Performance Analysis Report – Video Streaming Application

Overview

This performance analysis was conducted to identify bottlenecks in the video streaming application that could degrade the user experience. The focus areas include page load speed, video loading, search responsiveness, and resource usage.

Identified Performance Issues

1. **Slow video load time** – Videos take more than 5 seconds to start, especially on slower networks.
2. **Search lag** – Typing in the search box causes noticeable delay before displaying results.
3. **No lazy loading of thumbnails** – All video thumbnails load at once, affecting initial page render.
4. **Large JavaScript bundle** – Initial load includes all scripts, leading to increased time to interactive (TTI).
5. **Uncompressed media assets** – Images and videos are not optimized or compressed.
6. **No caching strategy** – Static assets are re-fetched on every reload.
7. **Auto-refresh reloads entire page** – On refreshing, new content replaces the current session, which is resource-intensive.
8. **No pagination or infinite scroll** – Loading all videos on the same page impacts DOM rendering.
9. **Redundant API calls** – Same API endpoints are hit multiple times without cache or memoization.
10. **Inefficient DOM updates** – State changes or UI interactions trigger re-renders unnecessarily.

Optimization Recommendations

1. **Implement video preloading** only when the user clicks Play or hovers over the thumbnail.
2. **Use debounce or throttle** techniques on search input to reduce frequent API hits.
3. **Enable lazy loading** for images and video thumbnails using loading="lazy".
4. **Split JavaScript bundles** using dynamic imports and code-splitting with Webpack.
5. **Compress images and videos** using tools like ImageOptim, TinyPNG, or ffmpeg.
6. **Add caching headers** and use service workers for storing static content.

7. **Avoid full-page reloads** on refresh; store content in memory or sessionStorage.
8. **Introduce pagination or infinite scroll** to reduce the number of DOM elements rendered initially.
9. **Use memoization or caching** at the API layer to prevent redundant requests.
10. **Use virtual DOM diffing wisely** and avoid unnecessary re-renders by tracking state changes efficiently.

Metrics & Benchmarks (Observed)

Metric	Observed Value	Recommended Target
First Contentful Paint	~3.8 seconds	< 2.5 seconds
Time to Interactive (TTI)	~6.2 seconds	< 5 seconds
Video Initial Load Time	~5–7 seconds	< 3 seconds
Search Response Time	~1.8–2.5 seconds	< 1.5 seconds
Page Load (Homepage)	~4.5 seconds	< 3 seconds
Bundle Size (JS)	~2.4 MB	< 1 MB (split preferred)