Assignment: Build a Paginated React App using CoinGecko API with Caching

Objective:
Create a React app that fetches cryptocurrency data from the CoinGecko API with pagination functionality. Implement caching to optimize API calls using a caching library of your choice.

Requirements:

1. **Fetch Data from CoinGecko API:**
    - Use the CoinGecko API to fetch a list of cryptocurrencies.
    - API endpoint to use: `/coins/markets` .
    - Parameters:
        - `vs_currency` : Set this to USD.
        - `page` : The page number to implement pagination.
        - `per_page` : Number of results per page (e.g., 10 or 20).
2. **Implement Pagination:**
    - Add pagination controls that allow users to navigate through the pages of results.
    - Ensure the current page number is highlighted or indicated.
    - Only load data for the page the user is currently on.
3. **Cache API Responses:**
    - Implement a caching mechanism to store and retrieve API responses.
    - Cache data for a set amount of time (e.g., 5 minutes) to reduce unnecessary API requests.
    - Utilize a caching library such as **`react-query`** , **`swr`** , **`localforage`** , or **`axios-cache-adapter`** .
4. **Display Cryptocurrency Data:**
    - Display key information for each cryptocurrency:
        - Name
        - Symbol
        - Current price
        - 24-hour percentage change
        - Market cap
    - Provide a clean and responsive UI layout for the list.
5. **Error Handling and Loading States:**

- Display loading spinners or placeholders while data is being fetched.
- Show appropriate error messages if the API request fails.

6. **Bonus (Optional):**
    - Allow users to select different `vs_currency` options (e.g., EUR, GBP).
    - Implement client-side sorting for the results (e.g., sort by market cap, price).
    - Add a search input to filter cryptocurrencies by name.

Steps to Complete:

1. **Set up a new React app:**
    - Initialize a new React project using `create-react-app` or a similar setup.
    - Install necessary dependencies such as Axios for API calls and the caching library of your choice.
2. **Make the API Request:**
    - Create a service to handle API requests to the CoinGecko endpoint.
    - Implement the logic to fetch data for specific pages and handle the pagination state.
3. **Implement Pagination:**
    - Add buttons or controls that allow users to switch between pages.
    - Dynamically update the data based on the page selected.
4. **Set up Caching:**
    - Use a caching library (e.g., `react-query`, `swr`) to cache the API responses and reduce duplicate calls.
    - Ensure that cached data is automatically refreshed or expired based on your configuration.
5. **Test the Application:**
    - Test the app by navigating through different pages and verify that caching is working by checking API call frequency.
    - Test edge cases such as network failures, no data, etc.

Good luck!