Course name - Full stack development II
Course code – BCA57116

*** Multiple Choice Type Questions ***

1. Identify the main application of React in frontend development.
   a) Backend development
   b) Styling web pages
   c) Building user interfaces
   d) Managing databases
2. Identify the company that developed React JS.
   a) Microsoft
   b) Google
   c) Meta (Facebook)
   d) IBM
3. Identify the prerequisites for learning React JS.
   a) Java
   b) Python
   c) JavaScript
   d) C++
4. Select the correct npm command to install React globally from a list of options.
   a) npm install -g react
   b) npm add react
   c) install react
   d) npm install react-global
5. Identify the role of JSX in building React components.
   a) JavaScript file
   b) A templating engine
   c) A JavaScript syntax extension
   d) A server-side tool
6. Identify the folder that contains the main application component by default in a React project.
   a) src
   b) public
   c) dist
   d) config
7. Identify the file you must edit to define the root component rendered in a React application.
   a) index.js
   b) App.css
   c) index.html
   d) main.js
8. Identify the role of ReactDOM.createRoot() in initializing a React application.
   a) Creates a virtual DOM
   b) Renders components
   c) Enables concurrent rendering
   d) Connects database
9. Select the correct JSX comment syntax from the given options.
   a) // Comment
   b) {/* Comment */}
   c) / Comment */
   d) None of the above
10. Identify the correct way to use a JavaScript variable within JSX code.
    a) {{variable}}
    b) {variable}
    c) [[variable]]
    d) $variable
11. Identify the characteristics of a functional component in React.
    a) A component using class
    b) A function that returns JSX
    c) A function with no return
    d) A service

12. Select the correct keyword used to define a class component from a list.
    a) new
    b) class
    c) const
    d) component
13. Identify the default root HTML element of a React application.
    a) #app
    b) .root
    c) #root
    d) #main
14. Select the correct NPX command for initializing a new React app.
    a) npm create react-app
    b) create react app
    c) npx react init
    d) npx create-react-app
15. Identify the types of files typically stored in the public directory of a React app.
    a) Hosts build files
    b) Hosts static assets
    c) Contains JSX
    d) Holds tests
16. Identify the folder that contains the index.html file in a React application.
    a) /src
    b) /node_modules
    c) /public
    d) /build
17. Identify the main distinctions between JSX and HTML.
    a) JSX uses className instead of class
    b) JSX is written in .xml files
    c) JSX ignores case sensitivity
    d) JSX cannot use JavaScript
18. Identify the command used to start a React application locally.
    a) npm start
    b) npm build
    c) node start
    d) react run
19. Select the correct statement about JSX.
    a) JSX allows raw HTML inside JS
    b) JSX converts HTML tags to React.createElement calls
    c) JSX is compiled by CSS
    d) JSX only runs on Node.js
20. Identify the command used to build a production version of a React application.
    a) npm prod
    b) npm build
    c) npm package
    d) npm deploy
21. Select the correct export syntax from multiple options.
    a) export className
    b) return component
    c) export default ComponentName
    d) module.export = ComponentName
22. Identify the file that defines meta tags such as <title> in a default React project.
    a) index.js
    b) App.js
    c) index.html
    d) App.css
23. Identify the role of the App.css file in styling React components.
    a) Defines routing
    b) Holds component logic
    c) Contains default styles
    d) Manages API calls
24. Identify the error that occurs when JSX elements are returned without a wrapping tag.
    a) Error: Adjacent JSX elements must be wrapped
    b) Only the first element is rendered

c) The app crashes
d) The elements merge

25. Select the option that is not allowed directly in JSX.
    a) JavaScript expressions
    b) Loops using for
    c) Conditional rendering using ternary
    d) Comments using {/* */}
26. Identify the reasons why function components are preferred in modern React.
    a) Require more memory
    b) Have lifecycle methods by default
    c) Simpler syntax and easier testing
    d) Harder to reuse
27. Identify the appropriate location for storing reusable components in a React file structure.
    a) /public
    b) /node_modules
    c) /components
    d) /build
28. Identify the syntax or techniques used for conditional rendering in JSX.
    a) if statements
    b) for loops
    c) Ternary or logical AND
    d) while loop
29. Identify the correct format for self-closing tags in JSX.
    a) <img>
    b)
    c) <hr>
    d) <meta>
30. Select the correct statement about using variables in JSX.
    a) They must be global
    b) Must use this. always
    c) Declared using let, const, or var
    d) Cannot be used
31. Identify the typical first line in most React component files.
    a) import react
    b) class App
    c) export default
    d) const variable
32. Identify the correct syntax for passing props to a React component.
    a) Through classes
    b) As function parameters
    c) Using useProp
    d) By importing them
33. Identify the file that contains the main application layout in a React project.
    a) App.js
    b) Main.js
    c) React.js
    d) Layout.js
34. Identify the correct syntax for writing a comment inside JSX.
    a) // This is comment
    b) <!-- comment -->
    c) {/* comment /}
    d) / comment */
35. Identify what the .js extension represents in a React project.
    a) It contains HTML only
    b) It contains CSS
    c) It contains JavaScript or JSX
    d) It contains server logic
36. Identify the purpose of using ReactDOM in rendering components.
    a) Defines CSS classes
    b) Handles HTTP requests
    c) Renders components into the DOM
    d) Adds dynamic routing
37. Identify the key similarity between JSX and HTML.
    a) Both are used to define server-side logic

b) Both allow writing elements with tag-based syntax
c) Both use JavaScript syntax for styling
d) Both require a separate CSS file

38. Identify the tool typically used to compile JSX into JavaScript.
    a) Webpack
    b) Babel
    c) Node.js
    d) ESLint
39. Identify the type of value a React component must return.
    a) A string
    b) A JSX element or null
    c) A class instance
    d) A style object
40. Identify the correct syntax for using props in a function component.
    a) this.props
    b) props.variable
    c) this.variable
    d) props()
41. Identify the result of omitting the return in a functional component.
    a) Nothing is rendered
    b) Component is auto-rendered
    c) All props are ignored
    d) Styling is lost
42. Identify the syntax used to define a functional component in React.
    a) class MyComponent extends React.Component {}
    b) React.createComponent(MyComponent)
    c) function MyComponent() { return Hello ; }
    d) component MyComponent() { return Hello ; }
43. Identify the advantages that make function components preferable over class components.
    a) Need less styling
    b) Integrate easily with hooks and require less boilerplate
    c) Cannot return multiple elements
    d) Require constructor method
44. Identify the correct way to define and assign inline styles in JSX.
    a) With HTML style tags
    b) By external stylesheet
    c) Using a style object
    d) Using <css> tag
45. Identify the correct format for assigning inline styles to elements in React.
    a) Strings
    b) Plain text
    c) JSON
    d) CamelCase JavaScript objects
46. Identify the syntax used for camelCase styling in React inline styles.
    a) font-size
    b) fontSize
    c) font_size
    d) fontsize
47. Identify the correct syntax for including a CSS stylesheet in React.
    a) Using require()
    b) Using <link>
    c) Using import './App.css'
    d) Using useCSS()
48. Identify the correct way to reference a class from a CSS Module in a React component.
    a) class="style.header"
    b) className={style.header}
    c) className="header"
    d) style="header"
49. Identify the library commonly used for component-level CSS-in-JS styling in React.
    a) ReactCSS
    b) Styled Components
    c) ReactBootstrap
    d) BootstrapJS

50. Identify the main features and advantages of react-bootstrap.
    a) Converts Bootstrap to HTML
    b) Provides Bootstrap components for React
    c) Installs jQuery
    d) Adds JavaScript routing
51. Identify the npm command used to add React Bootstrap to a project.
    a) npm bootstrap-react
    b) npm install bootstrap-react
    c) npm install react-bootstrap bootstrap
    d) npm add react-css
52. Identify the purpose of React Table in managing data display.
    a) Creating styled buttons
    b) Managing layout
    c) Rendering flexible, customizable data tables
    d) Creating navigation
53. Identify the command used to install React Table.
    a) npm i react-table
    b) npm install react-table
    c) yarn table
    d) npm get table
54. Identify the functionality React Router provides for navigation.
    a) Backend routing
    b) Client-side routing
    c) DOM rendering
    d) Data fetching
55. Select the correct package name from a list of options for installing React Router.
    a) react-router
    b) react-route-dom
    c) react-router-dom
    d) router-react
56. Identify the role props play in React for passing data.
    a) A hook
    b) A router
    c) A way to pass data to components
    d) A CSS file
57. Identify the syntax used to send props to child components.
    a) As attributes in JSX
    b) Through context only
    c) Using Redux
    d) Through state only
58. Identify the correct syntax to destructure props in a functional component.
    a) function MyComponent(props) { const name = props.name; }
    b) const name = this.props.name;
    c) function MyComponent({ name }) { return <h1>{name}</h1>; }
    d) component({ name }) => <h1>{name}</h1>
59. Identify the tools or libraries used for prop validation (e.g., PropTypes).
    a) propTypes
    b) validateProps
    c) props.type()
    d) React.validate()
60. Identify the package that must be imported to use BrowserRouter in React.
    a) react-dom
    b) react-router
    c) react-router-dom
    d) react-router-native
61. Identify the correct way to define a route in React Router from the following code snippets.
    a) <Route path="/" component={Home} />
    b) <Path route="/" />
    c) <Router path="/" />
    d) <Link route="/" />
62. Identify the correct use of the Link component from a set of code snippets in a React project.
    a) Loads external pages
    b) Creates navigational links without reloading

c) Creates HTTP request
d) Renders HTML

63. Identify the correct use of props.children in React from the following options.
    a) Router element
    b) Nested elements passed between opening and closing component tags
    c) Component name
    d) prop validation function
64. Select the statement that correctly describes props in React.
    a) Props are mutable and can be changed within the child component.
    b) Props are read-only and passed from parent to child.
    c) Props are used to pass data from child to parent.
    d) Props are used to manage internal component state.
65. Identify the library most commonly used for validating component props in React.
    a) redux-types
    b) react-validate
    c) prop-types
    d) type-check
66. Identify the consequences of omitting a required prop in a component that uses propTypes.
    a) React throws a runtime error
    b) React logs a warning in console
    c) React reloads
    d) App stops rendering
67. Identify the primary difference between state and props in a functional component.
    a) State is passed down
    b) Props are mutable
    c) State is internal and mutable
    d) State is readonly
68. Identify the correct syntax for declaring a state variable in a functional React component.
    a) this.state = ...
    b) let state = useState()
    c) const [value, setValue] = useState()
    d) const state = setState()
69. Identify the React hook responsible for managing state in function components.
    a) useContext
    b) useRef
    c) useState
    d) useEffect
70. Identify the syntax used to update a state variable declared using the useState hook.
    a) state.value = 10
    b) this.setState(10)
    c) setValue(10)
    d) value = setValue(10)
71. Identify a common rule you must follow when using state inside a React functional component.
    a) Call useState in conditionals
    b) Never call setState
    c) Only call useState at top-level of component
    d) Can be nested in loops
72. Identify the main purpose of using state in functional React components.
    a) Holds static values
    b) Tracks internal dynamic values of components
    c) Manages routing
    d) Applies styling
73. Identify the outcome of calling a state updater function (like setState or setCount) in React.
    a) Page reload
    b) Function call
    c) Component re-render
    d) Component unmount
74. Identify the method by which a child component can trigger state updates in its parent.
    a) Directly access it
    b) Use props
    c) Lift state up using callback functions
    d) Use context only
75. Identify the traits that define a reusable component in React development.
    a) Tied to one view

b) Fixed for single use
c) Can be reused across the app with different data via props
d) Applies global state

76. Identify the React hook that enables updating of state variables within a functional component.
a) useEffect
b) useContext
c) useState
d) useUpdate

77. Identify what must be imported and defined to successfully use styled-components in a React component.
a) Inline style tags
b) Traditional CSS syntax
c) Template literals with CSS
d) HTML5 classes

78. Identify the correct JSX syntax for passing a name prop from a parent to a child component.
a) <Child props={value} />
b) <Child value={props}>
c) <Child value={props.value} />
d) <Child name="value" />

79. Identify the key restriction on where useState can be called within a React component.
a) Can be used anywhere
b) Should be called inside event handler
c) Must be used at the top level of component
d) Can be used after return

80. Identify how React Router v6 determines which route to render when multiple matches are possible.
a) All are rendered
b) Only the first is rendered
c) Routes are ignored
d) The exact match route is rendered

81. Select the correct statement that describes the relationship between props and state in React.
a) Both are mutable
b) Props are mutable, state is not
c) Props are read-only, state is mutable
d) State cannot be updated

82. Identify the correct syntax to define default props in React components.
a) static defaultProps = {}
b) props.defaults()
c) useDefaultProps()
d) component.default()

83. Select the styling approach that applies styles locally to a component in React.
a) Global CSS
b) CSS Modules
c) HTML styling
d) External CDN

84. Identify the correct syntax for writing an arrow function in JavaScript.
a) function() => {}
b) () => {}
c) () = {}
d) => () {}

85. Identify scenarios where using arrow functions in event handlers prevents common issues like incorrect this binding.
a) They automatically bind this
b) They delay execution
c) They refresh the app
d) They fetch data

86. Identify the role of the handleChange function in managing input value changes in React.
a) Submits the form
b) Calls a function on input change
c) Prevents re-render
d) Binds event with HTML only

87. Identify the method used to prevent the default action of a form in JavaScript or React.
a) stopDefault()
b) preventReload()

c) preventDefault()
d) disableForm()

88. Identify the function of the name attribute in linking input fields to submitted form data.
    a) Validation
    b) Binding value to corresponding state
    c) Styling
    d) Route identification
89. Identify the React hook commonly used for managing form state.
    a) useState
    b) useRef
    c) useEffect
    d) useContext
90. Identify the default parameter passed to event handlers in React and its purpose.
    a) Event object
    b) State object
    c) Props
    d) DOM node
91. Select the correct arrow function syntax for an event handler from the given options.
    a) handleClick() => {}
    b) () => { console.log("Clicked") }
    c) = () {}
    d) onClick = function => {}
92. Identify the React technique used to manage and access form data through component state.
    a) From the DOM
    b) Via event.target.value
    c) Using a router
    d) Through props only
93. Identify key advantages of controlled components in ensuring consistent state management in forms.
    a) Reduce performance
    b) UI and state stay in sync
    c) Force reload
    d) Require Redux
94. Identify the main features and use cases of the React Developer Tools extension.
    a) Viewing backend logic
    b) Tracking Redux changes only
    c) Inspecting component hierarchy and state
    d) Running React CLI
95. Identify the command that prepares a React app for deployment.
    a) Writing backend code
    b) Building the app using npm run build
    c) Adding more routes
    d) Modifying node_modules
96. Identify the default folder that contains the production build of a React application.
    a) /public
    b) /src
    c) /build
    d) /index
97. Select the platform(s) that support free React app deployment from the list provided.
    a) Photoshop
    b) VS Code
    c) GitHub Pages
    d) Android Studio
98. Identify the configuration file required to manage custom redirects and rewrites on Netlify or Vercel.
    a) config.xml
    b) netlify.json
    c) _redirects
    d) build.js
99. Identify the benefits of using a .env file in React for setting environment-specific variables.
    a) Manage styles
    b) Handle CSS modules
    c) Store environment variables like API keys
    d) Import HTML
100. Identify the role of the build script defined in the package.json file.
    a) Starts the server

b) Builds production-optimized app
c) Runs unit tests
d) Adds new modules

101. Choose the correct function used to fetch data in modern JavaScript applications.
a) XMLHttpRequest
b) getJSON()
c) fetch()
d) load()

102. Choose the method commonly used to send data in a fetch POST request.
a) data: {}
b) payload
c) body
d) content

103. Choose the correct programming language used to develop the backend API in this module.
a) JavaScript
b) Java
c) PHP
d) Angular

104. Choose the commonly used method in PHP to establish a connection with a MySQL database.
a) PDO or mysqli
b) MongoDB
c) SQLServer
d) pg_connect()

105. Choose the appropriate HTTP method used for submitting new data to an API.
a) GET
b) PUT
c) POST
d) DELETE

106. Choose the appropriate header to set when sending JSON in an API request.
a) Accept-Type
b) Content-Type: text/plain
c) Content-Type: application/json
d) Content-Length

107. Choose the purpose of JSON.stringify() when used in a fetch request.
a) Parses JSON
b) Converts JS object to string
c) Minifies HTML
d) Compresses data

108. Choose the three core components that make up the MVC architecture.
a) Middleware, Views, Classes
b) Model, View, Controller
c) Main, Virtual, Compiler
d) Map, Validator, Controller

109. Identify the role of AngularJS in building single-page applications.
a) Backend library
b) JavaScript framework for building SPAs
c) CSS framework
d) PHP engine

110. Choose the correct distinctions between AngularJS and Angular (2+).
a) Based on TypeScript
b) Uses JSX
c) Runs on Node.js
d) AngularJS is older and based on JS

111. Choose the correct HTML tag or directive used to define an AngularJS app.
a) <html>
b) <body>
c) <ng-app>
d) <angular>

112. Choose the correct function of the ng-model directive in AngularJS.
a) Attaches class styles
b) Binds input to scope variable
c) Triggers events
d) Sends API requests

113. Choose the correct syntax to output data using AngularJS expressions in a web page.
    a) [[name]]
    b) { name }
    c) {{ name }}
    d) <%= name %>
114. Choose the correct explanation of AngularJS's forgiving behavior during runtime expression evaluation.
    a) Ignores undefined data gracefully
    b) Automatically corrects errors
    c) Removes loops
    d) Converts strings to numbers
115. Choose the correct AngularJS expression used to bind an object property in the view.
    a) {{ object[property] }}
    b) [[object.property]]
    c) {{ object::property }}
    d) object->property
116. Choose the AngularJS directive used to loop through items in an array.
    a) ng-if
    b) ng-model
    c) ng-loop
    d) ng-repeat
117. Choose the AngularJS directive used to bind an app to the HTML document.
    a) ng-app
    b) ng-start
    c) app-ng
    d) angular-init
118. Choose the part of the MVC architecture in AngularJS that handles data logic.
    a) View
    b) Model
    c) Controller
    d) DOM
119. Choose the correct file needed to include AngularJS in an HTML document.
    a) angular.html
    b) ng.module.js
    c) angular.min.js
    d) module.angular.js
120. Choose the default delimiter used for AngularJS expressions.
    a) <% %>
    b) { }
    c) {{ }}
    d) [[]]
121. Choose the correct AngularJS expression that represents a valid numeric operation.
    a) {{ 3 + 5 }}
    b) {3+5}
    c) <%= 3+5 %>
    d) [[3+5]]
122. Choose the correct AngularJS expression that displays the full name from an object.
    a) {{ user.fullname }}
    b) {{ fullname.user }}
    c) [[user.fullname]]
    d) <%= user.fullname %>
123. Choose the correct explanation of object binding in AngularJS.
    a) Combining multiple views
    b) Assigning string to class
    c) Binding object data to UI
    d) Creating nested routes
124. Choose the correct behavior of AngularJS when it encounters an invalid expression.
    a) Throws fatal error
    b) Ignores it silently
    c) Shows warning popup
    d) Removes scope
125. Choose the correct output of the AngularJS expression {{ 'Hello' + ' AngularJS' }}.
    a) HelloAngularJS
    b) "Hello AngularJS"

c) Hello AngularJS
d) {{ "Hello AngularJS" }}

126. Choose the AngularJS directive that is used to dynamically render lists.
a) ng-model
b) ng-bind
c) ng-loop
d) ng-repeat

127. Choose the key feature of AngularJS expressions from the following options.
a) Can evaluate complex logic
b) Cannot use operators
c) Run outside the scope
d) Allow only strings

128. Choose the correct purpose of the ng-bind directive in AngularJS.
a) Loops through values
b) Binds content of an HTML element to scope variable
c) Triggers click events
d) Declares filters

129. Choose the correct syntax for binding a value using the ng-bind directive in AngularJS.
a)
b)
c) <ng-bind="message">
d) bind.message()

130. Choose the correct output of the AngularJS expression {{ 10 / 2 }}.
a) 2
b) 5
c) 0.5
d) 45698

131. Choose the correct AngularJS expression that includes both a string and a number.
a) {{ "Count: " + 5 }}
b) [["Count" + 5]]
c) {{ 5 + "Count" }}
d) 5 + "Count"

132. Choose the correct advantage of using ng-init in AngularJS during development.
a) Declares new modules
b) Simplifies testing by initializing values
c) Disables validations
d) Enhances animations

133. Choose the correct AngularJS directive that conditionally adds an element to the DOM.
a) ng-show
b) ng-if
c) ng-bind
d) ng-model

134. Choose the correct function of the ng-show directive in AngularJS.
a) Always hides element
b) Shows element based on condition
c) Submits a form
d) Applies filter

135. Choose the correct statement that describes the primary role of a controller in AngularJS.
a) Styling
b) Managing scope and logic
c) Handling HTTP calls
d) Compiling templates

136. Choose the correct definition and role of $scope in AngularJS.
a) A built-in CSS class
b) A filter
c) A bridge between controller and view
d) A model validator

137. Choose the correct outcome of assigning a function to $scope within an AngularJS controller.
a) It becomes inaccessible in view
b) It throws an error
c) It can be called from the view
d) It deletes the controller

138. Choose the correct syntax for passing parameters to a $scope function in AngularJS.
a) $scope.myFunc[]

b) ng-func="myFunc(param)"
c) $scope.myFunc = function(param) {}
d) param.scope = function()

139. Choose the correct outcome when both parent and child controllers in AngularJS define the same $scope variable.
a) Conflict error
b) Inherited by child
c) Child overrides parent's value in its own scope
d) Values merge

140. Choose the correct statement about using multiple controllers in AngularJS.
a) Each controller has its own $scope
b) They all share the same $scope
c) Controllers overwrite each other
d) Cannot be defined in the same module

141. Choose the correct AngularJS filter that transforms text to uppercase.
a) upperCase
b) capital
c) uppercase
d) toUpperCase

142. Choose the AngularJS filter that displays a subset of array items.
a) slice
b) filter
c) select
d) ng-subset

143. Choose the correct function of the orderBy filter in AngularJS.
a) Reverses the array
b) Sorts array items
c) Adds elements
d) Converts to uppercase

144. Choose the correct way to define a custom filter in AngularJS.
a) app.directive()
b) app.filter('name', function() {})
c) app.ngFilter()
d) scope.filter()

145. Choose the correct purpose of the currency filter in AngularJS.
a) It converts a number into a percentage format.
b) It formats a number as a currency string, including a currency symbol.
c) It sorts a list of numbers in descending order.
d) It removes all non-numeric characters from a string.

146. Choose the option that is not a built-in filter in AngularJS.
a) filter
b) limitTo
c) sort
d) date

147. Choose the correct directive used to define a form in AngularJS.
a) ng-model
b) ng-class
c) ng-form
d) ng-submit

148. Choose the correct directive used in AngularJS to apply a CSS class based on a condition.
a) Using ng-if
b) Using ng-style
c) Using ng-class
d) Using ng-model

149. Choose the correct purpose of the ng-submit directive in AngularJS.
a) It binds input values directly to the database.
b) It executes a specified expression or function when the form is submitted.
c) It prevents form submission and disables the submit button.
d) It applies validation styling to form fields.

150. Choose the correct approach to define a custom validation directive in AngularJS.
a) Use only ngModel without directives and write logic in the controller.
b) Use require: 'ngModel', define the link function, and set custom logic via $validators.
c) Apply validation using CSS classes only.
d) Import external JavaScript files without defining a directive.

*** Short answer questions ***

ReactJS Fundamentals

1. Describe the benefits of using ReactJS in web development.
2. Discuss the role of JSX in React.
3. Explain the purpose of the class and function components in React.
4. Distinguish between HTML and JSX with an example.
5. Identify the purpose of directory structure in a React application.
6. Interpret how JavaScript variables are used inside JSX.
7. Describe the significance of virtual DOM in React.
8. Illustrate the use of code snippets in a React project.
9. Evaluate the advantages of using ReactJS over traditional JavaScript-based UI development.
10. Defend the decision to choose ReactJS for building modern single-page applications.

ReactJS Components & Props

11. Describe the purpose of a module in a React application.
12. Identify the role of props in ReactJS.
13. Discuss the concept of reusable components in React.
14. Describe how to pass props from a parent to a child component.
15. Explain the significance of useState in functional components.
16. Clarify the difference between state and props in React.
17. Discuss the use of prop validation in React.
18. Describe the rules for using state in React.
19. Analyze how props are used to pass data in React components.
20. Examine how React Router enhances single-page applications.

React Forms & Event Handling

21. Explain the concept of controlled components in React forms.
22. Describe how event handling works in React.
23. Discuss the role of arrow functions in handling events.
24. Explain the significance of onChange in form elements.
25. Illustrate how state updates enable real-time form validation.
26. Discuss the purpose of the value attribute in form controls.
27. Summarize how to manage multiple input fields in a form.
28. Describe how handleSubmit is implemented in React forms.
29. Explain the purpose of data binding in React.

React API Handling

30. Describe how React fetches data from an API.
31. Illustrate how useEffect() supports API calls in React.
32. Describe how to create a simple API using PHP and MySQL.
33. Explain the function of JSON in API communication.

AngularJS & Comparative Concepts

34. Summarize how MVC architecture is applied in AngularJS.
35. Describe how event handling differs between React and AngularJS.
36. Explain how React handles dynamic data updates.

React Practical Implementation

37. Implement a React form that captures user name and email.
38. Demonstrate the use of arrow functions to handle a button click event in React.
39. Use fetch API in React to retrieve user data from a JSON API.
40. Write a React code snippet to send form data to a server using fetch POST.
41. Implement a dropdown form in React and store selected value in state.
42. Apply basic validation logic to a React form.
43. Use a button event to toggle a message display in React.
44. Develop a contact form that uses local state and alerts the submitted values.
45. Justify the use of state in managing form inputs in React.

React Best Practices & Evaluation

46. Evaluate the performance of controlled vs uncontrolled components in form handling.
47. Evaluate how data binding in React differs from AngularJS.
48. Evaluate the best practices in organizing form validation logic in React.
49. Judge the use of data binding in React compared to traditional JavaScript methods.
50. Assess the benefits of using PHP and MySQL for backend API creation in a React project.

Long answer questions

React JS Questions

1. Explain how controllers are represented in a ReactJS-based architecture.
2. Identify how state changes trigger re-rendering in React.
3. Describe how to create a basic form in React.
4. Explain the use of developer tools for debugging React forms.
5. Discuss the advantages of using controlled forms over uncontrolled forms.
6. Explain how input elements interact with the component state.
7. Apply event handlers to manage state updates in a text input field.
8. Use React Developer Tools to inspect component props and state.
9. Create a basic React form with controlled components.
10. Apply the concept of lifecycle (via useEffect) to fetch data once a component is mounted.
11. Evaluate the effectiveness of arrow functions in handling events in React.
12. Critique the deployment process of a React app on a server.
13. Assess the pros and cons of using useState for form field control.
14. Critique the approach of handling multiple form inputs with a single event handler in React.
15. Justify using APIs in a React application to fetch dynamic form data.
16. Evaluate the use of useEffect for synchronizing API data with component state.
17. Assess the effectiveness of rendering form errors conditionally in JSX.
18. Justify the use of modular form components in React applications.
19. Evaluate the reliability of API error handling using try-catch in async React functions.
20. Critique how form data is structured and submitted using React and APIs.
21. Evaluate the effectiveness of React in creating dynamic forms.
22.

22. Determine the advantages of integrating AngularJS expressions for simple data rendering.
23. Evaluate the use of object binding in AngularJS in comparison to React state handling.
24. Examine how form validation improves overall data quality in React applications.
25. Assess the effectiveness of MVC architecture in AngularJS for frontend development.
26. Design a form in React that collects user profile data and validates required fields.
27. Create a component that fetches data from an external API and displays it in a table format.
28. Build a complete data-binding example using AngularJS expressions and object models.
29. Develop a React component that dynamically renders input fields based on API data structure.
30. Create a small AngularJS application to bind a list of student records and filter them.
31. Evaluate the use of conditional directives in AngularJS and their impact on DOM manipulation.
32. Assess the advantages and limitations of using multiple controllers in AngularJS applications.
33. Critically evaluate the role of $scope object in controller-based programming in AngularJS.
34. Evaluate the significance of nested controllers and scope inheritance in complex AngularJS applications.
35. Judge the effectiveness of built-in filters for real-time data transformation in AngularJS views.
36. Evaluate the impact of custom filters on user experience in AngularJS-based web apps.
37. Examine how AngularJS form events contribute to dynamic form interactions.
38. Critically evaluate the use of CSS classes in AngularJS to reflect real-time data state.
39. Evaluate the effectiveness of AngularJS's validation techniques in maintaining data integrity.
40. Assess the architectural benefits of using services in AngularJS.
41. Create a custom AngularJS directive that hides content based on a condition.
42. Build a reusable AngularJS filter to display capitalized text.
43. Create a multi-controller AngularJS app with scope inheritance.
44. Construct a form in AngularJS with custom validations for email and password.
45. Develop a directive in AngularJS to change text color based on user interaction.
46. Create a working example of dependency injection in AngularJS using multiple services.
47. Design a simple AngularJS application using a built-in filter like orderBy and limitTo.
48. Construct an AngularJS app that toggles form visibility using conditional directives.
49. Create a custom filter in AngularJS that hides sensitive parts of a string like an email.
50. Develop a form in AngularJS styled using CSS classes that change dynamically.
