NAVYA NARAYAN PANICKER

WEEK 2

DAY 4: 03/10/25 [FRIDAY]

TASK : Polish responsive layout; deploy on GitHub Pages; write README

QUESTIONS/REFLECTIONS:

1. Describe how client-side routing works (history API or hash routing).
   Client -side routing basically means that the views and url gets updated without reloading the entire webpage.

   Hash routing:
   Hash routing uses the fragment identifier of a URL(the part after the # symbol) to track the navigation state.
   When the hash changes the js reads it in the hashchange event, reads the location and updates the UI accordingly.

   History API:
   The history API give more clear URL and it is a better way of routing.
   It lets apps use pushState() and replaceState() to change the browser URL path without a reload
   The app waits for thet stae change event and once that's heard it will change the browsers url accordingly and renders the right content.

2. What happens when you navigate: how React Router matches route and renders components.
   the library checks the current URL path against the defined route paths in the routing configuration
   it uses matching algo to find the matching url.
   After the match is found React Router determines which components correspond to the matched route(s) and renders them.
   This process happens without a full page reload therefore it is more fasters and efficient to use.

3. How to pass params or query params; nested routes.
   Route parameters are passed by id in the route path and accessed in components via hooks such useParams() in reacts routers.
   Query parameters are simply appended to the url and can be accessed using hooks.
   Nested routes, it defines child components inside parent routes, react will match and render both parent and child components.

4. What is the flow: writing to localStorage → reading on app startup?
   When data changes, covert to string using json.stringify() and save to localstorage
   Read the stored string from local storage.
   Pass the stored string back to its original form
   Initialize the app state with the parsed data to restore the previous state.

5. How do you sync state with localStorage safely (e.g. updates, JSON parse/stringify)?
   When data changes, covert to string using json.stringify() and save to localstorage
   Read the stored string from local storage.
   Pass the stored string back to its original form
   Initialize the app state with the parsed data to restore the previous state.

6. What performance / size concerns with storing too much in localStorage?
   - Cause date write failure- most browsers will have 5mb per domain and exceeding will cause this issue.
   - large or frequent reads/writes can stall the UI and degrade user experience
   - ata in localStorage is accessible by any JavaScript running on the domain, posing risk if sensitive information is stored.
   - localStorage only stores strings without indexes, so complex or large datasets can be inefficient to manage