Bookmarks
1.Reactjs = akreact
2.virtual dom works = akvdom
3.pass data from parent to child component and from child component = akptoc
4.how to achieve life cycle methods in functional components = aklifec
5.Hooks in react? = akhook
6.Difference between useState and useRef? = akusr
7.Difference between useEffect and useLayoutEffect? akuele
8.Difference between class based components(stateful) and functional based components(stateless)? = akcf
9.Difference between controlled and uncontrolled components in react exaplian? = akcunc
10.how to handle errors in react hooks and explain? = akerb
11.higher order components? = akhoc
12.Custom Hooks? = akch
13.what are the optimization techniques in react? = akots
14.How to avoid rerendeing in class components and functional components? akrercf
15.Difference between React.Memo and useMemo? = akrmum
16.react router? = akrr
17.state and props? = aksp
18.can we directly update the state and why? what happens if we directlu update? = akds
19.is setState asynchronus and explain how?(Note:same question and answer for useState) = aksa
20.what are forward Refs and use cases? = akfr
21.synthetic events in react? = aksy
22.Difference between shadom dom and virtual dom? = akshad
23.Strict Mode? = aksmode
24.React Fragments? = akfrag
25.React Portals? = akport
26.Protected Routes,authentication and authorization in jwt? = akjwt
27.difference between Authentication and authorization? = akaa
28.What is Redux? = akred
29.what is redux and advantages over the context api? = akcont
30.Redux thunk middleware? = akmid

1.Rectjs Features and disadvantages? akreact

ReactJS is an open-source JavaScript library used for building user interfaces, particularly single-page applications where data can change over time without requiring a page reload. React allows developers to create large web applications that can update and render efficiently in response to data changes.

Features of ReactJS

Component-Based Architecture:

ReactJS is all about the components. Components are self-contained modules that render some output. components can be nested, managed, and handled independently, making the code more readable and easier to maintain.

Virtual DOM:

React uses a virtual DOM to improve performance. Instead of manipulating the actual DOM directly, React creates a virtual copy of the DOM in memory. When the state of a component changes, React updates the virtual DOM and then efficiently updates the real DOM to reflect those changes. This minimizes the number of direct DOM manipulations.

JSX (JavaScript XML):

JSX is a syntax extension for JavaScript that looks similar to HTML. It allows developers to write HTML structures in the same file as their JavaScript code, which can make the code easier to understand and debug.JSX is not understand by the browser we need a transpiler like babel tool to convert into pure javascript code,then only the broswer understand the javascript code for execution.

One-Way Data Binding:

React follows unidirectional data flow or one-way data binding. This means that the data flows in a single direction, which makes it easier to debug because you can pinpoint the state and where it is being updated or modified.

Disadvantages of ReactJS:

Rapid Pace of Development:

The React ecosystem is constantly evolving, which can be both a positive and a negative. While it means improvements and updates are frequent, it also means that developers need to continually learn new ways of doing things, which can be challenging to keep up with.

SEO Challenges:

Single-page applications (SPAs) built with React facing challenges with search engine optimization (SEO) because they heavily depends on client-side rendering. even though we have server-side rendering (SSR) with frameworks like Next.js,but it adds complexity to the project.

2.How the virtual dom works explain? akvdom

When a React application is first rendered, React creates a virtual representation of the actual DOM. This virtual DOM is a lightweight copy of the real DOM and is stored in memory.
When the state or props of a component change, React updates the virtual DOM rather than the real DOM. This is because updating the real DOM is costly in terms of performance.
React compares the new virtual DOM with the previous virtual DOM. This process is called "diffing." The difference between the two virtual DOMs is calculated to determine which parts of the actual DOM need to be updated.
React identifies the minimal set of changes required and updates the real DOM accordingly. This approach minimizes the number of direct manipulations to the real DOM, improving performance.
When a component’s state or props change, React calls the render method to create a new virtual DOM tree.
React uses an algorithm to compare the new virtual DOM tree with the previous one. The algorithm focuses on two main principles:
Element Type Comparison: If the elements have different types (e.g., vs), React tears down the entire subtree and builds it again from scratch.
Keyed Elements: For lists of elements, React relies on keys to identify which items have changed, been added, or removed. Keys should be stable, unique, and consistent across renders.
The diffing process generates a list of changes or "patches" that need to be applied to the real DOM. T
React batches multiple changes together and applies them in a single update. This reduces the number of times the real DOM is accessed and modified, which can significantly improve performance.
Finally, React applies the calculated patches to the real DOM, updating only the parts that need to be changed. This ensures that the real DOM is always in sync with the virtual DOM.

3.how to pass data from parent to child component and from child component to parent component? akptoc

Passing Data from Parent to Child Component
To pass data from a parent component to a child component, we can use props. Props are short for properties and allow you to pass data from one component to another.

Passing Data from Child to Parent Component
To pass data from a child component to a parent component, we can use callback functions. The parent component have a function and passes it to the child as a prop. The child component then calls this function, normally with some data as an argument.

4.explain how to achieve life cycle methods in functional components? aklifec

useEffect is used for side effects, similar to componentDidMount, componentDidUpdate, and componentWillUnmount.
if we pass empty array of depency as a second parameter means it runs the useEffect only once this is the componentDidMount
if we pass some depency value in that empty array,It can run after every render or only when certain dependencies change,this is the componentDidUpdate
Cleanup functions within useEffect handle unmounting or preparing for the next effect.for this after return keyword we can pass clear Function like abortController.abort() function to cancel network requests,or clearTimeout() for invalidating timers like this.

5.Hooks in react? akhook
Hooks are functions that allows us "hook into" React state and lifecycle features from function components.

Rules of Hooks

1.Only Call Hooks at the Top Level:
Do not call Hooks inside loops, conditions, or nested functions. Always use Hooks at the top level of our React function so that they are called in the same order each time a component renders.

2.Only Call Hooks from React Functions:
Do not call Hooks from regular JavaScript functions. Instead, you can
Call Hooks from React function components.
Call Hooks from custom Hooks

1. useState
it is used to Manage local state within a functional component.
useState when we need to handle local component state that gives the re-renders based on the state changes.

2. useReducer
it is used to Manage more complex state logic in a component.
useReducer when we have complex state logic that involves when the next state depends on the previous one. It’s also useful for centralizing state updates in a way similar to Redux.

3. useCallback
it is used to Memoize callback functions to prevent unnecessary re-creations.
useCallback to memoize functions when we pass those as props to optimized child components that depend on reference to prevent unnecessary renders.

4. useMemo
it is used to Memoize the expensive calculations to optimize performance.
useMemo to memoize the result of an expensive calculation, so it is only recomputed when its

dependencies change.

5. useContext
it is used to Access context values in a component.
useContext to share the data or state to the in depth level components and it avoids the props drilling.

props drilling & How to avois it:
props drilling means to share the data from one component to any level of component by passing down the props manually at each and every level of the components and we can avoid props drilling with Context or Redux Library.

6. useRef
it is used to Access and manipulate DOM elements directly, or store mutable values without causing re-renders.
useRef to hold a reference to a DOM element or to persist values across renders without causing re-renders and also we can use for managing focus, accessing values from the input field.

7.useEffect
useEffect is used to Perform side effects in function components, like data fetching, subscriptions, or changing the DOM manipulations like changing the text content or styles.

8. useLayoutEffect
useLayoutEffect is used to Perform side effects synchronously after all DOM mutations.
useLayoutEffect for synchronously reading layout from the DOM and synchronously re-rendering. It runs after React has updated the DOM but before the browser has painted.

6.Difference between useState and useRef? akusr

useState:
in the useState Changing the state with the setter function gives the component re-render.
useState won't useful for accessing or manipulating DOM elements.
it is used when we need to maintain state that affects rendering.

useRef:
Changing the current property of the ref doesn't notify the changes don't cause the component re-render.
UseState useful when we need to keep a mutable value that persists across renders or need to directly manipulate DOM elements.
useful for accessing and manipulate DOM elements directly.

7.Difference between useEffect and useLayoutEffect? akuele

useEffect:
useEffect Runs asynchronously after the render and DOM update.
useEffect is used for data fetching, event listeners, and updating state based on props.

useLayoutEffect:
useLayoutEffect Runs synchronously after the render but before the browser layout and drawing.
useLayoutEffect is used for measuring and synchronously updating the DOM, avoiding flickering or layout shifts.

Avoiding flickering Means:
Avoiding flickering or layout shifts in web development isthe preventing visual changes that can negatively

impact the user experience. it happens when elements on the page suddenly change their size, position, or visibility, causing the layout to "jump" or "shift.

8.Difference between class based components(stateful) and functional based components(stateless)? akcf

class based components(stateful):

in class components we are Using this.state and this.setState for managing state.
in class components we can Use lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount.
in class components we have more boilerplate code like constructor, binding methods, this keyword.
in class components we Must use this keyword to access props, state, and methods.

functional based components(stateless):

in functional components we can Use hooks like useState for managing state.
in functional components we can Use useEffect hook to handle side effects and lifecycle methods.
in functional components we have Less boilerplate, simpler syntax, easier to read and write.
in functional components we dont’t have this keyword; directly use variables and functions.

9.Difference between controlled and uncontrolled components in react exaplian? akcunc

Controlled Components:

in the Controlled Components Form data is handled by React state.for example State is managed by React using useState or useReducer.
in the Controlled Components form input values are explicitly handled event handlers like onChange events.
in the Controlled Components Easier to debug and test because the state is managed by React.
in the Controlled Components Easier to validate and transform input values in react as they are directly managed by React state.

Uncontrolled Components:

in the Uncontrolled Components Form data is handled by the DOM itself. React does not directly manage the state of the input elements.
in the Uncontrolled Components No need for event handlers to manage input values.we can Access the form values using refs.
in the Uncontrolled Components Less boilerplate code, simpler setup but its harder to manage for complex forms.
in the Uncontrolled Components More challenging to perform real-time validation or transformation as values are not immediately accessible in React state.

10.how to handle errors in react hooks and explain? akerb

we can use the ErrorBoundary component to catch errors in the rendering tree and display a fallback UI. React provides an ErrorBoundary component that catches JavaScript errors anywhere in its child components. we can create an ErrorBoundary component to wrap around components that we want to catch errors for.
error boundaries include methods like static getDerivedStateFromError and componentDidCatch. These methods are part of the React class component lifecycle and are used within error boundary components to catch and handle errors that occur in their child components.

11.higher order components? akhoc

Higher-Order Components (HOCs) are components in React it allows us to reuse component logic. Higher-Order Component is a function it takes a component and returns a new component with additional props or functionality.

use cases:
in my e commerce application i have different pages accessible only to authenticated users, like user profile or order history page in this scenorios i have used.
Fetching data from an API and sharing the logic across multiple components is a common use case.

12.Custom Hooks? akch
Custom hooks allow us to reuse the stateful logic in a way that can be easily shared across mutliple components.

Use Cases:
we have loading spinners, themes like beautifications that we can share across multiple components.

13.what are the optimization techniques in react? akots

1. Code Splitting and Lazy Loading

in my e-commerce application, split the code for different pages (like product details, checkout, and user profile) to reduce the initial load time.using code splitting and lazy loading.

2. Memoization

Prevent re-rendering of components unless their props change. This is useful for product listings, where each product card component should only re-render if the product data changes using react.memo.

3. useCallback and useMemo

Optimizing the performance of a product filter or search functionality to avoid unnecessary re-renders using useCallback and useMemo.

4.React Query

using React Query Efficient data fetching and caching in my e-commerce app product list, where we want to minimize the number of network requests and provide a seamless user experience.

5. Virtualization

using react window Efficiently render large lists of products using a virtualized list to improve performance.

6. Use of Web Workers

Using Web Workerswe can Perform heavy calculations or data processing without blocking the main thread, like processing a large set of product data or complex filtering logic.

14.How to avoid rerendeing in class components and functional components? akrercf

in class components:

1. PureComponent

Using React.PureComponent instead of React.Component. PureComponent implements shouldComponentUpdate with a shallow prop and state comparison.and it will avoid the unnceccessary rerenders.

2. shouldComponentUpdate
Override the shouldComponentUpdate lifecycle method to prevent re-renders if the component's props and state haven't changed.

in functional components

1. React.memo
Wrap functional components with React.memo to prevent re-renders if props haven't changed (shallow comparison).

2. useCallback
Using useCallback to memoize callback functions, preventing re-creation of functions on every render.

3. useMemo
Using useMemo to memoize expensive computations and avoid recalculating on every render.

15.Difference between React.Memo and useMemo? akrmum

React.memo and useMemo are both optimization techniques in React, but they used for different purposes.

React.memo

React.memo is a higher-order component that memoizes the rendered output of a functional component. It prevents functional component from re-rendering if its props have not changed, optimizing performance by avoiding unnecessary re-renders.
React.memo Uses a shallow comparison of props by default to determine if a re-render is necessary.

When to Use in an E-commerce Application(React.memo)
Product Item Component :

Using React.memo for individual product components to avoid re-renders when product details do not change.
Example: ProductItem component that displays product details and an add-to-cart button.

useMemo

useMemo is a hook that memoizes the result of a computation or function.
It returns a memoized value, recalculating it only when the dependencies change.
Uses dependency array to determine when to recompute the memoized value.

When to Use in an E-commerce Application(useMemo)
Filtered and Sorted Product List :

Using useMemo for filtering and sorting products based on user input to prevent recomputation on every render.
Example: ProductList component that filters and sorts a list of products based on search term and sort order.

useCallback:

useCallback hook in is used to memoize functions.
It returns a memoized version of the callback function that only changes if one of the dependencies has changed. This is useful for optimizing performance, especially in scenarios where passing functions as props to child components could cause unnecessary re-renders.

use cases in my e-commerce application:

Handling User Authentication:
In my e-commerce application,i used useCallback to memoize the function for adding items to the cart to prevent unnecessary re-renders of the product list component.
When managing the shopping cart, we optimized the quantity update functionality using useCallback to ensure that only the necessary components re-rendered.
used useCallback to memoize the login function to avoid re-creating the function on every render, improving the performance of the authentication component.

16.react router? akrr

React Router is a library for managing routing in React applications. It allows us to create navigional paths in our app, allows us to switch between different views or components without reloading the page.

React Router has several components to define and manage routes like BrowserRouter, Routes and Route and Link

BrowserRouter:
This is the main wrapper uses HTML5 history API (pushState, replaceState, and the popstate event) to keep our UI in sync with the URL.

Routes and Route:
Components used to define individual routes in your application.

Link:
Component used to create navigation links.

and also React Router provides shooks to work with routing within functional components.

useHistory:
useHistory Allows us to access to the history instance used for navigation.

useLocation:
useLocation Returns the current location object.

useParams:
useParams Returns the parameters of the current route.

useRouteMatch:
useRouteMatch Returns match data about the current route.

17.state and props? aksp

state:

State is an object used to maintain data that in a component.
it is a local data storage.

State can be changed within the component.
State is managed within the component and is used to control the component's rendering.
we can update the state using setState method,
state is mutable.
When state changes, the component re-renders to reflect those changes in the UI.

Props
Props (short for "properties") are used to pass data from a parent component to a child component.
Props are read-only.
Props are immutable.
we cann't change props values
Props are used to pass external data into a component, making it flexible and reusable.

18.can we directly update the state and why? what happens if we directlu update? akds

In React, we should not directly update the state of a component. Instead,we should always use the setState method in class components or the state updater function returned by useState in functional components.

Why Not Directly Update State?

1.No Re-rendering:

Directly mutating the state does not cause a re-render. React depends on the setState method or the state updater function to know when to re-render the component. Without this, the component will not update to reflect the new state chamges in the UI.

2.Debugging:

React DevTools and other debugging tools depends on proper state updates. Direct mutations make debugging much harder, as the tools won't find the state changes correctly.

19.is setState asynchronus and explain how?(Note:same question and answer for useState) aksa

In React, setState is asynchronous. when we call setState, React does not immediately update the state and re-render the component. Instead, React schedules an update and may batch multiple setState calls together for performance reasons.

Why setState is Asynchronous

Batching Updates:

React batches multiple state updates for better performance. it minimizes the number of re-renders and DOM manipulations, making your application more efficient.

Reactivity:

By making state updates asynchronous, React assure that the component's state is always consistent with the rendered output.

Event Handling:

In event handlers, multiple setState calls are batched together and applied at once, reducing the number of re-renders.

20.what are forward Refs and use cases? akfr

Forward refs (references) in React allow us to pass refs through components to access DOM nodes or React elements of child components. it is useful when we need to interact with a child component's DOM element directly from a parent component.

Use cases:

Useful for accessing and manipulating DOM elements in child components.
Great for higher-order components that need to attach refs to the wrapped components.
Helps when integrating with third-party libraries that require direct DOM access.

Real time use case

In my e-commerce application, i want to focus an input field in a child component from a parent component, like when adding a new item to a shopping cart.

Explanation
Child Component (AddItemInput):

Child component accepts a ref and forwarding it to the input element using React.forwardRef.

Parent Component (ShoppingCart):

A ref (inputRef) is created using useRef.
The AddItemInput component is rendered with the ref attached.
A button is provided to trigger focusInput, which calls inputRef.current.focus() to focus the input element.

21.synthetic events in react? aksy

Synthetic events are the cross-browser wrapper around the browser's native events in React. They will provide a consistent interface for handling events across all browsers.In my e-commerce applications, Synthetic events were used for managing user interactions, form handling and button clicks to implementing advanced features like drag-and-drop sorting.and also in my e-commerce application, handling form submissions is a common task. Synthetic events were used to manage form data and validation.synthetic events are used to handle changes to the form fields and to manage form submission.

Features of Synthetic Events

Synthetic events normalize the differences between browsers,it will provide a consistent API.
React reuses event objects through a process like event pooling. event object might be reused and recycled after the event handler has been called.

22.Difference between shadom dom and virtual dom? akshad

Shadow DOM:
Shadow DOM is all about encapsulation and isolation of components, useful for creating reusable and encapsulated UI elements.
used Shadow DOM when we need to create isolated and reusable components with encapsulated styles and behavior, especially for third-party integrations or custom UI elements.

Real-Time Scenario(use case) in my E-commerce App:

Custom Checkout Widget:When we were building a custom checkout widget that needed to be embedded

on different pages of our e-commerce app, we used the Shadow DOM to encapsulate its styles and scripts. This ensured that the checkout widget's styles and functionality did not interfere with the rest of the app.

Virtual DOM
Virtual DOM is all about efficient rendering and updates, optimizing performance by minimizing actual DOM manipulations.
Used Virtual DOM for efficient UI updates and rendering optimizations, especially when work with the dynamic data and frequent UI changes.

Real-Time Scenario(use case) in my E-commerce App:

Product Catalog Updates: In my e-commerce app, when the product catalog changes (e.g., new products added, prices updated), React's Virtual DOM can efficiently update the UI without re-rendering the entire page. This improves performance and provides a smoother user experience.

23.Strict Mode? aksmode

Strict Mode is a feature helps us to identify issues in React application.Strict Mode is useful during development and does not affect the production build.
Strict Mode in React is a tool for highlighting problems in the application. It is activated by wrapping a part of our application with the <StrictMode> component.
it helps us to write secure javascript code.it will avoid the accidental errors.It helps us to detect unexpected side effects by double-invoking certain lifecycle methods and hooks.

24.React Fragments? akfrag

React Fragments allow us to group a list of children without adding extra nodes to the DOM. This is useful when we need to return multiple elements from a component without wrapping them in an additional div or any other HTML element. It helps to keep the DOM clean and avoid unnecessary wrapper elements.

Benefits of Fragments:
Cleaner DOM:
No extra nodes added, keeping the DOM structure simple and clean.

Performance:
Reduces the number of DOM nodes, potentially improving rendering performance.

Styling:
Avoids unnecessary wrappers that might interfere with styling.

25.React Portals? akport

React Portals provide a way to render children into a DOM node that exists outside the hierarchy of the parent component. This is useful for scenarios where we need to break out of the usual parent-child hierarchy, like modals, tooltips, and dropdowns.

Benefits:

Event Bubbling:
Events still propagate correctly from the portal to the parent component, maintaining the event system's consistency.

Isolation:
Useful for isolating complex components like modals or tooltips that need to be visually separate from the

rest of the app.

26.Protected Routes,authentication and authorization in jwt? akjwt

implementing authentication and authorization mechanisms to ensure that users can only access the resources JWT (JSON Web Token) is useful for securing routes and API endpoints.

steps:
Token validation happens in the auth middleware on the server side.
Client Login Request:

The client sends a login request with username and password to the server.
Server Generates Token:

The server verifies the credentials. If valid, it generates a JWT using jwt.sign and sends it back to the client.
Client Stores Token:

The client stores the token (typically in localStorage or sessionStorage).
Client Accesses Protected Route:

When the client tries to access a protected route, it sends the stored token in the Authorization header.
Server Validates Token:

The server middleware extracts the token from the Authorization header.
The middleware verifies the token using jwt.verify.
If the token is valid, it allows the request to proceed. If not, it returns an error.

27.difference between Authentication and authorization? akaa

Authentication and authorization are two concepts in security, especially in web apps.

Authentication
Authentication is the process of verifying the identity of a user. It ensures that the user trying to access the system.
involves the user providing credentials, such as a username and password.
The system checks these credentials against a database. If they match, the user is considered authenticated.
Once authenticated, the system may generate a token (e.g., JWT) that the user can use for subsequent requests.

Authorization
Authorization is the process of determining whether an authenticated user has the necessary permissions to access a particular resource or perform some specific action. It controls what the user can do and what parts of the application they can access.
Users are assigned roles, and each role has specific permissions.
The system checks if the authenticated user’s role grants them access to the requested resource or action.
Implemented on both the client and server sides to access control.

28.What is Redux? akred
Redux is a predictable state container for JavaScript applications. it is a state management library and helps us to manage the state of our application in a consistent way, making it easier to develop, test, and maintain.

redux components
Action: An action is an object that describes what happened. It has a type property and have additional data like payload.

Reducer: A reducer is a pure function that takes the current state and an action as arguments and returns a new state. It determines how the state changes in response to an action.

Store: The store holds the whole state of the application. It provides methods to access the state, dispatch actions, and register listeners.

Dispatch: The dispatch function sends actions to the store. This is how the state is updated.

Selector: A selector is a function that selects a piece of state from the store. In React, this is often done using the useSelector hook.

Middleware (redux-thunk): Middleware like redux-thunk allows you to write action creators that return a function instead of an action. This is useful for handling asynchronous logic like API calls.

Wrap our React application with the Provider component from react-redux to make the store available to all components.

Use the useSelector hook to access the state and the useDispatch hook to dispatch actions in your component.

29.what is redux and advantages over the context api? akcont

Both Redux and the Context API are tools for managing state in React applications.

Context API:

Best for smaller, simpler applications.
Can become less manageable as the application’s complexity grows.
Suitable for simpler state management needs.
Lack of built-in middleware support, making complex asynchronous logic harder to manage.
Lack of advanced debugging tools. Debugging complex state interactions more challenging in this.
May suffer from performance issues due to unnecessary re-renders when state updates are not managed properly.

Redux:

Better suited for large-scale applications with complex state interactions.
it can handle multiple interconnected states and actions efficiently in large-scale applications.
Provides a single source of truth, making state management more predictable and easier to debug.
Centralized state is used in my e-commerce application where data like user profiles, shopping carts, and order histories need to be consistently accessible across various components.
Middleware like redux-thunk or redux-saga allows handling asynchronous logic (e.g., API calls) more efficiently.
redux thunk Middleware used in my e-commerce application for managing side effects like fetching product data, processing payments, and handling user authentication.
Integration with Redux DevTools provides powerful debugging capabilities, including time-travel debugging and state inspection.
Debugging complex state changes in my e-commerce application (e.g., user interactions, multi-step checkout processes) became easier.
Performance optimizations were important in my e-commerce application where rendering performance could impact user experience, especially during dynamic updates like adding items to a cart.

30.Redux thunk middleware? akmid

Redux Thunk is a middleware that allows us to write action creators that return a function instead of an action. This is useful for handling asynchronous operations in Redux, such as API calls,we can conditionally dispatch actions based on the current state or results of asynchronous operations.

How Redux Thunk Works
Action Creators: action creators return an action object. With Thunk, they can return a function that takes dispatch and getState as arguments.
Middleware: Redux Thunk middleware intercepts actions before they reach the reducer. If the action is a function, Thunk invokes it with dispatch and getState as arguments.
Async Logic: Inside the Thunk function, we can perform asynchronous operations (e.g., API calls), dispatch other actions based on the results of those operations, and access the current state.

Use Cases for Redux Thunk in my E-Commerce Applications
Fetching Product Data: we could fetch product listings from an API and manage loading and error states.
User Authentication: we handled asynchronous user login, signup, and logout processes.
Cart Management: we added items to the cart, removed items, and updated quantities with server-side synchronization.
Order Processing: we submitted orders and handled order confirmation and status updates.
Fetching User Data: we retrieved user profile information, order history, and preferences