```
// ------------------------------------------------------------

// Array of user names (our sample database for searching)

// ------------------------------------------------------------

const users = [

    "Alice Johnson",
    "Bob Smith",
    "Charlie Brown",
    "David Williams",
    "Evelyn Taylor",
    "Frank Miller",
    "Grace Lee",
    "Hannah Davis",
    "Ian Thomas"

];

// ------------------------------------------------------------

// Getting references to HTML elements

// ------------------------------------------------------------

// Input box where the user types the search query

const searchInput = document.getElementById("search");

// UL element where the list of users will be displayed

const userList = document.getElementById("userList");

// Element that shows "No Results Found"

const noResults = document.querySelector(".no-results");
```

```
38
39  // ------------------------------------------------------------
40
41  // Function: displayUsers
42
43  // Purpose: Displays a list of users in the UI
44
45  // ------------------------------------------------------------
46
47  const displayUsers = (filteredUsers) => {
48
49      // Clear previous results before showing new ones
50
51      userList.innerHTML = "";
52
53      // If no matched users found, show "No Results" message
54
55      if (filteredUsers.length === 0) {
56
57          noResults.style.display = "block";
58
59          return;  // Stop the function here
60      }
61
62      // If results exist, hide the "No Results" message
63
64      noResults.style.display = "none";
65
66      // Loop through each user and create a <li> for it
67
68      filteredUsers.forEach((user) => {
69
70          const li = document.createElement("li");  // create list item
71
72          li.textContent = user;                    // set user name inside <li>
73
74          userList.appendChild(li);                 // add <li> to the list
75      });
```

```
};

// ------------------------------------------------------------

// Function: debounce

// Purpose: Delays execution of a function until user stops typing

// ------------------------------------------------------------

// Example: delay = 1000 means "run the function only after 1 second of no typing"

// Why debounce?

// - Prevents function from running on every keystroke.

// - Improves performance, especially for search or API calls.

const debounce = (fn, delay) => {

    let timeoutId;  // stores the timer ID

    return function (...args) {  // rest operator collects all arguments

        // Clear previously set timer (if user is still typing)

        clearTimeout(timeoutId);

        // Start a new timer – this will run only if user stops typing

        timeoutId = setTimeout(() => {

            fn(...args);  // call the original function with its arguments

            console.log("✅ Debounce fired after", delay / 1000, "seconds");

        }, delay);
    };
```

```
};

// ------------------------------------------------------

// Function: searchUsers

// Purpose: Filter users based on typed text

// ------------------------------------------------------

const searchUsers = (searchTerm) => {

    // Convert both user name & search term to lowercase

    // to make search case-insensitive

    const filteredUsers = users.filter((user) => user.toLowerCase().includes(searchTerm.toLowerCase()));

    // Display matching results

    displayUsers(filteredUsers);
};

// ------------------------------------------------------

// Show all users when the page first loads

// ------------------------------------------------------

displayUsers(users);

// ------------------------------------------------------

// Create a debounced version of searchUsers

// delay = 1000 ms (1 second)

// ------------------------------------------------------
```

```
152
153 const debouncedSearch = debounce(searchUsers, 1000);
154
155 // --------------------------------------------------------
156
157 // Add event listener to input box. This triggers on every keystroke
158
159 // --------------------------------------------------------
160
161 searchInput.addEventListener("input", (event) => {
162
163     // Send the typed value to the debounced search function
164
165     debouncedSearch(event.target.value);
166 });
```