1. Imports:

```
import React, { useState, useEffect } from 'react';
```

- **useState:** Hook for managing component state (e.g., player data, filter values).
- **useEffect:** Hook for performing side effects (e.g., fetching data from an API) when component mounts or updates.

2. Custom Hooks:

`useFetchPlayers`

- Purpose: Fetches player data (from an API endpoint, currently replaced with sample data).
- Functionality:
  - `useState` to store player data (`players`) and loading state (`loading`).
  - `useEffect` to trigger the data fetching process:
    - `fetchPlayersData` function attempts to fetch data asynchronously.
    - Sample player data (`initialPlayers`) is used for now.
    - Updates `players` with fetched data and `loading` to `false` on success or error.

```
const useFetchPlayers = () => {
  const [players, setPlayers] = useState([]);
  const [loading, setLoading] = useState(true);
 useEffect(() => {
    const fetchPlayersData = async () => {
      try {
        // Replace with API endpoint
        // const response = await fetch('your-api-endpoint/
players');
        // const data = await response.json();
        setPlayers(initialPlayers); // <-- Use initial data for
now
      } catch (error) {
        console.error('Error fetching player data:', error);
      } finally {
        setLoading(false);
      }
    };
    fetchPlayersData();
  }, []);
  return { players, loading };
};
```

`useFilterPlayers`

- Purpose: Filters the `players` array based on a search `filter`.
- Functionality:
  - Takes `players` and `filter` as arguments.
  - Returns a new array containing players whose names match the filter (case-insensitive).
  - Can be expanded to filter on other criteria (position, age, etc.).

```
const useFilterPlayers = (players, filter) => {
  return players.filter(player =>
    player.name.toLowerCase().includes(filter.toLowerCase())
    // Add other filter criteria as needed (e.g., position, age)
  );
};
```

`useSortPlayers`

- Purpose: Sorts the `players` array based on the `sortOrder` (ascending or descending).
- Functionality:
  - Takes `players` and `sortOrder` as arguments.
  - Creates a copy of the `players` array to avoid mutating the original.
  - Returns the sorted array based on player names.

```
const useSortPlayers = (players, sortOrder) => {
  return [...players].sort((a, b) => {
    if (sortOrder === 'asc') {
      return a.name.localeCompare(b.name);
    } else {
      return b.name.localeCompare(a.name);
    }
  });
};
```

3. Main Component: `PlayerSearch`

- Purpose: The main user interface for searching and displaying player data.
- Functionality:
  - Uses `useFetchPlayers` to get the player data and loading status.
  - `useState` to manage the search filter (`filter`) and sort order (`sortOrder`).

- Uses `useFilterPlayers` and `useSortPlayers` to filter and sort the player data based on user input.
- Renders:
  - Input field for searching by name.
  - Dropdown to select sort order.
  - "Loading..." message while data is being fetched.
  - List of filtered and sorted players when data is available.

```
// Main Component
function PlayerSearch() {
  const { players, loading } = useFetchPlayers();
  const [filter, setFilter] = useState('');
  const [sortOrder, setSortOrder] = useState('asc');

  const filteredPlayers = useFilterPlayers(players, filter);
  const sortedPlayers = useSortPlayers(filteredPlayers,
sortOrder);

  return (
    <div>
      <h2>Player Search</h2>
      <input
        type="text"
        placeholder="Search by name"
        value={filter}
        onChange={e => setFilter(e.target.value)}
      />

      <select value={sortOrder} onChange={e =>
setSortOrder(e.target.value)}>
        <option value="asc">Name (A-Z)</option>
        <option value="desc">Name (Z-A)</option>
      </select>

      {loading ? (
        <p>Loading players...</p>
      ) : (
        <ul>
          {sortedPlayers.map(player => (
            <li key={player.id}>
              {player.name} - {player.position}
            </li>
          ))}
        </ul>
      )}
    </div>
```

```
  );
}
```

4. Sample Player Data:

JavaScript

```
const initialPlayers = [
  { id: 1, name: "Mohamed Salah", position: "Forward" },
  { id: 2, name: "Virgil van Dijk", position: "Defender" },
  { id: 3, name: "Alisson Becker", position:
"Goalkeeper" },
  // ... add more players here
];
```

- This array represents sample player data used in the absence of a real API call.

```
// Sample Player Data (Replace with API call)
const initialPlayers = [
  { id: 1, name: "Mohamed Salah", position: "Forward" },
  { id: 2, name: "Virgil van Dijk", position: "Defender" },
  { id: 3, name: "Alisson Becker", position: "Goalkeeper" },
  // ... add more players here
];

export default PlayerSearch;
```