A- How I mudlarise a large react app:

Leverage Frameworks: If possible,i will opt for frameworks like Next.js. They offer built-in solutions for routing, authorization, theming, and more, saving development time and effort compared to building everything from scratch, will help for SSR also.

App Structure: Organize the application using either a domain-based or feature-based architecture with clear folder architecture for components, utilities, pages, hooks, and other logical groupings.

Reusable Components: i would by developing small, reusable "micro-components" like inputs, labels, etc. This fosters consistency and efficiency across the project.

Conventions: setting team-wide rules for naming, styling, and component reuse. Enforce these rules with tools like ESLint and Prettier to maintain a clean and unified codebase.

TypeScript: While it might initially extend development time, TypeScript significantly improves code maintainability in large projects. It catches potential errors early and provides helpful data typing.

API Performance : Use libraries like React Query to streamline data fetching and caching. This helps avoid misusing hooks like useEffect and enhances performance by reducing redundant API calls.

State Management: i would employ state management solutions like Context or Redux when appropriate. They can boost performance on both frontend and backend by minimizing unnecessary API requests. However, I would use them judiciously to avoid over-complicating your code.

Libraries: i would choose libraries carefully, prioritizing those with active maintenance and support. An over-reliance on libraries can lead to bloated initial load times and potential compatibility issues for non-supported libs.

Security: Large-scale applications often have complex code structures that can introduce vulnerabilities. I will implement tools like Snyk to identify and address security flaws, and SonarQube to monitor code quality. Supplement this with regular code reviews and CI/CD pipelines.

Test: Depending on my team size and project timelines,i will invest in unit, integration, and end-to-end testing. This significantly reduces the risk of introducing errors and ensures a more reliable application.

Documentation: Writing clear code comments and annotations to explain the functional and business logic is important , also maintaining a comprehensive documentation for features to aid onboarding and troubleshooting.

2- Difference between React and Angular :

This debate has no end 😅, Both Technologies are leading the market and Both used in Big projects .

1-Structure: React is library and Angular is a Framework this means that Angular comes with set of tools like CLI , Component Injections , Typescript and more but that leads us to next difference .

2- Learning Curve: React comes with a lower learning curve since react can be built by js without the types complexity that Typescript comes with , moreover angular forces devs to use folder strcuture , and complex component communication , this takes us to next point.

3- component communication: react support data flows in a single direction, from parent components to child components which is easier to use and debug than the two-way binding in Angular, but that also comes with a challenge when performing child to parent action.

4- DOM manuipluation : React uses a virtual DOM that stays in the memory and performs quicker UI updates by comparing the in-memory DOM versions to the old one to identify and deliver only the changed parts. which is more complex to understand than Angular DOM .

5- HTML vs JSX: react uses embedded HTML in a JS file which make it easier to show and manipulate data but this representation is not HTML it’s called JavaScript XML thet later be compiled to HTML , this different than HTML specialty in elements fields, Angular use plain HTML , better for separation of concerns and a familiar syntax for those comfortable with standard HTML.

6 Developer Experience: Personally, I prefer React's flexibility and vast library ecosystem, especially with frameworks like Next.js. I consistently find the right tools for any task. While Angular's structured approach can be beneficial, it has felt limiting at times, making certain tasks slightly more complex .

Choosing between React and Angular is often a tough decision with no easy answer. In my experience with moderate-sized projects, companies tend to prioritize their existing developers' skillsets. If a company has more React developers, they'll likely choose React, and vice versa. Ultimately, the "best" choice depends on the specific project needs and the team's expertise.
I prefer React 😅