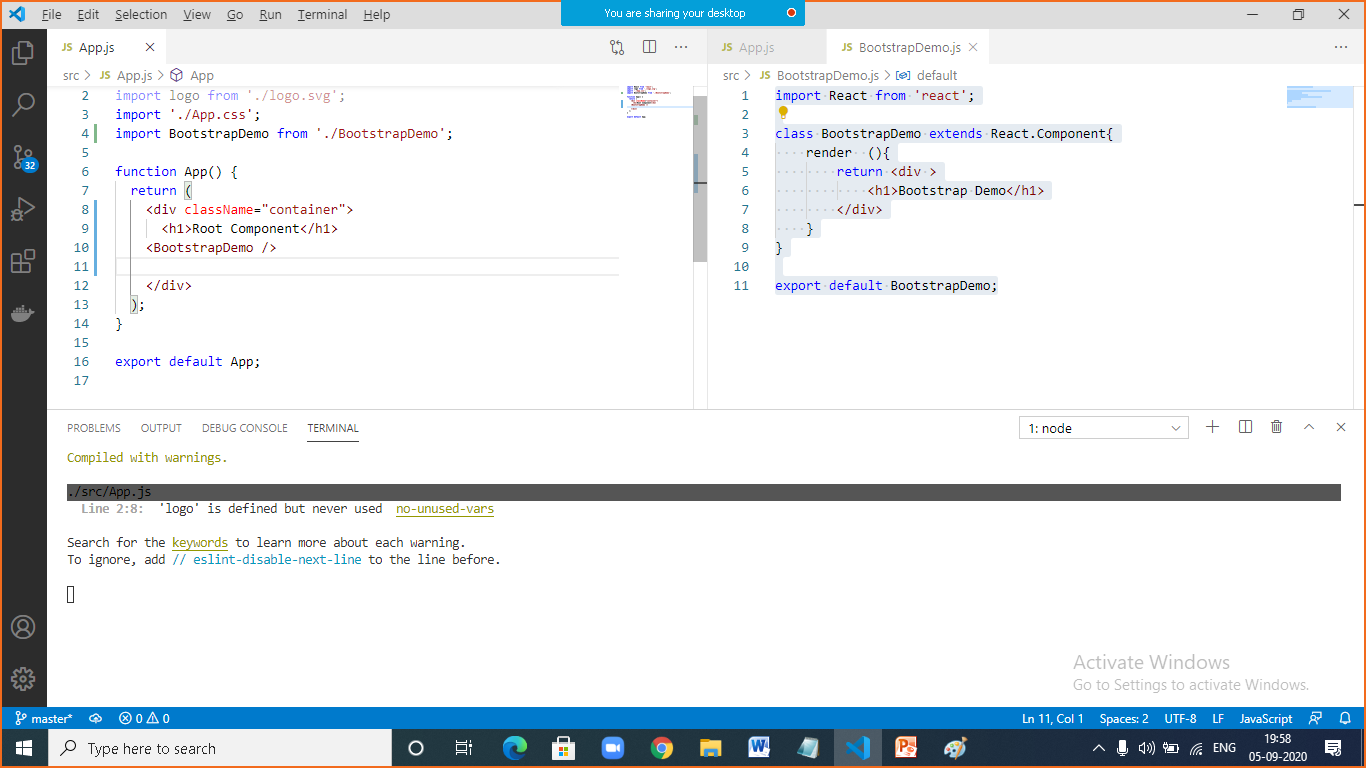
|  |  |
| --- | --- |
|  |  |
|  | import React from 'react';  class EmployeeCrud extends React.Component{  constructor(){  super();  this.state={  employees:[  {"id":1001,"ename":"Anil","job":"Developer","salary":5500},  {"id":1002,"ename":"Bobby","job":"Programmer","salary":5900},  {"id":1003,"ename":"Cathe","job":"Developer","salary":5300},  {"id":1004,"ename":"David","job":"Trainer","salary":5100}  ],  emp:{},msg:''  };    }    getRowById(id){  this.setState({emp:this.state.employees.find(x=>x.id==id)});    }  deleteRowByid(id){  var employees = this.state.employees;  var index = employees.findIndex(x=>x.id==id);  if ( window.confirm ("Are you sure?"))  employees.splice(index,1);  this.setState({employees:employees});  }  addEmployee(){  var employees = this.state.employees;  var emp = {id:this.refs.id.value,  ename: this.refs.ename.value,  job: this.refs.job.value,  salary:this.refs.salary.value  };  employees.push(emp);  this.setState({employees:employees});  }  clearAll(){  this.setState( { emp: {} });  }  editEmployee(){  var id = this.refs.id.value;  var index = this.state.employees.findIndex(x=>x.id==id);  var emp = {  id:id,  ename:this.refs.ename.value,  job:this.refs.job.value,  salary:this.refs.salary.value  };  var employees = this.state.employees;  employees[index]=emp;  this.setState({employees:employees})  }  render() {  var employees = this.state.employees;  return (  <div>  <h2>List of Employees</h2>  <table border="1">  <tr>  <th>ID</th> <th>EName</th> <th>Gender</th> <th>Age</th> <th>Operations</th>  </tr>  {employees.map(e=> <tr>  <td>{e.id}</td> <td>{e.ename}</td> <td>{e.job}</td> <td>{e.salary}</td>  <td><a href="#" onClick={()=>this.getRowById(e.id)}>Select</a> |  <a href="#" onClick={()=>this.deleteRowByid(e.id)}>Delete</a> </td>  </tr>  )}  </table> Message: {`Count is : ${this.state.employees.length}`} <hr/>    ID: <input type="number" ref="id" value={this.state.emp.id} /> <button onClick={()=>this.addEmployee()}>Add</button> <br/>  Ename: <input type="text" ref="ename" value={this.state.emp.ename} /> <br/>  Job: <input type="text" ref="job" value={this.state.emp.job} /> <br/>  Salary: <input type="number" ref="salary" value={this.state.emp.salary} /> <br/>  <button onClick={()=>this.clearAll()}>Clear All </button>  <button onClick={()=>this.editEmployee()}>Edit Employee</button>  </div>  );  }  }  export default EmployeeCrud; |

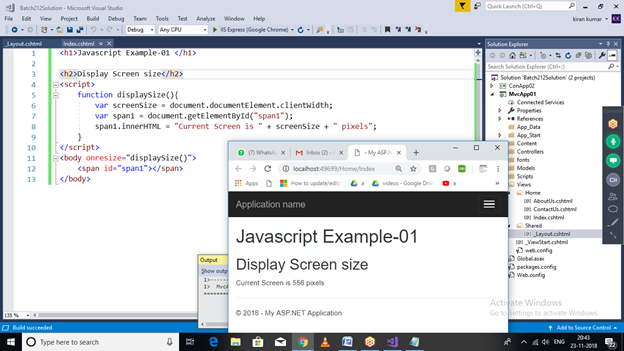








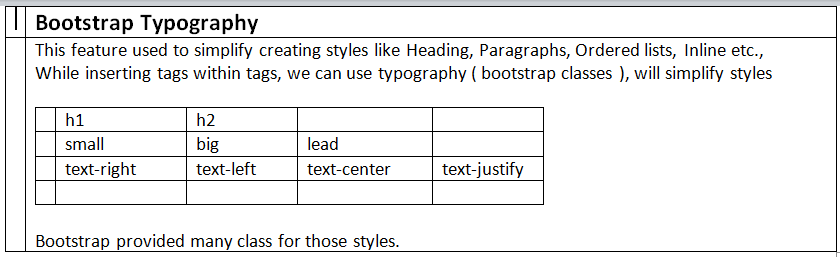


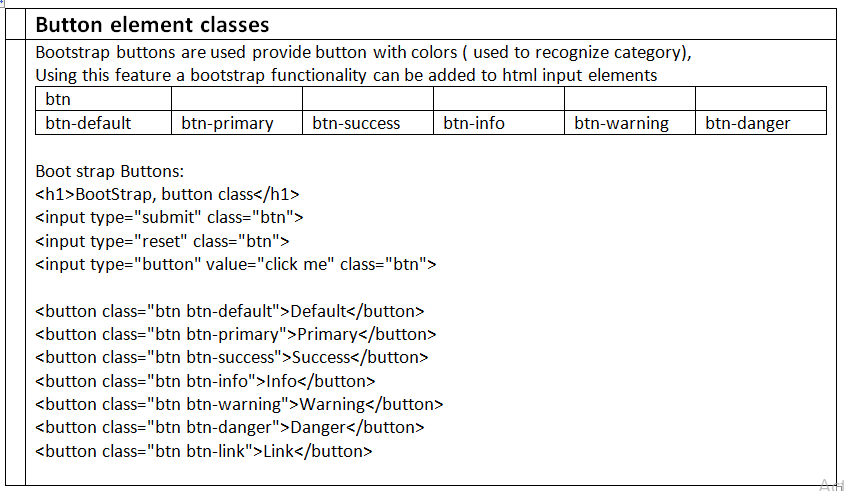


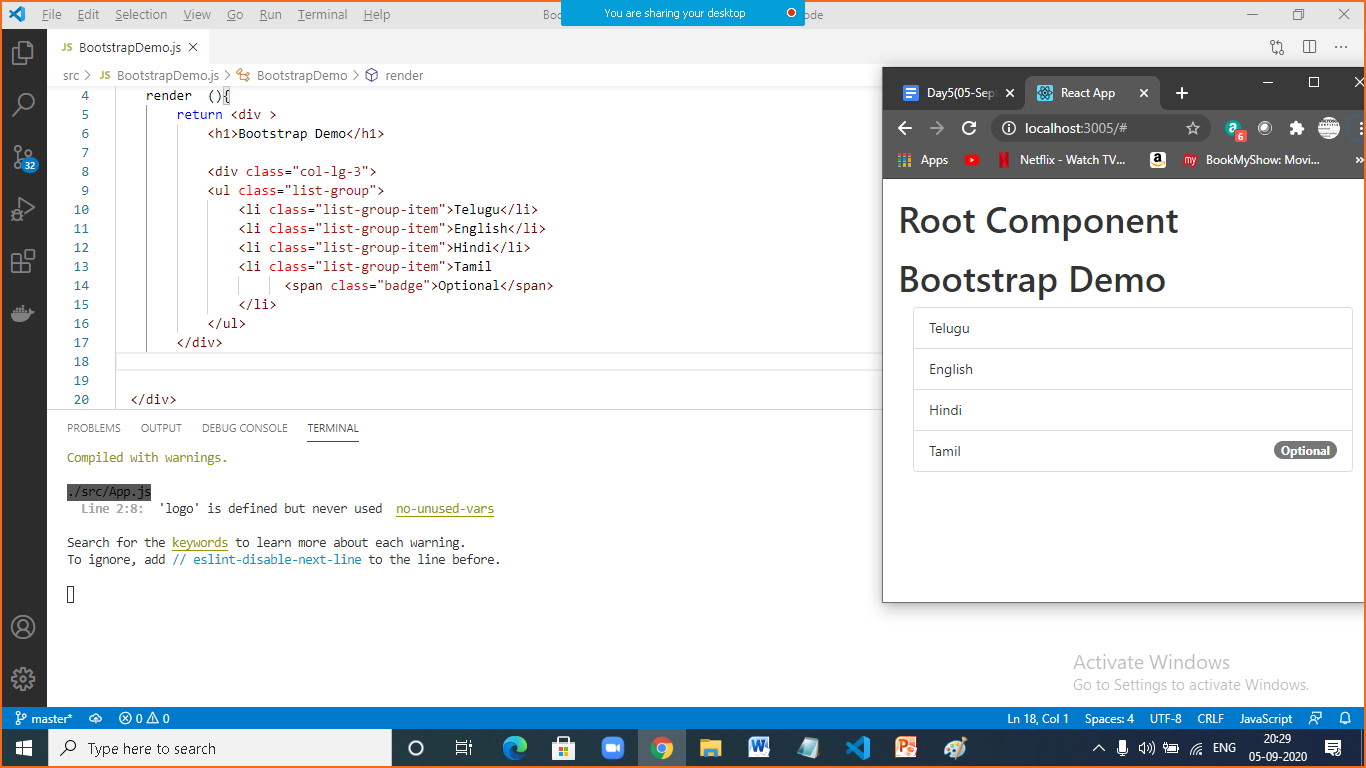
|  |  |
| --- | --- |
|  | **Display current screen size and screen type** |
|  | <script>  function displaySize(){  var screenSize = document.documentElement.clientWidth;  var span1 = document.getElementById("span1");  span1.innerHTML = "Current Screen is " + screenSize + " pixels<br/>";  if (screenSize >= 1200)  span1.innerHTML += "Current Screen Type is : Large Screen Display";  else if (screenSize >= 992)  span1.innerHTML += "Current Screen Type is : Medium Screen Display";  else if (screenSize >= 786)  span1.innerHTML += "Current Screen Type is : Small Screen Display";  else  span1.innerHTML += "Current Screen Type is : Extra Small Screen Display";  }  </script>  <body onresize="displaySize()">  <h2>Javascript code to display current screen size</h2>  <span id="span1"></span>  </body> |

|  |  |
| --- | --- |
|  | Display different no. of column in each category of screen size |
|  | import React from 'react';  import './App.css'  class BootstrapDemo extends React.Component{  render (){  return <div >  <h1>Bootstrap Demo</h1>  <div class="row">  <div class="visible-lg">  <h1>Currently Large Screen, You can see 4 columns per row</h1>  </div>  <div class="visible-md">  <h1>Currently Medium Screen, You can see 3 columns per row</h1>  </div>  <div class="visible-sm">  <h1>Currently Small Screen, You can see 2 columns per row</h1>  </div>  <div class="visible-xs">  <h1>Currently Extra Small Screen, You can see 1 columns per row</h1></div>  </div>  <div class="row">  <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">  <div class="greenBorderClass"> <h3>Div1: First Column </h3> </div>  </div>  <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">  <div class="greenBorderClass"> <h3>Div2:Second Column </h3> </div>  </div>  <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">  <div class="greenBorderClass"> <h3>Div3: Third Column </h3> </div>  </div>  <div class="col-lg-3 col-md-4 col-sm-6 col-xs-12">  <div class="greenBorderClass"> <h3>Div4:Fourth Column </h3> </div>  </div>  </div>    </div>  }  }  export default BootstrapDemo; |

|  |  |
| --- | --- |
|  | **Visible and hidden classes** |
|  | <h1>BootStrap Grid System, Demo-1</h1>    Four Screen Sizes:  <div class="container">  <div class="row">  <div class="visible-lg" style="color:GREEN">Large Screen Size</div>  <div class="hidden-lg" style="color:RED"> Large Screen Size</div>    <div class="visible-md" style="color:GREEN">Medium Screen Size</div>  <div class="hidden-md" style="color:RED"> Medium Screen Size</div>    <div class="visible-sm "style="color:GREEN">Small Screen Size</div>  <div class="hidden-sm" style="color:RED"> Small Screen Size</div>    <div class="visible-xs" style="color:GREEN">Extra Small Screen Size</div>  <div class="hidden-xs" style="color:RED"> Extra Screen Size</div>    </div>  </div> |

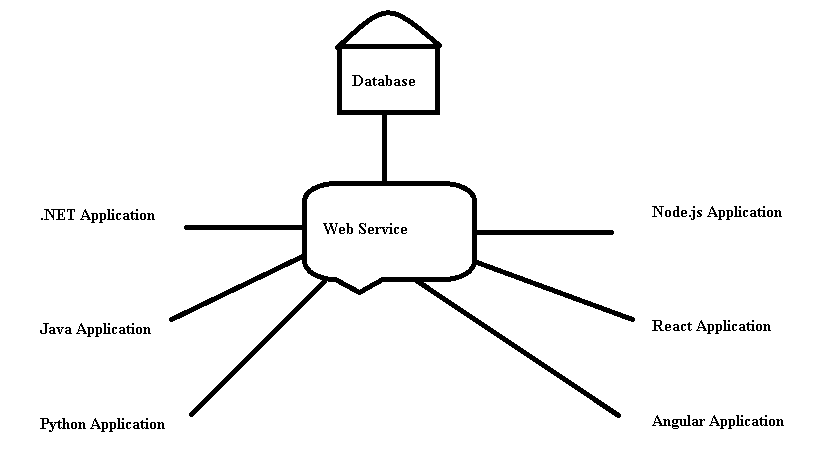


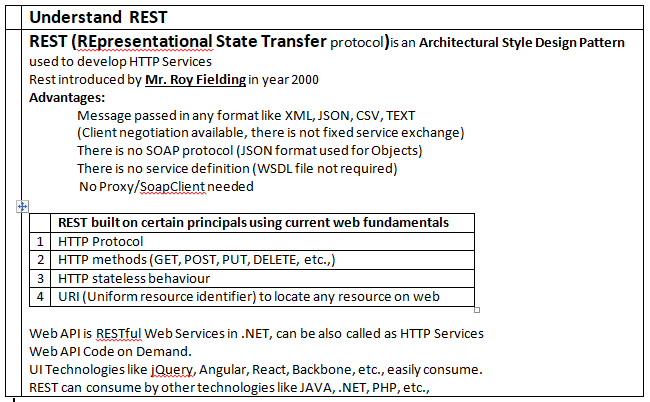


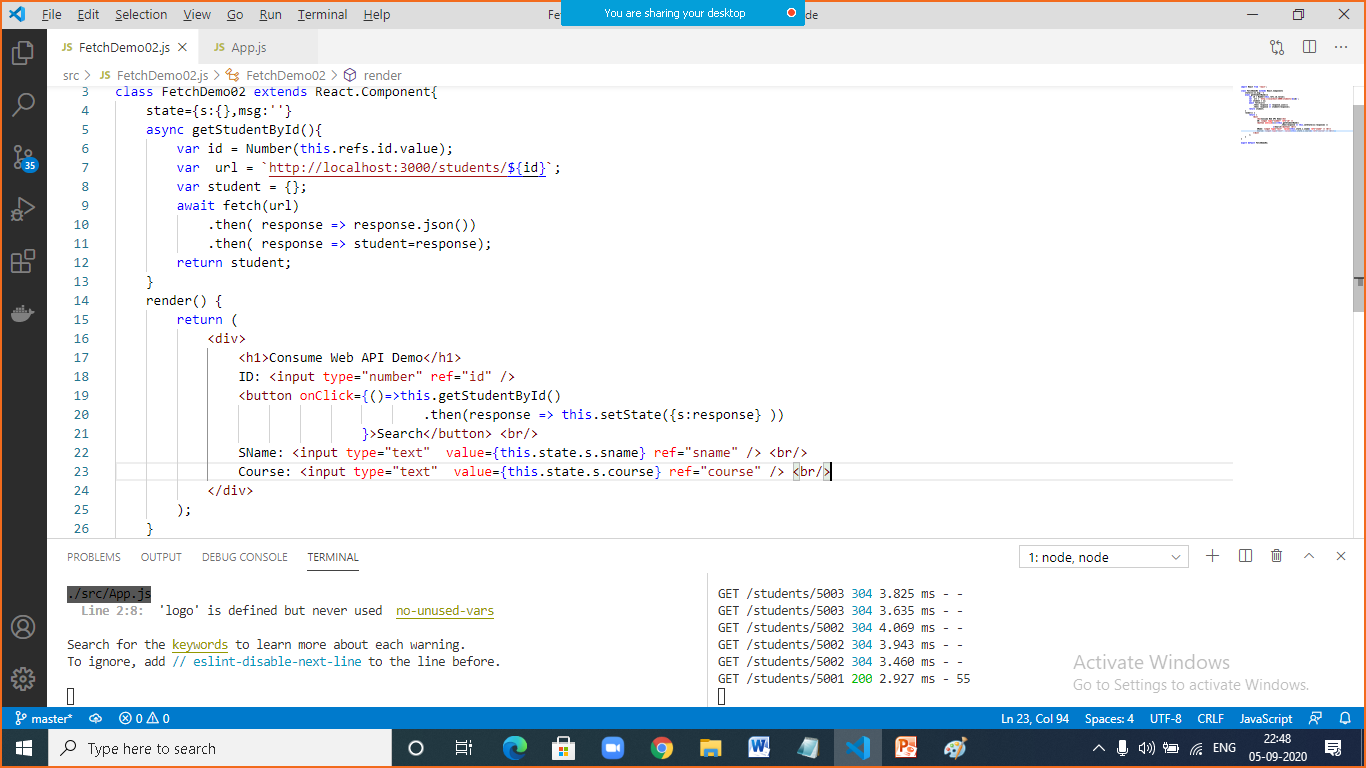


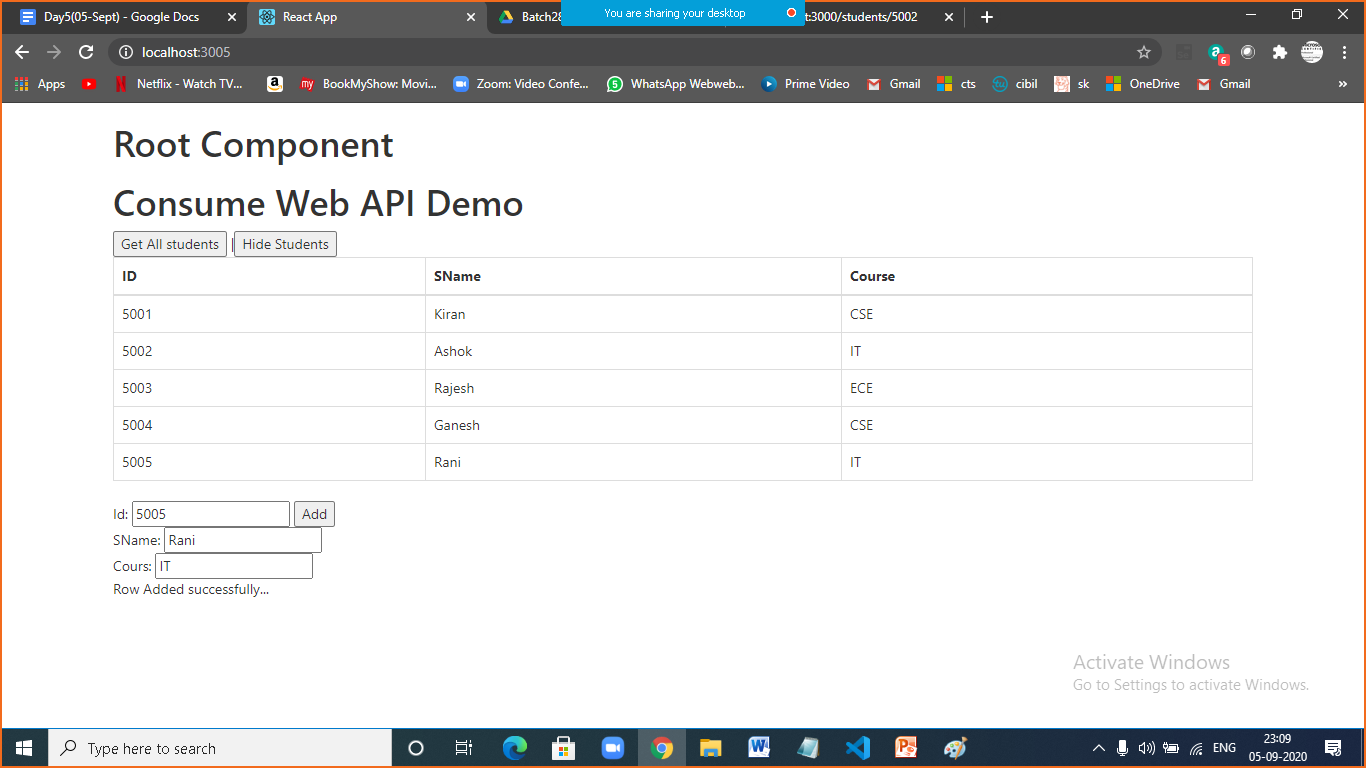
|  |  |
| --- | --- |
|  |  |
|  | import React from 'react';  import './App.css'  class BootstrapDemo extends React.Component{  render (){  return <div >  <h1>Bootstrap Dynamic Tabs Demo</h1>  <div class="row">  <ul class="nav nav-tabs">  <li><a href="#index" data-toggle="tab">Index</a></li>  <li><a href="#about" data-toggle="tab">About</a></li>  <li><a href="#contact" data-toggle="tab">Contact</a></li>  <li><a href="#login" data-toggle="tab">login</a></li>  </ul>    <div class="tab-content">  <div id="index" class="tab-pane fade">  <h1>User Login </h1>  </div>  <div id="about" class="tab-pane fade">  <h1>About</h1>  </div>  <div id="contact" class="tab-pane fade">  <h1>Contact</h1>  </div>  <div id="login" class="tab-pane fade">  <h1>Login </h1>  </div>  </div>  </div>  </div>  }  }  export default BootstrapDemo; |

|  |  |
| --- | --- |
|  | **Understand web service** |
|  | It is an inter-operable service, which can be consumed by any application developed using any technology.  Two types of web services:  SOAP Services  These are available using SOAP protocol  RESTful Services  These are http services  How to develop web service?  Technologies like .NET, Node.js, java, python, etc., used to develop web services  UI application cannot develop web services  UI application can consume Web Services |

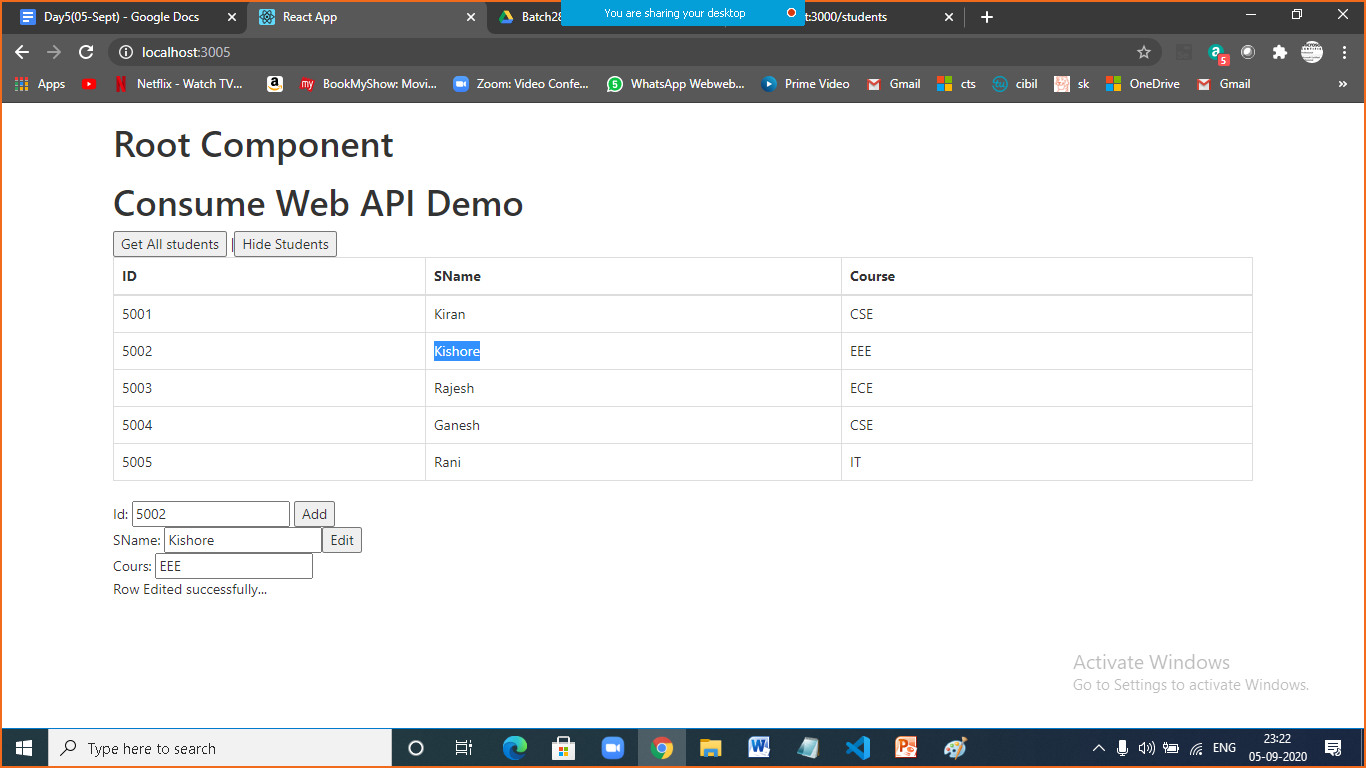




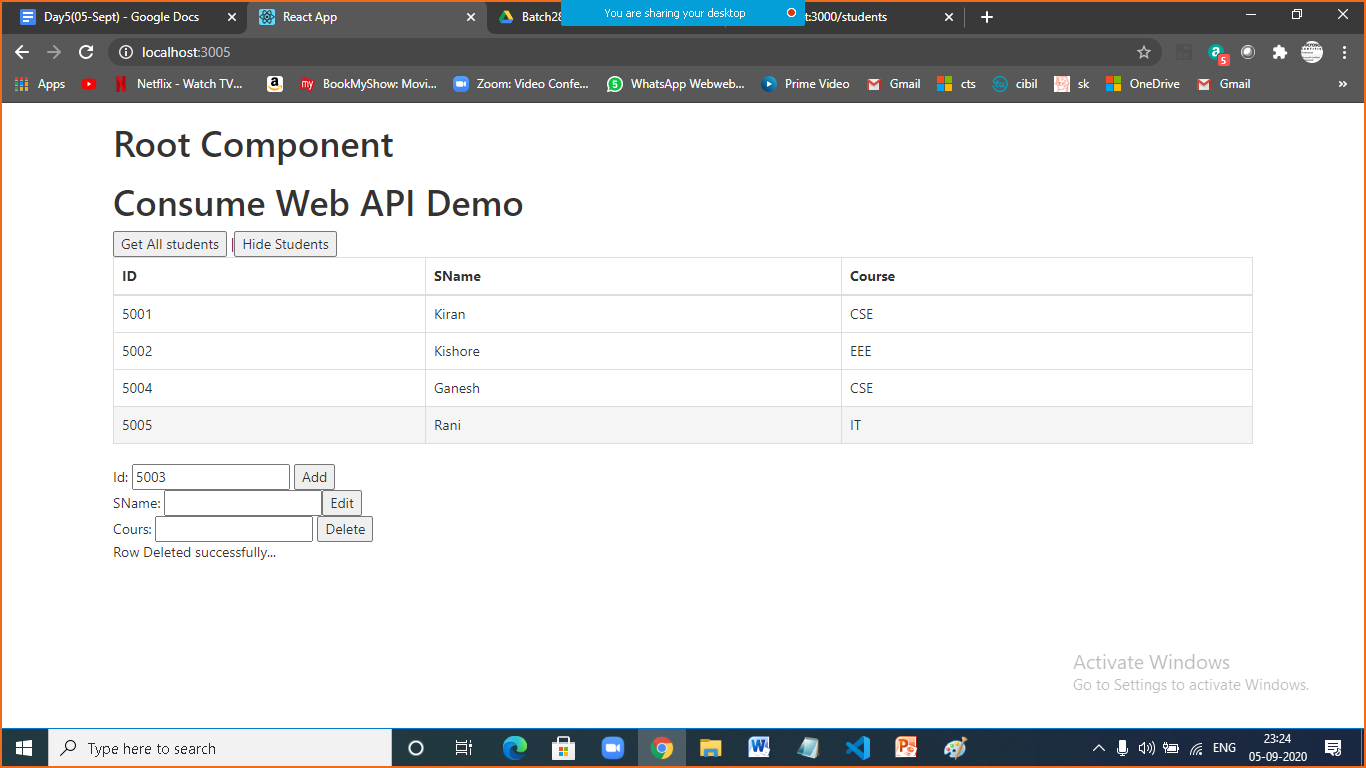




|  |  |
| --- | --- |
|  |  |
|  | import React from 'react';  class FetchDemo03 extends React.Component{  state={  students:[],  url: `http://localhost:3000/students`,  msg:''  }  async getStudents(){  var students = {};  await fetch(this.state.url)  .then( response => response.json())  .then( response => students=response);  return students;  }  hideStudents(){  this.setState({students:[]})  }  addStudent(){  var s = {  "id":Number(this.refs.id.value),  "sname":this.refs.sname.value,  "course":this.refs.course.value  };  fetch(this.state.url,{method:'POST',body:JSON.stringify(s), headers:{"content-type":"application/json"}})  .then( () => this.setState({msg:'Row Added successfully...'}))  .catch( error => this.setState({msg:error}));  }  render() {  return (  <div>  <h1>Consume Web API Demo</h1>  <button onClick={()=>this.getStudents()  .then(response => this.setState({students:response} ))  }>Get All students</button> |  <button onClick={()=>this.hideStudents()}>Hide Students</button> <br/>  <table class="table table-bordered table-hover">  <thead>  <tr> <th>ID</th> <th>SName</th> <th>Course </th> </tr>  </thead>  <tbody>  { this.state.students.map( s => <tr> <td>{s.id} </td> <td> {s.sname}</td> <td>{s.course}</td> </tr> )}  </tbody>  </table>  Id: <input type="number" ref="id" /> <button onClick={()=>this.addStudent()}>Add </button> <br/>  SName: <input type="text" ref="sname" /> <br/>  Cours: <input type="text" ref="course" /> <br/> {this.state.msg}  </div>  );  }  }  export default FetchDemo03; |

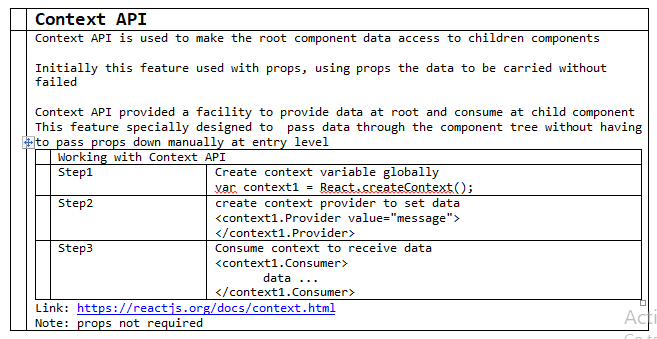


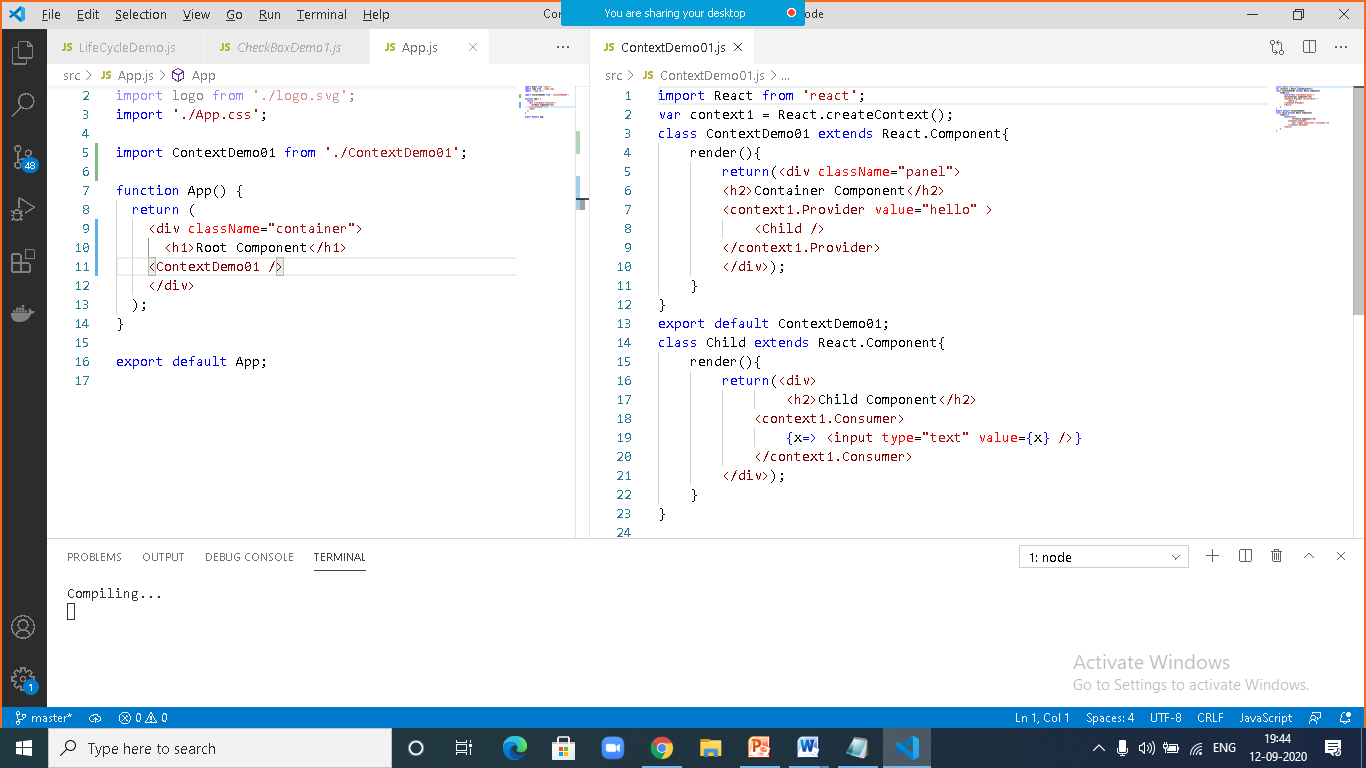
|  |  |
| --- | --- |
|  |  |
|  | import React from 'react';  class FetchDemo03 extends React.Component{  state={  students:[],  url: `http://localhost:3000/students`,  msg:''  }  async getStudents(){  var students = {};  await fetch(this.state.url)  .then( response => response.json())  .then( response => students=response);  return students;  }  hideStudents(){  this.setState({students:[]})  }  addStudent(){  var s = {  "id":Number(this.refs.id.value),  "sname":this.refs.sname.value,  "course":this.refs.course.value  };  fetch(this.state.url,{method:'POST',body:JSON.stringify(s), headers:{"content-type":"application/json"}})  .then( () => this.setState({msg:'Row Added successfully...'}))  .catch( error => this.setState({msg:error}));  }  editStudent(){  var s = {  "id":Number(this.refs.id.value),  "sname":this.refs.sname.value,  "course":this.refs.course.value  };  var url = `${this.state.url}/${s.id}`;  fetch(url,{method:'PUT',body:JSON.stringify(s), headers:{"content-type":"application/json"}})  .then( () => this.setState({msg:'Row Edited successfully...'}))  .catch( error => this.setState({msg:error}));  }    render() {  return (  <div>  <h1>Consume Web API Demo</h1>  <button onClick={()=>this.getStudents()  .then(response => this.setState({students:response} ))  }>Get All students</button> |  <button onClick={()=>this.hideStudents()}>Hide Students</button> <br/>  <table class="table table-bordered table-hover">  <thead>  <tr> <th>ID</th> <th>SName</th> <th>Course </th> </tr>  </thead>  <tbody>  { this.state.students.map( s => <tr> <td>{s.id} </td> <td> {s.sname}</td> <td>{s.course}</td> </tr> )}  </tbody>  </table>  Id: <input type="number" ref="id" /> <button onClick={()=>this.addStudent()}>Add </button> <br/>  SName: <input type="text" ref="sname" /><button onClick={()=>this.editStudent()}>Edit </button> <br/>  Cours: <input type="text" ref="course" /> <br/> {this.state.msg}  </div>  );  }  }  export default FetchDemo03; |



|  |  |
| --- | --- |
|  |  |
|  | import React from 'react';  class FetchDemo03 extends React.Component{  state={  students:[],  url: `http://localhost:3000/students`,  msg:''  }  async getStudents(){  var students = {};  await fetch(this.state.url)  .then( response => response.json())  .then( response => students=response);  return students;  }  hideStudents(){  this.setState({students:[]})  }  addStudent(){  var s = {  "id":Number(this.refs.id.value),  "sname":this.refs.sname.value,  "course":this.refs.course.value  };  fetch(this.state.url,{method:'POST',body:JSON.stringify(s), headers:{"content-type":"application/json"}})  .then( () => this.setState({msg:'Row Added successfully...'}))  .catch( error => this.setState({msg:error}));  }  editStudent(){  var s = {  "id":Number(this.refs.id.value),  "sname":this.refs.sname.value,  "course":this.refs.course.value  };  var url = `${this.state.url}/${s.id}`;  fetch(url,{method:'PUT',body:JSON.stringify(s), headers:{"content-type":"application/json"}})  .then( () => this.setState({msg:'Row Edited successfully...'}))  .catch( error => this.setState({msg:error}));  }  deleteStudent(){  var id =Number( this.refs.id.value);  var url = `${this.state.url}/${id}`;  fetch(url,{method:'DELETE'})  .then( () => this.setState({msg:'Row Deleted successfully...'}))  .catch( error => this.setState({msg:error}));  }  render() {  return (  <div>  <h1>Consume Web API Demo</h1>  <button onClick={()=>this.getStudents()  .then(response => this.setState({students:response} ))  }>Get All students</button> |  <button onClick={()=>this.hideStudents()}>Hide Students</button> <br/>  <table class="table table-bordered table-hover">  <thead>  <tr> <th>ID</th> <th>SName</th> <th>Course </th> </tr>  </thead>  <tbody>  { this.state.students.map( s => <tr> <td>{s.id} </td> <td> {s.sname}</td> <td>{s.course}</td> </tr> )}  </tbody>  </table>  Id: <input type="number" ref="id" /> <button onClick={()=>this.addStudent()}>Add </button> <br/>  SName: <input type="text" ref="sname" /><button onClick={()=>this.editStudent()}>Edit </button> <br/>  Cours: <input type="text" ref="course" /> <button onClick={()=>this.deleteStudent()}>Delete </button><br/> {this.state.msg}  </div>  );  }  }  export default FetchDemo03; |

|  |  |
| --- | --- |
|  | Task |
|  | Validate credentials from users rouces in api at login view |





|  |  |
| --- | --- |
|  | **Context API example-2** |
|  | import React from 'react';  import ReactDOM from 'react-dom';  import './index.css';  import registerServiceWorker from './registerServiceWorker';  var context1 = React.createContext();  class App extends React.Component{  state={  t1:React.createRef(),  msg:''  }  changeMessage(){  this.setState({msg:this.state.t1.current.value});  }  render(){    return(<div className="panel">  <h2>Container Component</h2>  Enter msg: <input type="text" ref={this.state.t1} /> <br/>  <button onClick={()=>this.changeMessage()}> Update </button>  <context1.Provider value={this.state.msg} >  <Child1 />  <Child2 />  </context1.Provider>    </div>);    }  }  class Child1 extends React.Component{  render(){  return(<div>  <h2>Child1 Component</h2>    <context1.Consumer>  {x=> <input type="text" value={x} />}  </context1.Consumer>  </div>);  }  }  class Child2 extends React.Component{  render(){  return(<div>  <h2>Child2 Component</h2>  <context1.Consumer>  {x=> <input type="text" value={x} />}  </context1.Consumer>  </div>);  }  } |

|  |  |
| --- | --- |
|  | **Context API, Example-3** |
|  | import React from 'react';  import ReactDOM from 'react-dom';  import './index.css';    import registerServiceWorker from './registerServiceWorker';  var context1 = React.createContext();  class App extends React.Component{  constructor(){  super();  this.t1=React.createRef();  this.state={value:5}  }  render(){  return(<div className="container GreenBorder">  <h2>Root Component </h2>  Enter No: <input type="number" ref={this.t1} /> <br/>  <button onClick={()=>this.setState({value:this.t1.current.value})}>Proceed</button>  <context1.Provider value={this.state.value}>  <Child1 />  <Child2 />  </context1.Provider>  </div>  );  }  }  class Child1 extends React.Component{    render(){  return(<div className="BlueBorder">  <h2>Child1 Component</h2>  <h3>Square Calculator</h3>  <context1.Consumer>  {data => <span> {`Square of ${data} is ${data\*data}` } </span>}  </context1.Consumer>  </div>);  }  }    class Child2 extends React.Component{  render(){  return(<div className="BlueBorder">  <h2>Child2 Component</h2>  <h3>Cube Calculator</h3>  <context1.Consumer>  {data => <span> {`Cube of ${data} is ${data\*data\*data}` } </span>}  </context1.Consumer>  </div>);  }  }  ReactDOM.render(<App />,document.getElementById('root'));  registerServiceWorker(); |

