**React Component Patterns & Best Practices (Based on SpermTrackResults)**

**1. Overall Structure**

**Component**

* SpermTrackResults is a **functional React component**.
* It contains both presentation and logic in one file.
* Ends with export default SpermTrackResults; for reuse in your app.

**JSX Layout**

* Uses semantic HTML (e.g., <section>, <table>, <video>) for structure.
* Uses Tailwind CSS utility classes for styling and layout.

**2. React Hooks Used**

**useState**

const [selectedTracker, setSelectedTracker] = useState("ByteTrack");

* Stores UI state (which tracker is currently selected)
* Clicking buttons updates this value and triggers re-rendering

**useRef**

const videoRefs = useRef([]);

* Stores direct references to DOM video elements
* Used for setting playback speed and custom behaviors
* A react hook that gives mutable reference to a DOM element (like <video> or <input>

**useEffect**

useEffect(() => {

videoRefs.current.forEach((video) => {

if (video) video.playbackRate = 0.4;

});

}, []);

* Runs once on mount (empty dependency array [])
* Applies setup logic (e.g., playback rate) to the video elements

**3. Data-Driven Rendering Pattern**

**trackerVideos object**

const trackerVideos = {

ByteTrack: [...],

"BoT-SORT": [...]

};

* Holds video metadata grouped by tracker type
* The UI is driven by this structure instead of hardcoded elements

**Dynamic Rendering with .map()**

{trackerVideos[selectedTracker].map((video, idx) => (...))}

* Automatically loops through and displays the correct videos based on state

**4. UI Structure and Layout**

**Layout Techniques**

* Responsive paddings: px-4 md:px-8 lg:px-12
* Containers: max-w-7xl, w-full, rounded-lg, border
* Typography: text-xl, font-semibold, leading-relaxed
* Layout helpers: grid grid-cols-2, flex flex-col items-center

**Fix for Centering Video + Description**

Change this:

<div key={idx}>

To this:

<div key={idx} className="flex flex-col items-center text-center">

This aligns both video and text in the same centered block.

**5. Suggested Next Steps to Learn**

| **Topic** | **Why It’s Useful** |
| --- | --- |
| useState, useEffect | React core for interactivity & lifecycle |
| useRef | Managing DOM elements directly |
| Component composition | Break big UIs into smaller reusable pieces |
| .map() and keys | Rendering lists of data |
| Tailwind CSS | Fast styling with utility classes |
| Prop passing | Share data across child components |

**6. Future Modularization Ideas**

You could break this into reusable components:

* <PerformanceTable /> — for the detection results table
* <TrackerSelector /> — for the ByteTrack vs BoT-SORT buttons
* <VideoCard /> — for rendering each video with title and description

This makes the code more maintainable and easier to reuse across other pages.

**7. Resources**

* [React Docs](https://reactjs.org/docs/getting-started.html)
* [Tailwind Docs](https://tailwindcss.com/docs/installation)
* [JavaScript .map()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map)
* [React useEffect](https://reactjs.org/docs/hooks-effect.html)

You can keep adding to this document as you explore more React patterns!