Offline Functionality & PWA Implementation

Section 1: Offline Functionality Implementation

- **State Persistence with Zustand Persist Middleware**

Zustand's **persist** middleware is used to save fetched data in **localStorage**. This ensures data like character lists and details are preserved across sessions, reloads, and remain accessible offline.

- **Local Caching with CharacterInfoRecords**

CharacterInfoRecords uses a **Record<number, Character>** to cache character details by their IDs. This prevents redundant API calls and supports offline detail access.

- **Early Return Optimization for Cached Data**

Before making a network request for a character, the code checks if it already exists in the cache:

eg.

```
if (characterDetails.data[id]?.id) return;
```

- **Graceful Error Handling with errorMsg**

Both **characters** and **characterDetails** include a **errorMsg** property to track errors during data fetches, helping to display fallback UIs or retry prompts when offline.

- **Separate Loading States for List and Detail**

isLoading and **isFetchingNewCharacter** are independent loading states that distinguish between network operations and cached data usage, providing more accurate feedback.

Encapsulation of Fetch Logic

Data-fetching is encapsulated in:

- **fetchCharacters()**
- **fetchCharacterDetail(id)**

– Key Implementation Steps

- Imported dependencies: zustand, persist, API utility, and types.
- Defined interfaces: **CharactersState, CharacterInfoRecords, CharacterStore.**
- Zustand store initialized using **create and wrapped with persist** .
- Configured default state.
- Implemented fetch functions with error/loading state handling.

Section 2: Offline Navigation & PWA Implementation

– PWA Support via Serwist

The app integrates PWA capabilities using **@serwist/next**, allowing previously visited pages to be accessible offline.

Next.js Configuration with withSerwistInit

A custom configuration enables:

- Pre-caching
- Navigation caching

- Reload on reconnect
- Dynamic service worker injection

Custom Service Worker using Serwist

The service worker setup:

- Uses **defaultCache** for runtime caching
- Supports navigation preload
- Provides fallback for offline navigation

Offline Fallback Page

The UI gracefully handles offline scenarios where the route hasn't been cached yet.

Thank you .

Prepared by:
James Mensah