

hrdept@iconnectsolutions.com

HACK ROUND STATEMENT

Java script

*<Please fill in the details in the sections provided below before submitting this document back to HR>*

This document outlines the hack statement and evaluation details for Hack round.

V2.0

**Candidate Details**

|  |  |  |
| --- | --- | --- |
| **Name** | **Email ID** | **Mobile** |
| Swapnil Sanjay Kale | swapnilsanjaykale@mitacsc.edu.in | 9370885243 |

**Problem Statement**

|  |
| --- |
| Create a Javascript application :  Below are two list of User  List1=[‘Arjun’, ‘Adwait’, ‘Swapnil’,’Amit’, ‘Vishal’, ‘Adnan’]  List2=[ ‘Adwait’,’Laxman’,’Amit’, ‘Adnan’,’Nitin’,’Gaurav’]  Find following output (Use collection)  a. Get a unique set of users from List1 which are not in List2  b. Get a unique set of users from List2 which are not in List1  c. Get a set of users who are present in in List1 and List2 both (intersection of list1 & list2)  INSTRUCTIONS  You are allowed to browse internet, but not copy code from other sites. Your submission should strictly be via GitHub. Please note that the purpose of test is not only completion but to ascertain how well you follow coding standards. |

**Timeline**

|  |
| --- |
| 24 hrs.  Please ensure that your test is delivered within this timeline. Any deviation has to be brought to our attention before you cross the allocated timeline. |

**Git Repo**

|  |
| --- |
| https://github.com/SwapnilSanjayKale/Iconnect\_Solutions- |

**Understanding of the problem**

|  |
| --- |
| We have two lists of users, List 1 and List 2. Each list contains information about different users, and user has an object with properties such as `id` and `name`. The `id` property is unique.   1. We have to find list1 users present in list2 2. We want to find List2 users present in List1 3. We want to find the unique set of users who are present in both List 1 and List 2 |

**Approach**

|  |
| --- |
| To find the intersection of List 1 and List 2, we can use a set-based approach in JavaScript. Sets are data structures that allow us to store unique values. By creating sets from the user IDs in each list, we can easily find the common user IDs present in both sets using intersection operations. After obtaining the common user IDs, we then filter out the actual user objects from List 1 that correspond to these common IDs to get the final result. |

**Positive and Negative Test Cases**

|  |
| --- |
| POSITIVE:   1. List with duplicate 2. List with no duplicate   NEGATIVE :   1. Empty List 2. Null List 3. Invalid user object |

**FAQs**

|  |
| --- |
|  |