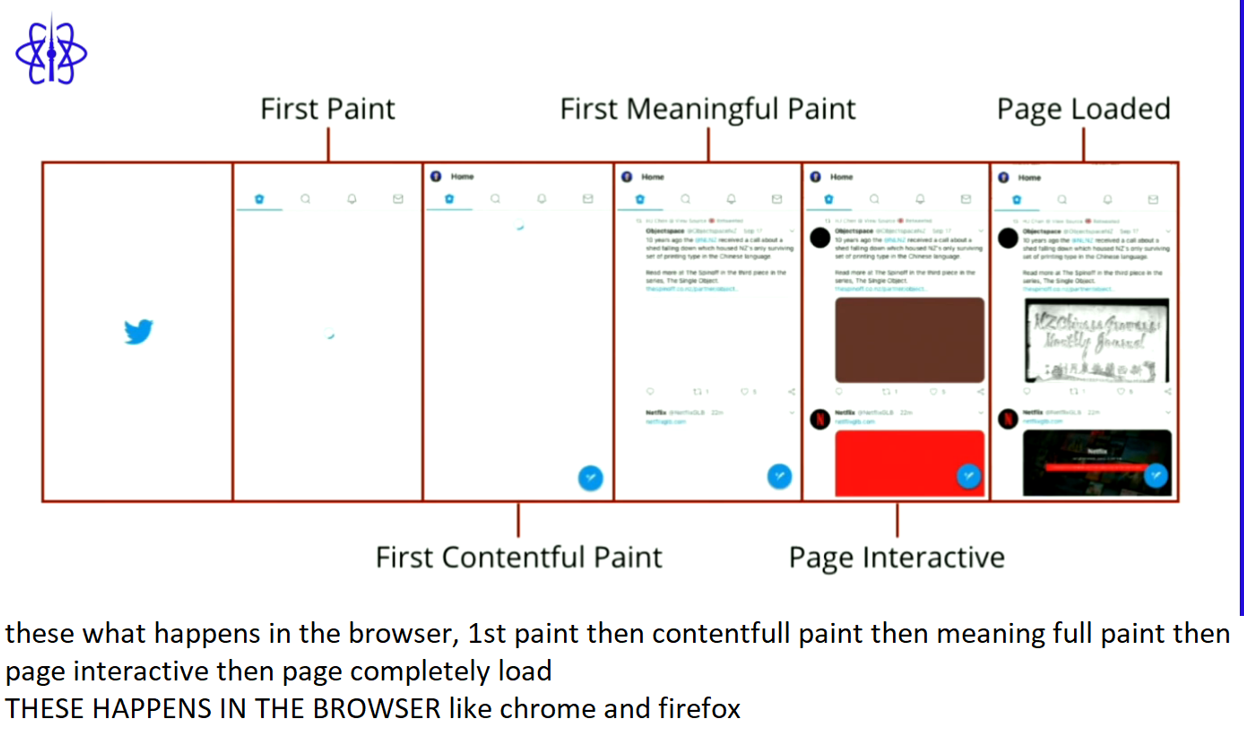
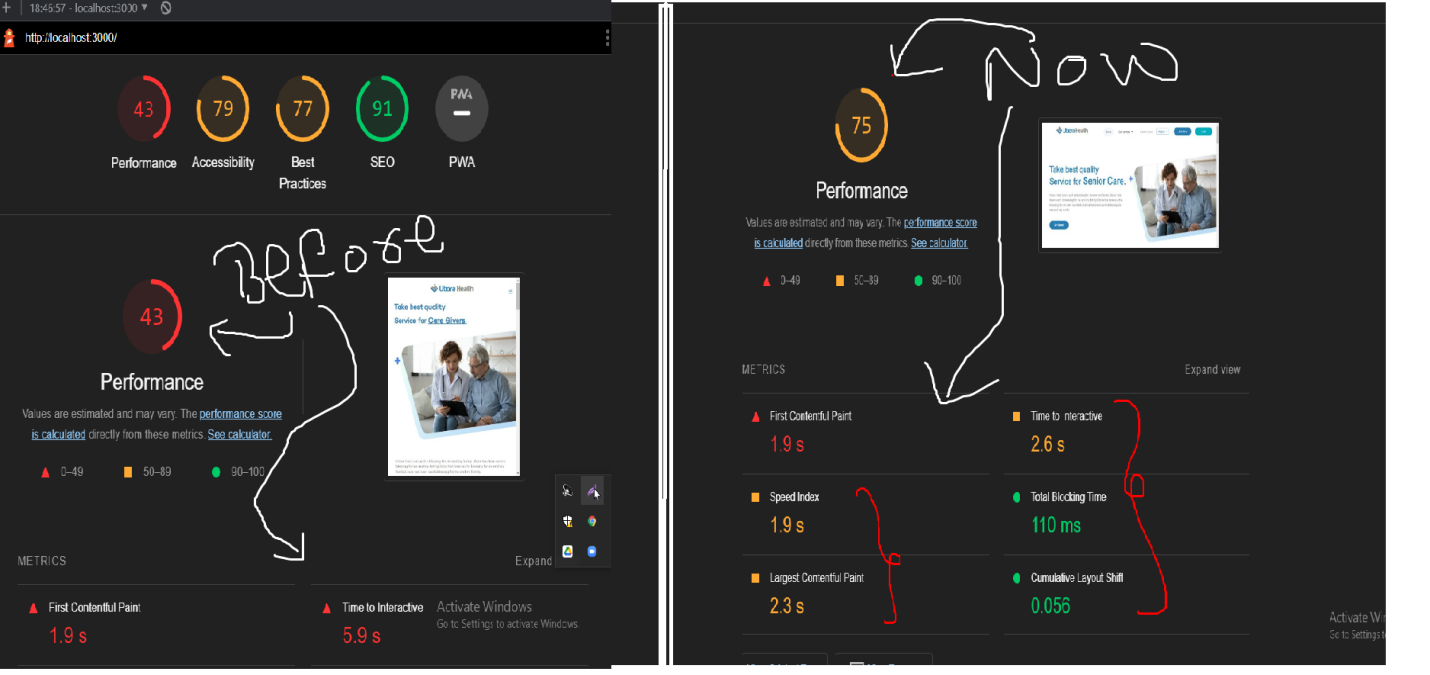
**React Optisamation**

**Parsley compelted(still need to do Research )**

1. we must check, our self is our components is working good or bad, if its not working good in rendering , then just go to that components try ourselves to optimize the code, and take the help of the Google devtools
2. when we use much data in map (loop) or in rendering then its take much time to compute and it take much time to render, So data also play key role in the rendering
3. lots of data=>slow rendering =>bad User experience
4. we must need to use, react memos in our code so that the component will render only when its props changes , not on its parents changes, these help us to render react app fast  
   react memo doesnt solve all the problems but its help in optimsation if u dont like react memo then go to react life cycle methods
5. if ur not good friendly with the React classes components then go to the React useCallback hooks these function will be exguate only when its defentity changes   
   Please dont compare useCallback() hook with useEffect(), there is must difference in both of them ( see offical react doc for more info)
6. always use destructing, becoz compoenet will render again when attritubes changes, not when all props Object changes, it help in react optimasation
7. try to avoid using react lifecycle hooks which run on each render.
8. try reducing any scripts runing on every render.
9. use javascript profiling tool, to see which javascript methods took most time.
10. Try using redux with react, to manage state
11. When using redux use an appropriate batching stategy. You can also use batch middleware in redux.
12. Also, similarly, in react try to do events in batched manner so as to reduce amount of time spent in react's renderings and diffing algorithms.
13. Try clubing setStates in react or actions in redux to reduce react scripting time.
14. Always use appropriate throttling/debouncing techniques while implementing input controls, to get immediate response. You can also use uncontrolled components to have immediate response, if your business logic allows. Idea is to not to run any javascript, when user is typing or interacting with your page in any way, else he would notice the jank in devices particularly bad in computing power.
15. Your action handlers should not be lengthy. If lengthy, try to do them in chunks, asynchronously, with redux-actions or with just promises.
16. ****
17. css and js and api consume much time and stop the browser from the rendering   
    NOTE : when the broweser see the css and js files , the browser main thread will stop rending and just focus on the prasing the css and javascript file so these is the main reason why we see the blank screen in the browser
18. if we are using the node.js or mongodb as a backend, theyy we have high changes that our pages load faster, let see how, if we use renderToString method in backened for rending the APP, the how parsing of the css and javscript and images and html and eveything will be happened in the backend itself only(yeah i accept backend meda baga load paduthadi but mana website oka range lo load aethadi) so that user can get the page instanteous and can interactive and evertyting will be happened in quick, if we can implemenet then go for these approach, no need of any other tools or methods or 3rd party things to optismeie the react APP
19. when we use async or defer keywords in our script then browser en-counter the script its doesnt parse the script code but keep them in queue so that they are exguate at last when html and css are completely pasred the javscript code will be parsed
20. these is the major issues, as we are developer we dont know, that is when when we use and cdn or extranal font-family that is linked to our project NOTE: browser will parse the font at last when js ,css and html are parsed then font turns comes, OK, now lets comes to point when the browser parse the html, css,js file its ready for rendering in dom but broswer stop it from the render becoz still the font are not came and not added so now broswer will fetch the font and then added to our code then its render KEY-POINT : use the internal script for fonts, dont depend on the cdn or extrenal links
21. font-display: swap / block / fallback to overcome these, use swap keyword, that will display a browser native font untill the cdn font cames
22. take the help of the 3rd party site or libray so that hole code dont load, which is much time taking process so just load the code which comes under 100vh becoz user see 1st 100vh lo unna content a kada
23. just load the css which comes under 100vh so that all other css code will be load when user demands
24. almost all usages, https 1 only, but when we switch to http2 then our code site will load faster   
    1) http1 doesnt -> its dont parse any code, it just hand over the code to the broswer then browser will parse the code   
    2) if we use htpp2 then http2 will prase the code for us and give it to browser the browser will just render the code
25. lighthouse is fantastic tools which is inbuilt in the chrome, we can use these tool for testing our website speed or muvh more, its give much information like use alt attritbue in image if we didnt used it, so-on, there are so many feature , for more info, refer to google
26. 1) Server Render your Application!   
     2) Prioritize Above the Fold Content Defer JavaScript Execution

3)Optimize Images

4)Use resource hints Update your stack (http/2)

1. parse the React app code at server and then send that code (static code) to browser so that our website iwll load as soon as possible
2. as the Ul is loaded user may think that DOM is ready for use, and user may start interacting with website , actuallty here , still now js and script file are not pasese, OK broswer is multi-thread but when comes to js file parsing , browser runs in a single loop so user may not interact which is major drawback of browser, tell see in upcoming photos how to solve these issues
3. The first step was to split each hydration into its own task using a well-placed   
   setTimeoutand a dynamic import with Webpack's magic comment webpackMode: "eager". Then the use of IntersectionObserver and dynamic import allowed us to differ the code download and the hydration of the footer right before the user scrolls it into view.   
   By using the webpackMode plugin or extension in webpack.config.js file we can achieve these behiour in the browser With these two steps, we were able to reduce our Total Blocking Time by over 40%.
4. React uses several clever techniques to minimize the number of costly DOM operations required to update the UI   
   If you're benchmarking or experiencing performance problems in your React apps, make sure you're testing with the minified production build. If your application renders long lists of data (hundreds or thousands of rows), we recommend using a technique known as "windowing”. This technique only renders a small subset of your rows at any given time, and can dramatically reduce the time it takes to re-render the components as well as the number of DOM nodes created. react-window and react-virtualized are popular windowing libraries
5. Load your components on demand with code-splitting and dynamic imports   
   Batch your DOM operations / dispatch if using redux   
    Use PureComponents   
   HOC's
6. 1. Using Immutable Data Structures   
   2. Function/Stateless Components and React.PureComponent   
   3. Multiple Chunk Files third-party library code from your application code by taking advantage of CommonsChunkPlugin for webpack   
   4.Using Production Mode Flag in Webpack Doing this will limit optimizations, such as minification or removing development-only code, to libraries   
   5. Dependency optimization loadash. Let's say you are only using 20 of the 100+ methods, then having all the extra methods in your final bundle is not optimal. So for this, you can use lodash-webpack-plugin to remove unused functions,   
   6. Use React.Fragments to Avoid Additional HTML Element Wrappers or use <>   
   7. Avoid Inline Function Definition in the Render Function. this means React Code lo render() method lo, inline function definatation we should write becoz, when state upadted then remder function will exguate then for each exguation of render function a new inline function refernce is created in the Heap which lead to so many garbage collection ( previous function refernce will be lost and now React will remove that refernece from the heap)
7. Avoid using Index as Key for map even if u use, ur may get sideEffect in future
8. In React, when we render a page server side, we get the benefit of a better First Contentful Paint where the user does not have to wait for the Javascript to boot up and render the page. But, we still need to
9. Use React.Suspense and React.Lazy for Lazy Loading Components
10. Use React. Fragment to Avoid Adding Extra Nodes to the DOM
11. Reselect library encapsulates the Redux state and checks the fields of the state and tells React when to render or not if the fields haven't changed.
12. Web worker JS code runs on a single thread. Running a long process on the same thread will seriously affect the UI-rendering code, so the best bet is to move the process to another thread. This is done by Web workers. They are the gateway where we can create a thread and run it parallel to the main thread without hampering the UI-flow
13. Keeping your app performant Virtualize Long Lists Lazy loading Cache data Minimize everything Handle errors
14. ****
15. By the help of those above tech we have achieved the user app performance landing page frpm 43% to above 70%