**React:(Helps to make complex UI) (Front End) (Not SEO optimized)**

1. **Library:** React is a JavaScript library for building user interfaces (UIs). (you have the architecture control of the dependencies)
2. **User Interface:** React focuses primarily on the UI layer. It helps you create interactive and dynamic web pages.
3. **Rendering:** React can render UI components on the client-side (CSR - Client-Side Rendering) by default. You can also use libraries like ReactDOM Server to render on the server-side (SSR).
4. **Routing:** React doesn't have built-in routing. You typically use third-party libraries like React Router for client-side routing.
5. **API Integration:** React doesn't have built-in server-side capabilities. You need to use other technologies (e.g., Node.js) for server-side logic and API integration.

**Next.js:(React based) (SEO Optimized)**

1. **Framework:** Next.js is a React framework for building complete web applications. (you don’t have the architecture control of the dependencies)
2. **User Interface:** Like React, Next.js can build user interfaces, but it also handles other aspects of web development.
3. **Rendering:** Next.js supports both CSR and SSR out of the box. You can choose how you want to render your pages.
4. **Routing:** Next.js has built-in routing. You can define routes for your pages without relying on external libraries.
5. **API Integration:** Next.js includes server-side capabilities, making it easier to build APIs and handle server-side logic.

**When to Use React**:

* Use React when you want to build a single-page application (SPA) or when you need fine-grained control over the rendering process.
* React is suitable for building UI components within larger web applications.

**When to Use Next.js**:

* Use Next.js when you're building a full-stack web application with both client and server components.
* Choose Next.js for server-side rendering (SSR), search engine optimization (SEO), and better performance out of the box.
* Next.js simplifies routing and provides an integrated development experience.
* **Client-Side Rendering (CSR)**:
  + Initial page is minimal, loaded quickly.
  + Content generated in the browser using JavaScript.
  + Good for single-page apps but may have SEO and initial load issues.
* **Server-Side Rendering (SSR)**:
  + The initial page is fully generated on the server.
  + Content is immediately visible to the user.
  + SEO-friendly and faster initial rendering.

Choose CSR for dynamic single-page apps and SSR for better SEO and faster initial display.

**Automatic Installation**

**Command: -** to get the boiler plat from the framework

* **npx create-next-app@latest ./**
* **npm run dev**