1. When and why do we use lazy() ?
   We use React.lazy() when we want to **optimize the performance of large React applications** by implementing **code-splitting**. In a large app with many components, bundling everything into a single file can slow down the initial load time. Instead, React.lazy() allows us to **dynamically import** components **only when they're needed**, rather than at startup.

This improves performance by:

- Reducing the size of the initial JavaScript bundle.
- Loading components **on demand**, improving load times and responsiveness.
- Helping the bundler (like Vite, Parcel, or Webpack) split the app into smaller chunks.
- React.lazy() works well with Suspense to display a fallback UI (like a loader) while the lazy component is being loaded.

2. What is Suspense in React?

Suspense is a built-in React component used to **handle the loading state** when you're using features like React.lazy() for **code-splitting**. When a lazily-loaded component is being fetched, it may take time for the browser to download and execute the corresponding JavaScript chunk. During this time, Suspense allows you to **display a fallback UI** (like a loader or placeholder) until the actual component is ready.

3. Why we got this error : A component suspended while responding to synchronous input. This will cause the UI to be replaced with a loading indicator. To fix, updates that suspend should be wrapped with startTransition? How does suspense fix this error?

This error occurs when a **synchronous user action** (like typing or clicking) triggers an update that causes a component to **suspend** (e.g., due to lazy loading or data fetching with Suspense), **without properly deferring the update** using startTransition. React expects updates triggered by user input to be fast. If a component suspends during such an update (e.g., while waiting for a lazy-loaded chunk or async data), React throws this warning because it doesn't want to block user interactions with a loading indicator.

❓ Advantages and Disadvantages of Code Splitting in React

✅ Advantages:

1. **Improved Performance**
   - Reduces the size of the initial JavaScript bundle, leading to faster page loads.
   - Only loads code when it's actually needed (on-demand).
2. **Better User Experience**

   - Users interact with the app faster as only critical components are loaded initially.
   - Suspense allows showing loaders or placeholders during component loading.
3. **Scalability**
   - Makes large applications easier to maintain by organizing code into smaller, logical chunks.
   - Reduces memory usage by loading components as needed.
4. **Optimized Bandwidth Usage**
   - Particularly helpful on slower networks where downloading everything upfront would be costly.

❌ **Disadvantages:**

1. **Complexity in Handling Asynchronous Loading**
   - You need to manage loading states carefully using Suspense and sometimes startTransition.
2. **Potential UI Jank**
   - If not handled properly, users might see spinners or blank screens, especially during frequent navigation.
3. **SEO Challenges (for SSR)**
   - Lazy-loaded components may not be rendered immediately on the server, requiring additional configuration for server-side rendering (SSR).
4. **Chunk Loading Failures**
   - Network issues or outdated chunks can cause loading failures (e.g., "Loading chunk failed" errors).
5. **Debugging Can Be Harder**
   - Lazy-loaded components can complicate the stack traces or behavior during debugging if chunks fail to load correctly.