

Register No.

--	--	--	--	--	--	--	--

BTech Degree Examination December 2022

Fifth Semester

Information Technology

20ITE03 – USER INTERFACE DESIGN

(Regulations 2020)

Time: Three hours

Maximum: 100 marks

Answer all Questions

Part – A ($10 \times 2 = 20$ marks)

1. What is the role of Virtual DOM? [CO1,K1]
2. Specify the advantages of ReactJS. [CO1,K1]
3. Differentiate between class component and function component. [CO2,K2]
4. Mention the purpose of fragment in ReactJS. [CO2,K1]
5. How do you perform type checking for props? [CO3,K2]
6. Write the syntax to Create state object in class component. [CO3,K2]
7. Specify the purpose of bind() method in ReactJS. [CO4,K2]
8. Give a simple example for event handling in ReactJS. [CO4,K2]
9. Draw the Redux Architecture. [CO5,K1]
10. List any four types of Hooks supported by ReactJS. [CO5,K1]

Part – B ($5 \times 16 = 80$ marks)

11. a. i) Write a Javascript program to manage the product details like product id, product name, manufacturer, quantity and price information using classes and objects. Initialize the product information using constructor and display the information in table format in the webpage. (10) [CO1,K3]
- ii) Elaborate the process involved in creating and rendering the elements in ReactJS with suitable example. (6) [CO1,K2]
- (OR)
- b. i) Create a webpage for E-bike manufacturing company using different types of CSS supported by ReactJS. (10) [CO1,K3]
- ii) Illustrate arrow function with suitable example. (6) [CO1,K2]
12. a. i) Outline the working of functional component in ReactJS. (10) [CO2,K1]
- ii) Explain in detail about component extraction with example. (6) [CO2,K2]

(OR)

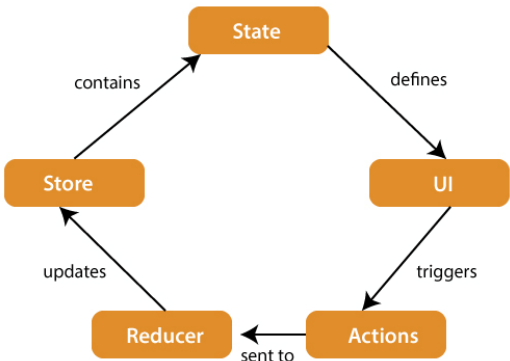
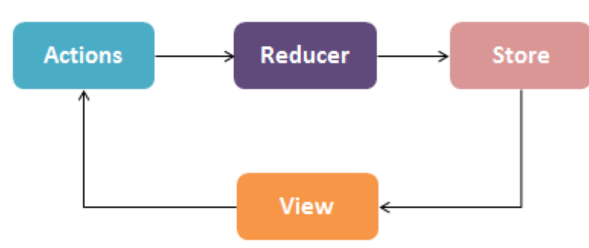
- b. i) Summarize the steps involved in creating a class component. (10) [CO2,K1]
 ii) How does component composition works in ReactJS? Given an example. (6) [CO2,K2]
13. a. Illustrate the React component lifecycle methods with suitable example. (16) [CO3,K2]
 (OR)
 b. Outline the usage of props and state in React application with suitable example. (16) [CO3,K2]
14. a. i) Design a controlled component based form to collect the book details like (10) [CO4,K3]
 book title, author name, year of publication, price, ISBN number and publisher information and display the book information.
 ii) Demonstrate conditional rendering in react with necessary example. (6) [CO4,K2]
 (OR)
 b. i) Develop a React application using uncontrolled component to receive (10) [CO4,K3]
 Employee information like 1D, Name, Designation, Department, Branch, Mobile No and E-mail and display the Employee information on the webpage.
 ii) Outline the use of list and keys with example. (6) [CO4,K2]
15. a. i) Create a website for a reputed multi speciality hospital in Coimbatore with (10) [CO5,K3]
 the components like Home, About, Departments, Facilities, Emergency and contract. Enable navigation between the components using react-router library.
 ii) Consider a Grandparent component that needs to pass properties to its (6) [CO5,K3]
 descendent components without props drilling. Writ a react program using hook.
 (OR)
 b. Develop a React application using Redux to manage the available seats in the (16) [CO5,K3]
 restaurant based on the user check-in and check-out process. Consider the number of available seats in the library is 50.

Bloom's Taxonomy Level	Remembering (K1)	Understanding (K2)	Applying (K3)	Analysing (K4)	Evaluating (K5)	Creating (K6)
Percentage	17	43	40	-	-	-

B.Tech Degree Examinations Nov / Dec 2022
Fifth Semester
Information Technology
20ITE03 – USER INTERFACE DESIGN
Answer Key
(Regulation 2020)

Part A (10 x 2 = 20 Marks)

1.	The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as ReactDOM → (2)	
2.	Any four advantages, → (2) <ul style="list-style-type: none"> • Reusability • Simple and easy to use • Fast, efficient • Composable • JSX • Virtual DOM • One-way binding • Component based 	
3.	Consider any two points → (2)	
	<p>Class Component:</p> <ul style="list-style-type: none"> • More complex than functional component • It must extend React.Component class and implement render method • supports life cycle methods by default • Also called as stateful component • Eg: <pre>class MyComponent extends React.Component { render() { return (<div>This is main component.</div>); } }</pre>	<p>Functional Component:</p> <ul style="list-style-type: none"> • Simpler than class component • Should return html or jsx syntax • Utilizes hooks to impart life cycle methods • Also called as stateless component • Eg: <pre>function WelcomeMessage() { return <h1>Welcome to the , {props.name}</h1>; }</pre>
4.	<p>React fragments serve as a cleaner alternative to using unnecessary divs in our code. Fragments do not produce any extra elements in the DOM. Fragment's child components will render without any wrapping DOM node.</p> <p>→ (2)</p> <p>Syntax:</p> <pre><React.Fragment> </React.Fragment></pre> <p>OR</p> <pre><> any html or JSX syntax </></pre>	
5.	<p>→ (2)</p> <p>React uses propTypes property for props typechecking</p> <p>Eg: import PropTypes from 'prop-types';</p> <pre>class Greeting extends React.Component { render() {</pre>	

	<pre> return (<h1>Hello, {this.props.name}</h1>); } } </pre> <p>Greeting.propTypes = { name: PropTypes.string };</p>
6.	<p>→(2)</p> <pre> class className extends React.Component{ constructor(props){ this.state={property1: value1, property2: value2} } } </pre>
7.	<p>→(2)</p> <p>The bind() method allows us to easily set which object will be bound by the this keyword when a function or method is invoked.</p> <p>It is also used to pass the data as an argument to the function of a class based component.</p>
8.	<p>→(2)</p> <p>Simple example of event handling in React, Program using functional and class components are acceptable. Following is an example of onChange event, similar events and its corresponding event handlers are acceptable.</p> <pre> const Demo=()=>{ const [data, setData] = useState("") const handleChange=(e)=>{setData(e.target.value)} return(<> <input type='text' value={data} onChange={handleChange}/> </>) </pre>
9.	<p>The architecture consists of UI(react components), Actions, Reducer and Redux Store. Architecture similar to the following are acceptable</p> <div style="display: flex; align-items: center; justify-content: space-around;">  <div style="text-align: center;"> <p>(OR)</p>  <p>Redux Architecture</p> </div> </div> <p>→(2)</p>

10.	Any 4 hooks, → (2)	
	<ul style="list-style-type: none"> • useState • useEffect • useRef • useMemo • useContext • useCallback • custom hook 	
PART – B (5 x 16 = 80 Marks)		
11.	<p>a) JavaScript Classes are templates for JavaScript Objects. Use the keyword class to create a class. Always add a method named constructor(). The constructor method is called automatically when a new object is created.</p> <p>i) Program to display single or multiple product information may be considered</p> <p>Example:</p> <pre> <!DOCTYPE html> <html> <body> <h2>Product Details</h2> <p id="demo"></p> <script> var products=[] class Product { constructor(pid, name, manufacturer, quantity, price) { this.pid = pid; this.name = name; this.manufacturer = manufacturer; this.quantity = quantity; this.price = price; } } const myProduct1 = new Product(101, "1Litre bottle", "Milton", 10, 500); const myProduct2 = new Product(102, "1Litre bottle", "Cello", 50, 300); products=[myProduct1, myProduct2] var html = "<table border='1 1'>"; for (vari = 0; i<products.length; i++) { html+="<tr>"; html+="<td>" + products[i].pid + "</td>"; html+="<td>" + products[i].name + "</td>"; html+="<td>" + products[i].manufacturer + "</td>"; html+="<td>" + products[i].quantity + "</td>"; html+="<td>" + products[i].price + "</td>"; html+="</tr>"; } html+="</table>"; document.getElementById("demo").innerHTML =html; </script> </body> </html> </pre>	(10)

11.	a) ii)	<p>React elements are plain objects, and are easy to create. React DOM takes care of updating the DOM to match the React elements.</p> <ul style="list-style-type: none"> To Create a element: const element = <h1>Hello, world</h1>; Rendering the element: const root = ReactDOM.createRoot(document.getElementById('root')); root.render(element); 	(2) (4)
11.	b) i)	<p>Website for E-Bike:</p> <p>React Supports the following CSS methods:</p> <p>Inline styling CSS stylesheets CSS Modules</p> <p>Program using three types of CSS similar to following is acceptable.</p> <p>Inline styling:</p> <p>To style an element with the inline style attribute, the value must be a JavaScript object: const root = ReactDOM.createRoot(document.getElementById('root')); const Header = () => {</p> <pre>//Using JS object const myStyle = { color: "white", backgroundColor: "DodgerBlue",padding: "10px",fontFamily: "Sans-Serif" }; return (< <h1 style={ myStyle}> E-Bike Manufacturing Company</h1> <h2 style={ { color: "red" } }>ERODE</h2></>); }</pre> <p>root.render(<Header/>);</p> <p><u>CSS Stylesheet:</u></p> <p>write your CSS styling in a separate file, just save the file with the .css file extension import it in your application. import './index.css';</p> <p><u>Index.css</u></p> <pre>body { background-color: #282c34; color: white; padding: 40px; font-family: Sans-Serif; text-align: center; }</pre> <p>import './Index.css';</p> <pre>const Header = () => { return (< <h1>RS 100</h1> <p>100kms per charge.</p> </>) }</pre>	(1) (3) (3)

		<pre>); }</pre> <p>const root = ReactDOM.createRoot(document.getElementById('root')); root.render(<Header />);</p> <p><u>CSS Modules:</u></p> <p>CSS Modules are convenient for components that are placed in separate files. <u>my-style.module.css</u></p> <pre>.bigblue { color: rgb(134, 3, 58); padding: 40px; font-size: 24px; font-family: Sans-Serif; text-align: center; }</pre> <p>import styles from './my-style.module.css'; <h1 className={styles.bigblue}> E-Bike Manufacturing Company </h1></p>	(3)
11.	b) ii)	<p>Arrow function:</p> <p>Example using React js may be considered. functionDeclarationfunctionname = (parameterList)=> {function body}</p> <p>Example:</p> <pre>const Display=()=>{return(<>Hello World</>)}</pre>	(6)
12.	a) i)	<p><u>Functional components:</u></p> <ul style="list-style-type: none"> Functional components are easy to create compared to class component. Functional component looks similar to javascript functions the difference is that functional component returns a HTML or JSX syntax and first character of the functional component name must be uppercase. Functional component can be created using either regular java script functions or arrow functions. <p>Example using regular function:</p> <pre>function Car(name){ return(<h1>The Car {name} looks cool</h1>) }</pre> <p>Example using Arrow function: const Car=()=>{return(<h1>The Car {name} looks cool</h1>)}</p>	(3) (7)
12.	a) ii)	<p><u>Component extraction:</u></p> <ul style="list-style-type: none"> Component extraction is the process of removing the complexity of a component and improve reusability. if a part of your UI is used several times (Button, Panel, Avatar), or is complex enough on its own (App, FeedStory, Comment), it is a good to extract to a separate component. Example similar to the following is acceptable: function Comment(props) { return (<div className="Comment"> //call to Avatar component <Avatar/> <div className="UserInfo-name"> {props.author.name} </div> 	(6)

		<pre> <div> <UI /> <button>Register</button> </div>); }} export {Register}; index.js import Demo_extract from './Demo_extract'; const root = ReactDOM.createRoot(document.getElementById('root')); root.render(<Register />); </pre>	(2)
13.	a)	<p>Life cycle methods are,</p> <ul style="list-style-type: none"> • Initial phase • Mounting phase • Updating phase • Unmounting phase • componentDidMount() • componentDidUpdate() • componentWillUnmount() <p>Program similar to following is acceptable.</p> <pre> import React, { Component } from 'react'; class App extends React.Component { constructor(props) { super(props); this.state = {hello: "JavaTpoint"}; this.changeState = this.changeState.bind(this) } render() { return (<div> <h1>ReactJS component's Lifecycle</h1> <h3>Hello {this.state.hello}</h3> <button onClick = {this.changeState}>Click Here!</button> </div>); } changeState(){ this.setState({hello:"All!!- Its a great reactjs tutorial."}); } componentDidMount() { console.log('Component Did MOUNT!') } componentDidUpdate(prevProps, prevState) { console.log('Component Did UPDATE!') } componentWillUnmount() { console.log('Component Will UNMOUNT!') } } export default App; </pre>	<p>(5)</p> <p>(5)</p> <p>(2)</p> <p>(2)</p> <p>(2)</p>

13.	b)	<p>Props:</p> <ul style="list-style-type: none"> • Props stand for "Properties." They are read-only components • Props are arguments passed into React components • It is an object which stores the value of attributes of a tag and work similar to the HTML attributes <p>States:</p> <ul style="list-style-type: none"> • State data can be modified by its own component, but is private. • State is managed within the component. <p>(Any program similar to the following may be considered) Creating/ Accessing: State Object – 5 Mark, Props – 5 Mark & Rendering 2 Mark)</p> <p><u>Counter.js</u></p> <pre>import React, { Component } from 'react'; export default class Counter extends Component{ constructor(props) { super(props) //state object this.state={count:0} } handle=()=>{this.setState({count:this.state.count+1})} render() { return(<div className='mt-4'> <h1>{props.name}</h1> <button onClick={this.handle}> click me</button> {''} <p>{this.state.count}</p></center></div> </>) } }</pre> <p><u>index.js</u></p> <pre>import Counter from './Counter'; const name="counter example"; const root = ReactDOM.createRoot(document.getElementById('root')); root.render(<Counter name={name}/>)</pre>	<p>(2)</p> <p>(2)</p>
14	a) i)	<p>Controlled Component</p> <p>Program similar to the following are acceptable:</p> <pre>import React, { useState } from 'react' export default function Form() { const [title, setTitle] = useState(""); const [author, setAuthor] = useState(""); const [yearofpub, setYearofpub] = useState(); const [price,setPrice] = useState(); const [isbn, setIsbn]=useState(""); const [publisher, setPublisher] = useState(""); const display= (e) => { document.getElementById("show").innerHTML =<h3>Title: {title}
Author: {author}
Year of Publication: {yearofpub}
 Price: {price}
 ISBN: {isbn}</pre>	<p>(10)</p>

```

<br/> Publisher: {publisher}</h3>;
e.preventDefault()
}
return (
<p id='show'></p>
<div>
<form>
  Title:<input type='text' onChange={ (e) => setTitle (e.target.value)} value={ title } />
<br /> Author:<input type='text' onChange={ (e) => setAuthor (e.target.value)} value={
author } />
<br /> Year of Publication:<input type='text' onChange={ (e) => setYearofpub
(e.target.value)} value={ yearofpub } />
  Price:<input type='text' onChange={ (e) => setPrice (e.target.value)} value={ price } />
  ISBN:<input type='text' onChange={ (e) => setIsbn (e.target.value)} value={ isbn } />
  Publisher:<input type='text' onChange={ (e) => setPublisher (e.target.value)} value={
publisher } />

<br /><button onClick={ () => display()}> Display Info</button>
</form>
</div> )}

```

(OR)

Program using Class Component may be considered:

```
import React from "react";
```

```

class FormApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = { author: "", title: "", year: "", price: "", publisher: "", isbn: "" };

    this.handleSubmit = this.handleSubmit.bind(this);
  }

```

```

  handleSubmit(event) {
    const msg = "You have submitted the input successfully: " +
      "<br/>Author Name:" + this.state.author +
      "<br/>Title:" + this.state.title +
      "<br/>Year:" + this.state.year +
      "<br/>Publisher:" + this.state.publisher +
      "<br/>isbn:" + this.state.isbn +
      "<br/>price:" + this.state.price;
    document.getElementById("msg").innerHTML = msg;
    event.preventDefault();
  }
  render() {
    return (
      <form onSubmit={ this.handleSubmit }>
      <h1>Controlled Form </h1>
      <label>
        Author:
        <input name="author" type="text" value={ this.state.author } />
      </label>

```

		<pre> <label> Author: <input name="title" type="text" value={this.state.title} /> </label> <label> Author: <input name="year" type="text" value={this.state.year} /> </label> <label> Author: <input name="price" type="text" value={this.state.price} /> </label> <label> Author: <input name="isbn" type="text" value={this.state.isbn} /> </label> <label> Author: <input name="publisher" type="text" value={this.state.publisher} /> </label> <input type="submit" value="Submit" /> </form>); } } export default function App() { return (<div className="App"> <FormApp/> <div id="msg"></div> </div>); } </pre>	
14	a) ii)	<p>Conditional Rendering in react</p> <p>One method sufficient to award mark:</p> <p>Logical && Operator:</p> <pre> render() { const count = 0; return (<div> {count && <h1>Messages: {count}</h1>} </div>); } </pre> <p>(OR)</p> <pre> render() </pre>	(6)

		<pre> { if(5<18) return(<h1>Msg 1</h1>); else return(<h2>Msg 2</h2>); } (OR) Conditional Operator: render() { const isLoggedIn = this.state.isLoggedIn; return (<div> {isLoggedIn ? <LogoutButton onClick={this.handleLogoutClick} /> : <LoginButton onClick={this.handleLoginClick} /> } </div>); } </pre>	
14.	b) i)	<p>Uncontrolled Component</p> <p>Program similar to the following are acceptable</p> <pre> import React, { useRef } from 'react' export default function FormUncontrolled() { const ID=useRef(); const name=useRef(); const Designation=useRef(); const Department=useRef(); const branch=useRef(); const mobile=useRef(); const email=useRef(); const display = (e) => { document.getElementById("show").innerHTML = <h3>ID: { ID.current.value }
 Name: { name.current.value }
 Designation: { Designation.current.value }
 Department: { Department.current.value }
 branch: { branch.current.value }
 mobile: { mobile.current.value }
 email: { email.current.value } e.preventDefault() } return (<div> <p id='show'></p> <form onSubmit={display}> ID:<input type='text' ref={ID} />
Name:<input type='text' ref={name} />
Designation:<input type='text' ref={Designation} />
Department:<input type='text' ref={Department} />
Branch:<input type='text' ref={branch} />
Mobile:<input type='text' ref={mobile} />
Email:<input type='text' ref={email} />
<input type='submit' /> </form> </div>) } </pre>	(10)

		<pre>export default function DisplayStdInfo() { //List of Objects const Stddata=[{id:101,Name:'Arun'},{id:102,Name:'Kumar'},{id:103,Name:'Ram'}] return (//map function displaying List of student objects (uses key attribute) Stddata.map((O)=><div key={O.id}> <h3>{O.id}</h3> <h2>{O.Name}</h2> </div>)) }</pre> <p>(OR)</p> <pre>const numbers = [1, 2, 3, 4, 5]; const updatedNums = numbers.map((number)=>{ return <li key={index}>{number} ; });</pre>	
15.	a) i)	<p>Routing</p> <p>Program similar to the following are acceptable</p> <p>(For each Page creation 1 Mark)</p> <p><u>Home.js</u></p> <pre>import React from 'react' export default function Home() { return (<div>Home Page</div>)}</pre> <p><u>About.js</u></p> <pre>import React from 'react' export default function About() { return (<div>About Page</div>)}</pre> <p><u>Department.js</u></p> <pre>import React from 'react' export default function Departments() { return (<div>Departments Page</div>)}</pre> <p><u>Facilities.js</u></p> <pre>import React from 'react' export default function Facilities() { return (<div>Facilities Page</div>)}</pre> <p><u>Emergency.js</u></p> <pre>import React from 'react'</pre>	(6)

		<pre>export default function Emergency() { return (<div> Emergency Page</div>)} <u>Contract.js</u> import React from 'react' export default function Contract() { return (<div> Contract Page</div>)} <u>Index.js</u> import React from 'react' import { BrowserRouter, Routes, Route, Link } from 'react-router-dom' import Home from './Home'; import About from './About'; import Departments from './Departments'; import Facilities from './Facilities'; import Emergency from './Emergency'; import Contract from './Contract'; import './Nav.css' export default function Nav() { return (<BrowserRouter> <div className='a'> <Link to='/'>Home</Link>{' '} <Link to='/About'>About</Link> {' '} <Link to='/ Departments > Departments </Link> {' '} <Link to='/ Facilities > Facilities </Link> {' '} <Link to='/ Emergency > Emergency </Link> {' '} <Link to='/Contract'>Contract</Link> </div> <Routes> <Route path='/' element={ <Home /> } /> <Route path='/About' element={ <About /> } /> <Route path='/ Departments ' element={ < Departments /> } /> <Route path='/ Facilities ' element={ < Facilities /> } /> <Route path='/ Emergency ' element={ < Emergency /> } /> <Route path='/Contract' element={ <Contract /> } /> </Routes> </BrowserRouter>)} </pre>	(4)
15.	a) ii)	<p>using useContext hook props drilling can be avoided in react. Program similar to the following are acceptable</p> <p><u>Grandparent.js</u></p> <pre>import React from 'react' import Child1 from './Child1' export const userContext = React.createContext("KEC"); export const idContext = React.createContext(123); export default function DParent() { return (<div> </pre>	(2)

		<pre> <userContext.Provider value={'KONGU'}> <idContext.Provider value={123}> <Child1 /> </idContext.Provider> </userContext.Provider> </div>) } </pre> <p>Child1.js</p> <pre> import React, { useContext, useRef, useState } from 'react' import { userContext,idContext } from './DParent'; export default function Child1() { const [name,setName]=useState(""); const id=useContext(idContext); useEffect(()=>console.log('component loaded'),[name]) return (<div><center> <input type='text' value={ name } onChange={(e)=>setName(e.target.value)}/> <input type='text' ref={ Age }/> user Name: { name }
 user Id: { id}</center> Age: { Age.current.value } </div>)) </pre>	(2)
			(2)
15.	b)	<p>Redux Program similar to the following is acceptable</p> <p>Counttype.js:</p> <pre> export const restore='restore' export const book='book' </pre> <p>CountAction.js:</p> <pre> import { restore,book } from './counttypes' const restoreAction = () => { return { type: restore } } const bookAction = () => { return { type: book } } export { restoreAction, bookAction } </pre> <p>CountReducer.js:</p> <pre> import { restore, book } from './counttypes' const initState = { count: 50, Name:'IT' } function CountReducer(state = initState, action) { switch (action.type) { case restore: return {if(state.count<50) ...state, count: state.count + 1 } } </pre>	(1)
			(3)
			(4)

	<pre> case book: {if (state.count === 50) { return { ...state, count: 0 }} else { return{ ...state,count: state.count - 1 } } } default: return state } } export default CountReducer <u>Library.js:</u> import React from 'react' import { connect } from 'react-redux' import {restoreAction,bookAction} from './CountAction' function Count(props) { return (<div>Count: {props.count}
 Name:{props.Name}
 <button onClick={props.restore}>return</button>
 <button onClick={props.book}>book</button> </div>) } constmapStateToProps=state=>{ return{ count:state.count, Name:state.Name } } constmapDispatchToProps=dispatch=>{ return{ restore:()=>dispatch(restoreAction()), book:()=>dispatch(bookAction()) } } export default connect(mapStateToProps,mapDispatchToProps) (Count); <u>Store.js:</u> import { createStore } from 'redux' import CountReducer from './CountReducer' function configureStore() { return createStore(CountReducer); } export default configureStore; </pre>	(5)
	<pre> import { createStore } from 'redux' import CountReducer from './CountReducer' function configureStore() { return createStore(CountReducer); } export default configureStore; </pre>	(3)