**CSR VS SSR:**

Sveltekit project uses SSR (server side rendering) by default for your .svelte files. This occurs for the initial page loading the html page is pre rendered at the server side and sent to the browser without the java script then it contains with CSR (client server rendering). The application continues the client side by performing hydration on the html element. Hydration means adding interactivity to the html page.

Inside my code I ensured to use the CSR strategy during the users usage to be on the client side as our app requires high user interactivity and requires dynamic updates of the markers location and stores and retrieves the pins location from the IndexedDB database. This causes very smooth real time map interactions without extra server delays. And also this helps for non-blocking updates to the main UI thread

We used the SSR at the beginning to gain advantage of the fast initial page load then continued with CSR for more smooth and light weight user experience.

**Local storage VS IndexedDB**

Both are local storage in the client side at the browser. This allows web application to persist data between page reloads.

Local storage: is used for simple and small amount of data and less complex. Its also synchronous can block the main thread when reading and writing data.

IndexedDB: stores larger data sets with more complex data structures. Operates asynchronously improving the performance of the app.

Because my data is more structured and contains latitude and longitude information and requires frequent updates and asynchronous operations, I used IndexedDB.

**Asynchronous Updates for handling asynchronous operations**