## JavaScript basics

const users = [

{

firstName: "Oliver",

lastName: "Jake",

isActive: true,

role: "student",

registeredAt: 1625112000000

},

{

firstName: "Connor",

lastName: "Liam",

isActive: true,

role: "student",

registeredAt: 1609477200000

},

{

firstName: "Charlie",

lastName: "",

isActive: true,

role: "admin",

registeredAt: 1619841600000

},

{

firstName: "Thomas",

lastName: "",

isActive: true,

role: "student",

registeredAt: 1612155600000

},

{

firstName: "George",

lastName: "Reece",

isActive: true,

role: "superAdmin",

registeredAt: 1614574800000

},

{

firstName: "Oscar",

lastName: "Rhys",

isActive: false,

role: "superAdmin",

registeredAt: 1617249600000

},

{

firstName: "William",

lastName: "Damian",

isActive: false,

role: "student",

registeredAt: 1609477200000

}

];

1. Use ES6 methods to do the processing on data
   1. Write a function named 'filterDeactivatedUsers' which will return a list of users who have deactivated their profile.
   2. Write a function named as 'getUserFirstAndLastNameAppended' Return array of all users with firstName and lastName appended and separated by space. If lastName is not available then trim the name.
   3. Write a function named 'getCountOfUsersAfterGivenDate', the function will take an object parameter and the object will have keys "role" and "date". The value of "date" will be a Javascript time in milliseconds. Return the Count of users with a given role and the registration date of the user should be greater than the given date. In data, the registeredAt value is a Javascript time in milliseconds.
   4. Write a function named ‘sortUsersByDate’, the function should take sortOrder as parameter, the value of sortOrder passed can be ‘asc’ or ‘desc’. The function should sort the users by date in a given sort order respectively. Please note, the original data/array should not be mutated.
2. Simulate API calls using promises and write definitions for the invoking calls. Following are the guidelines.
   1. getUsers should return an API promise
   2. The above return functions should be used in the below given calls

// Invocation 1

getUsers()

.then(filterDeactivatedUsers)

.then(getUserFirstAndLastNameAppended)

.then(users => console.log(users));

// Invocation 2

getUsers()

.then(getCountOfUsersAfterGivenDate({role:"student", "date": ""}))

.then(count => console.log("count of users", count));

## NodeJs API

1. Guidelines
   1. The database used can be MongoDB
   2. All api's should have a strict validation on the server side and should return relevant errors with http 400 bad request
   3. For routing strictly use Expressjs
   4. For token strictly use JWT
   5. All protected API's should only be allowed through a JWT, if JWT not given should return http 400 unauthorized access. Also, only users with role ‘admin’ and ‘superAdmin’ will have access to the protected API’s
   6. No raw query to be used, strictly use Mongoose ORM
2. Database Schema
   1. Users schema

|  |  |
| --- | --- |
| user\_id | ObjectId |
| first\_name | string not null |
| last\_name | string not null |
| role\_id | integer foreign key Role(role\_id) not null |
| email | string not null |
| password | String (encoded) |
| is\_active | boolean default true |
| created\_at | timestamp not null |
| updated\_at | timestamp not null |

* 1. Role schema

|  |  |
| --- | --- |
| role\_id | ObjectId |
| name | string |
| is\_active | boolean default true |
| created\_at | timestamp not null |
| updated\_at | timestamp not null |

1. Guest API's

API - /api/v1/login

Method - POST

Params - email, password

1. Protected API's
   1. list of all users

API - /api/v1/users

Method - GET

Response -

{

"success": true,

"data": {

"users: [

{

"firstName": "",

"lastName": "",

"role": "", // this is name in role table

"registeredAt": "" // this is created\_at in user table

},

... more users

]

}

}

* 1. list of users by role

API - /api/v1/users?role=student

Method - GET

Response - // Response should be same as /api/v1/users

* 1. count of each users by role

API - /api/v1/users/role/count

Method - GET

Response -

// the count given is just an example

{

"success": true,

"data": {

"student": 5,

"admin": 2,

"superAdmin": 3

}

}