Search Data Flow Pseudo Code for the STIX App

This is a pseudo code lay-out of sorts to think through the data flow for the core element of the STIX Fun Finder application.

The application revolves around finding/searching for data that matches the search criteria.

Search Steps

1. User inputs a Zip Code – the zip Code is saved into the "**Search Criteria**" object

2. An **API Get Call** is generated with the Zip Code from the **Search Criteria**
   -- By default the first API call will be for a radius of 50 miles

3. Compare the API Zip Code Results with the DATABASE ZIP CODES

4. Put all the Matching Data from the COMPARE process into a **DATASET**

5. Format and Output the Results from the Dataset to the Screen

6. User updates / changes the **Search Criteria** – (5 search criteria to refine search results)
   -- **IF** Radius > 50 miles OR new Zip Code provided => New **API Get Call**
   -- Compare the API Zip Code Results with the DATABASE ZIP CODES
   -- Put all the Matching Data from the compare process into a **DATASET**

   -- ELSE …

7. Filter the **Dataset** results according to the **Search Criteria**

8. Format and Output the Results from the Dataset to the Screen

Details of the Search process …

The search process follows 6 steps:

1) Input
2) API Call
3) Data Comparison
4) Result storage
5) Filter results
6) Results Output

1) The user initiates the sequence of actions by specifying Search Criteria. Search Criteria will be saved in a Search Criteria array or object.
   - Initial Search Criteria will contain a user generated Zip Code and a radius of 50 miles (default value).
   - After initial results are returned the user can modify the Search Criteria by choosing from 5 different options on the search options bar.

2) In response to the user input, an API call is generated to satisfy the search request by using the Search Criteria
   - By default, the first API call will be for a radius of 50 miles + the user supplied radius.
   - If the user updates / changes the search Radius to more that 50 miles, or if a new Zip Code is entered, a new API call has to be made for the new search radius.

3) The API Zip Code Results that are returned need to be COMPARED with the Database Zip Code data
   - A filter() function will be needed to compare the Zip Codes between the 2 sources.

4) The matching data from the Comparison between the API results and the Database needs to be placed in a reusable Dataset object.
   - A Dataset object needs to be built that will keep data in matching key-valuer pairs.

5) Filter the Dataset results according to the user Search Criteria
   - In all probability a reduce() function will be needed to extract the correct data from the Dataset – according to the Search Criteria.
   - Filter criteria are …
     - Zip code – for a new zip code area
     - State – to narrow the search to a state
     - Radius – to specify a new radius for a search
     - Type – to specify club, group, school, or event
     - Style – to specify which style of FMA- Aris, Escrima, Kali, or All
   - The results will be stored in a new Array for output.

6) The results that have been extracted from the Dataset need to be presented on-screen for the user
   - If no matching results are found, a message needs to be generated that no results were found, and to please search again.
   - Results will be placed on the page in a grid pattern, listed from closest result to farthest result.