Basics of React:

1. What is React and its basics?
2. Why is React so popular?
3. Comparison of Angular and React.
4. What are state and props, and what is the difference between them?
5. What are stateless and stateful components?
6. Explain React Fragment.

React Concepts and Features:

7. What are lifecycle methods in class components?
8. What are hooks in React?
9. What is Redux, and how does it work?
10. Explain saga and thunk in Redux.
11. What is the Virtual DOM, and how does it differ from the real DOM?
12. Explain prop drilling.
13. What is the Context API in React?
14. What are Higher Order Components (HOCs)?
15. What is the difference between class and functional components in React?
16. Explain useEffect hook.
17. What is useRef hook?
18. How does lazy loading work in React routing?
19. How do you handle errors in React?
20. What is the use of 'super' in class-based components?
21. What are the limitations of React?
22. Explain the difference between fetching data with Fetch and Axios.
23. How do you handle routing in React?
24. Explain lazy routing in React.
25. What is the difference between className and class in React CSS?

React Ecosystem and Tooling:

26. How does Redux work in managing state?
27. What is the difference between Redux and Redux Toolkit?
28. Explain useMemo, useContext, and useCallback hooks.
29. How does data flow in React?
30. What is the difference between a framework and a library?

Miscellaneous:

31. Explain the difference between TypeScript and JavaScript.
32. Explain server-side rendering (SSR) in React.
33. How do you handle errors gracefully in React using error boundaries?
34. What are some techniques for code splitting in React?
35. How do you style React components using CSS-in-JS libraries?
36. How do you handle forms in React?

37. What are some common techniques for testing React components?
38. What are some techniques for performance optimization in React?
39. How do you internationalize React applications?
40. How do you implement authentication and authorization in React applications?

Possible questions related to each topic that an interviewer might ask:

Basics of React:

1. Can you explain what React is and its core principles?
2. Why do you think React has gained so much popularity in recent years?
3. Could you compare React with another popular frontend framework like Angular and highlight their differences?
4. Explain the concepts of state and props in React. Can you provide examples of when you would use each?
5. What distinguishes stateless and stateful components in React?
6. How would you use React Fragment in your components?

React Concepts and Features:

7. Walk me through the lifecycle methods of class components in React.
8. What are hooks in React, and why were they introduced?
9. How does Redux help manage state in a React application, and what are its core principles?
10. What are the differences between using saga and thunk middleware in Redux?
11. Describe the Virtual DOM and its advantages over the real DOM.
12. How would you prevent prop drilling in a large React application?
13. Explain the purpose of the Context API in React and when you might use it.
14. What are Higher Order Components (HOCs), and how do they enhance component reusability?
15. What factors would you consider when choosing between class and functional components in React?
16. When would you use the useEffect hook in React, and how does it differ from other lifecycle methods?
17. How does the useRef hook differ from useState in React?
18. What is lazy loading in React routing, and why is it important?
19. How do you handle errors in a React application?
20. What does the 'super' keyword do in class-based React components?

React Ecosystem and Tooling:

21. Can you explain how Redux manages state in a React application step by step?
22. What advantages does Redux Toolkit offer over traditional Redux?
23. When would you use useMemo, useContext, and useCallback hooks in React?
24. Explain the concept of data flow in React and how components communicate with each other.
25. Differentiate between a framework and a library in the context of React.

Miscellaneous:

26. What are the main differences between TypeScript and JavaScript, and why might you choose one over the other for a React project?
27. Describe how server-side rendering (SSR) works in React and its benefits.
28. How do you handle errors gracefully in a React application using error boundaries?

29. Can you explain code splitting and its advantages in a React application?
30. How would you style React components using CSS-in-JS libraries like styled-components?
31. What are the common techniques for handling forms in React?
32. How do you approach testing React components, and what tools might you use?
33. What are some techniques you can use to optimize the performance of a React application?
34. How would you internationalize a React application to support multiple languages?
35. Explain how you would implement authentication and authorization in a React application.