React and Material UI Assessment

1. Everything in React is a _____

a. Module

b. Component

c. Package

d. Class

2. ReactJS uses _____ to increase performance

a. Original DOM

b. Virtual DOM

c. Both 1 & 2

d. None of the Above

3. How many elements does a react component return?

a. 2 Elements

b. 1 Element

c. Multiple Elements

d. None of These

4. What is the state in React?

a. A persistent storage.

b. An internal data store (object) of a component.

c. Both 1 & 2

d. NA

5. What are the two ways that data gets handled in React?

a. State and Props

b. Services and Components

c. Both 1 & 2

d. NA

6. How can you access the state of a component from inside of a member function?

a. this.getState()

b. this.values

c. this.prototype.stateValue

d. this.state

7. Props are _____ into other components

a. Methods

b. Injected

c. Both 1 & 2

d. All of the above

8. What are the different phases of React component's lifecycle?

a. Initial Rendering Phase

b. Updating Phase

c. Unmounting Phase

d. All of the above

9. How to pass a parameter to an event handler or callback?

a. <button onClick={() => this.handleClick(id)} />

b. <button onClick={this.handleClick.bind(this, id)} />

c. Both A and B

d. None of These

10. Which of the following mounting methods are invoked before the component is inserted into DOM?

a. getInitialState, componentDidMount

b. componentWillMount

c. getInitialState

d. getInitialState, componentWillMount

11. What are forward refs?

a. Ref forwarding is a feature that lets some components take a ref they receive, and pass it further down to a child.

b. Ref forwarding is a feature that Enables Navigation System

c. Ref forwarding is a feature References are Made Static

d. None of the Above

12. What are hooks?

a. It's a Reference to a Component

b. That lets you use state and other React features without writing a class.

c. In Built React Components for UI Linking

d. NA

13. How to fetch data with React Hooks?

a. readData()

b. fetch()

c. useEffect

d. NA

14. What are the <Router> components of React Router v4?

a. <BrowserRouter>

b. <HashRouter>

c. <MemoryRouter>

d. All the Above

15. How to implement a NotFound page?

a. <Route page={NotFound} />

b. <Route component={NotFound} />

c. <Route component=[NotFound] />

d. None of the Above

Section II [Questions]: 2 Marks Each

1. Give the output

disabled = {false}

The element is fully enabled and interactive

2. What is the output of the below code ?

```
var courseVar=(props)=>
{
 return(
DSA Course 1
DSA Course 2
 );
}
ReactDOM.render(<DSA Course/>,mountNode)
```

Compiler error (Syntax error)

3. Which of the following is called a message that is closed inside the curly braces?

```
const messages="Hii Folks";
const element =<p>{messages}<p>
```

Expression language

4.

```
function helloWorld(){
 return <h1> Welcome to the New Era of Developing UI Using React </h1>
ReactDOM.render( ----,document.getElementById('root'));
 }
```

Syntax error

5. Which of the functions is used to change the state in React.js ?

useState()

Section III [Descriptive Questions]: 5 Marks Each

1. What are the features of React ? Benefits of Using React instead of other frameworks [Angular or Vue].

React is a popular JavaScript library (or framework-like tool) known for building dynamic, high-performance user interfaces, particularly for web and mobile apps. Here are the key features of React along with benefits of using React -

Features of React:

1) Component-Based Architecture - React builds user interfaces by breaking them into reusable and self-contained components. Each component manages its own structure and logic, making the codebase modular, easier to maintain, and scalable.

2) Virtual DOM - React uses a Virtual DOM, which is a lightweight in-memory copy of the actual DOM. When the app state changes, React updates this virtual DOM first, determines the minimal set of changes needed, and efficiently updates the real DOM. This approach improves rendering performance and responsiveness.

3) Declarative UI - React lets developers specify what the UI should look like for any given state declaratively. React then takes care of updating the UI whenever the data changes, simplifying UI management.

4) JSX (JavaScript XML) - React uses JSX, a syntax extension that allows writing HTML-like code within JavaScript. This makes the structure and logic of UI components clearer and more intuitive to write.

5) One-Way Data Binding - Data flows only from parent components down to child components through props, which enhances control and predictability in how data changes affect the UI.

6) Lifecycle Methods and Hooks - React provides lifecycle methods (in class components) and hooks (in functional components, e.g., useState, useEffect) to manage component behavior like initialization, updates, and cleanup in a clean and structured manner.

7) Performance Optimization - Beyond the Virtual DOM, React supports server-side rendering (SSR) and code-splitting, which help improve web app loading speed and SEO.

8) Rich Developer Tooling - It offers excellent tools for inspecting component trees, debugging props and state, and profiling performance, making development and optimization easier.

9) Cross-Platform Development - React's ecosystem includes React Native, which uses the same component-based model to build native mobile apps for iOS and Android, enabling code reuse across platforms.

Benefits:

1) Fast Development & Performance: React’s Virtual DOM and component reusability enable rapid development and efficient rendering, leading to smooth and responsive applications.

2) Flexibility to Customize: Unlike Angular’s all-in-one approach, React provides a minimal core, allowing developers to pick and choose libraries for tasks like routing and state management as needed.

3) Large Ecosystem and Strong Community Support: Backed by Meta (Facebook) and a vast community, React offers numerous libraries, tools, tutorials, and continuous innovation.

4) Cross-Platform Development: React Native allows you to develop mobile applications with the same React knowledge and component structure, increasing productivity across platforms.

5) SEO-Friendly: React's support for server-side rendering helps improve search engine rankings, a critical advantage

2. Explain Virtual DOM with suitable illustrations

Virtual DOM in React

The Virtual DOM is a lightweight copy of the real DOM (Document Object Model), stored in memory as plain JavaScript objects. React uses the Virtual DOM to optimize UI rendering by first updating this virtual version and then efficiently syncing only the necessary changes with the actual DOM.

1) Working of Virtual DOM

Initial Render:

React creates a Virtual DOM tree—a JavaScript object representation—mirroring the structure of the real DOM. This happens when the React app loads for the first time.

2) State/Props Update:

When the app's data changes (due to state or props updates), React generates a new Virtual DOM tree reflecting the updated UI.

3) Diffing (Reconciliation):

React compares the new Virtual DOM tree with the previous one using a diffing algorithm. This step identifies exactly what has changed between the two versions.

4) Efficient Update:

React updates only the parts that have changed in the real DOM rather than re-rendering the entire DOM tree. This selective update significantly improves performance.

Example Illustration:

Before update (Virtual DOM):

jsx

```
<div>
  <h1>Hello</h1>
</div>
```

After update (New Virtual DOM):

jsx

```
<div>
  <h1>Hi</h1>
</div>
```

Importance of Virtual DOM

Performance Boost

Efficiency

Smooth UI Experience

Example code:

```
import React, { useState } from 'react';
function Greeting() {
  const [text, setText] = useState('Hello');
  return (
    <div>
      <h1>{text}</h1>
      <button onClick={() => setText('Hi')}>Change Greeting</button>
    </div>
  );
}
```

3. What is a Component ? Explain Different types of Components in React

A component in React is an independent, reusable building block of a user interface that encapsulates a piece of the UI, including its structure, behavior, and rendering logic. It accepts inputs called props and returns React elements (like JSX) that describe what should appear on the screen, allowing you to build complex UIs by composing smaller, modular pieces.

Components promote reusability, maintainability, and efficient updates, as React only re-renders the parts that change.

Types of Components in React

React primarily has two main types of components: functional and class components.

1. Functional Components

These are simple JavaScript functions that return JSX elements. They are lightweight, easier to write, and can manage state or side effects using hooks (like useState or

useEffect). Functional components are preferred for most modern React development due to their simplicity and performance.

Example:

```
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}
```

This component takes a name prop and renders a greeting.

2. Class Components

These are ES6 classes that extend React.Component and include a render() method to return JSX. They support built-in state management (via this.state) and lifecycle methods (like componentDidMount). While still supported, they are less common now with the rise of hooks in functional components.

Example:

```
class Greeting extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}!</h1>;
  }
}
```

3. Higher-Order Components (HOCs)

HOCs are functions that take a component as input and return a new component with enhanced capabilities.They are used to reuse component logic (e.g., adding data fetching, authorization).

Example:

```
jsx
const withLoggedInCheck = (WrappedComponent) => {
  return function(props) {
    if (!props.loggedIn) {
      return <div>Please log in</div>;
    }
    return <WrappedComponent {...props} />;
  };
};
const Profile = (props) => <div>Welcome {props.user}</div>;
const ProfileWithAuth = withLoggedInCheck(Profile);
```

4. Presentational and Container Components

Presentational Components (Stateless): Focus on how things look; receive data via props and render UI. Often implemented as functional components.
Container Components (Stateful): Handle data fetching, state, and business logic; often class components or functional components with hooks.

4. Explain the Steps involved in Creating React App

Steps involve:
1) Install Node.js and npm.
2) Create React App:
   npx create-react-app my-app
3) Navigate to the App:
   cd my-app
4) Start Development Server:

npm start

5) Edit Source Code:

Use any code editor to modify components in the /src folder.

6) Build for Production:

npm run build

5. Explain Controlled Forms and UnControlled Forms in React with an example. Show the usage of Validations in React.

Controlled Forms - A Controlled Form in React is one where form data is managed by the React component's state. The form elements' values are set by state variables, and any user input updates the state via event handlers, ensuring that React is the "single source of truth" for the form data.

```
function ControlledForm() {
  const [value, setValue] = React.useState('');
  const handleChange = (e) => setValue(e.target.value);
  const handleSubmit = (e) => {
    e.preventDefault();
    alert('Value submitted: ' + value);
  };
  return (
<form onSubmit={handleSubmit}>
<input type="text" value={value} onChange={handleChange} required />
<button type="submit">Submit</button>
</form>
  );
}
```

Uncontrolled Forms - An Uncontrolled Form in React is one where form data is handled by the DOM itself. Instead of using React state, uncontrolled components use

refs to access form values from the DOM when needed (e.g., at form submission). This is closer to traditional HTML form behavior.

```
function UncontrolledForm() {
  const inputRef = React.useRef(null);
  const handleSubmit = (e) => {
    e.preventDefault();
    alert('Value submitted: ' + inputRef.current.value);
  };
  return (
<form onSubmit={handleSubmit}>
<input type="text" ref={inputRef} required />
<button type="submit">Submit</button>
</form>
  );
}
```

6. Create a Simple React App Using Material UI Components.

```
// src/App.js
import React from 'react';
import { Button, Typography, Container, Box } from '@mui/material';
function App() {
  return (
    <Container maxWidth="sm" sx={{ textAlign: 'center', marginTop: 8 }}>
      <Typography variant="h3" gutterBottom>
        Welcome to My Material UI App
      </Typography>

      <Typography variant="body1" gutterBottom>
        This is a simple example of using Material UI components in React.
      </Typography>
```

```
      <Box mt={4}>
        <Button variant="contained" color="primary" size="large">
          Click Me
        </Button>
      </Box>
    </Container>
  );
}

export default App;
```