1. Tell me about yourself (shortened):

I’m a Senior Full Stack Engineer with 12+ years of experience building .com websites using HTML5, CSS3, JavaScript, React.js, and Redux.

For the last 2 years at United Airlines, I’ve been on the React migration team converting .NET pages to React using our in-house ATMOS library. I delivered several security and account management features like Forgot Password, MileagePlus sign-in, Miles-Pooling, TSA Precheck, Known Traveler Number (KTN), and WCAG accessibility updates. Redux-Saga handled our async flows for data fetching and API calls.

Key initiatives included Miles-Pooling (rewards sharing), TSA Precheck integration, Under18 account restrictions, and enhanced account security.

Previously:

* **Visa Inc.** – Migrated MBDA modules (Application Mgmt, Portfolio Mgmt, Billing, Analytics) for Wells Fargo, Bank of America.
* **Capital Group** – Built Highcharts data visualizations in React, integrated with AEM, and created Creative Workbench for article publishing.
* **Cerner** – Developed medical examination forms.
* **Office Depot** – Built Black Friday reporting tools.
* **Satinos Technologies** – Developed a tax portal and school website for Vignan Schools.

2. **Core React Hooks**

* **useState, useEffect** – For state management and side effects (data fetching, DOM updates).
* **useRef** – For accessing DOM elements and storing mutable values without re-rendering.
* **useContext** – For global state and theme/config sharing without prop drilling.
* **useMemo, useCallback** – For performance optimization, memoizing expensive calculations and stable callbacks.

**Custom Hooks**

* Built **custom hooks** to handle reusable logic like API calls with error/loading states, form validations, and managing authentication flows.

**Advanced Scenarios**

* Combined hooks with **Redux-Saga** for async workflows like fetching user profiles, Miles-Pooling transactions, TSA Precheck data.
* Used hooks to trigger accessibility improvements (e.g., focusing error elements for WCAG compliance).
* Integrated hooks with **Highcharts** for data visualizations at Capital Group.

3. simplest JavaScript code for sum(3,4,5):

function sum(...nums) {

return nums.reduce((total, num) => total + num, 0);

}

console.log(sum(3, 4, 5)); // Output: 12

4. custom hook for sorting: A custom hook for sorting is a reusable function in React that uses hooks like **useState** and **useMemo** to manage sorting logic (key, order) and return the sorted data plus functions to control sorting. It keeps components clean and the logic reusable.

5. useMemo vs useCallback

* **useMemo**: Caches the result of an expensive calculation so it only recomputes when dependencies change. Example: sorting a big list or computing derived data.
* **useCallback**: Caches the function itself so React doesn’t create a new function on every render. Example: passing a stable callback to child components to prevent unnecessary re-renders.

Both help with **performance optimization** by reducing re-renders and recomputations.

6. Purpose of useMemo?

**useMemo** → Caches **values** to avoid recalculating expensive results on every render.

const sorted = useMemo(() => sortData(data), [data]);

* **useCallback** → Caches **functions** so they aren’t recreated on every render, preventing child re-renders.
* const handleClick = useCallback(() => doSomething(id), [id]);

Both help with **performance optimization**:

* useMemo → Memoize **results**
* useCallback → Memoize **functions**

**7.**