LOVELY PROFESSIONAL UNIVERSITY

# Assignment

**CAP919**

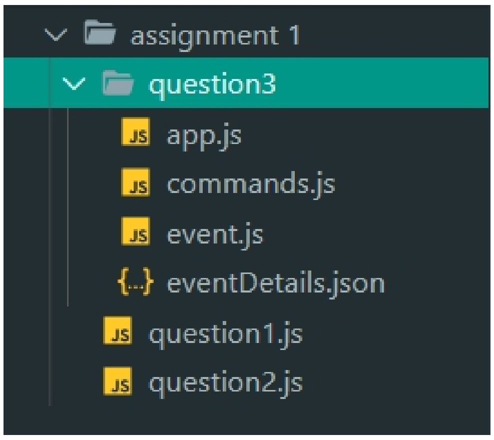
**SERVER SIDE DEVELOPMENT WITH NODE.JS**

**Submitted by**

**Nitesh khatri**
**(11813971)**

**Set E**

**Download the complete project ->**
https://drive.google.com/file/d/1029qOW3THm2qjAuJDciKdRhFoonOQfoA/view?usp=sharing

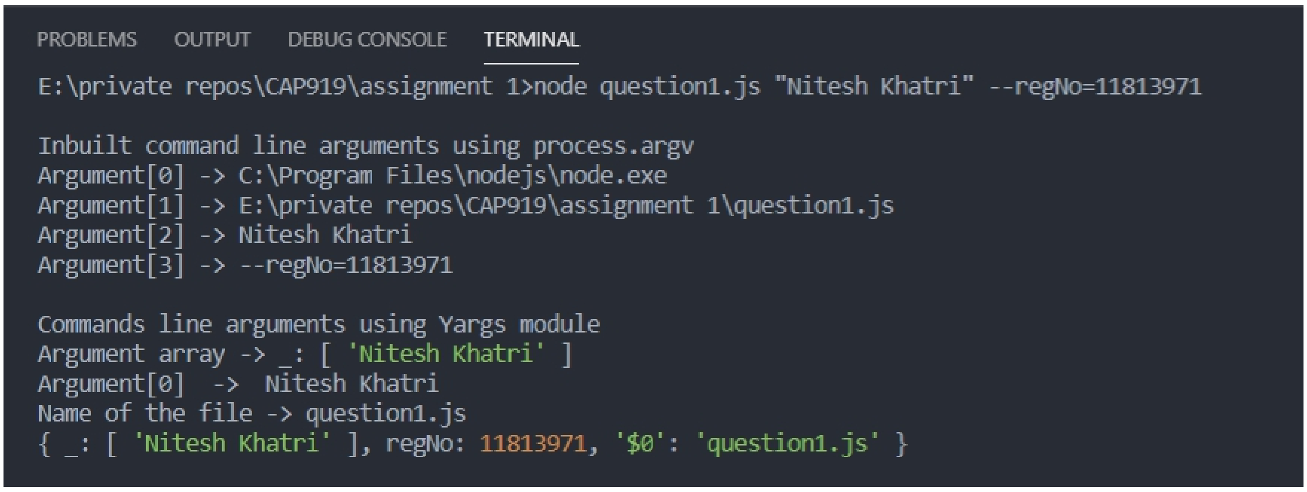


Q1. What are the various ways of providing user input have you practiced? What are the differences between those methods? [10]

```
// Q1 - What are the various ways of providing user input have you practiced?
// What are the differences between those methods?

// Mainly their are two ways of taking user input in node js which are following
// 1. Passing the user input from the input field in html page to node js file
// 2. Taking input from the command prompt using command line arguments

// Command line arguments using inbuilt method
// Global object 'process' is used to capture cmd line arguments in a array
// argv[] is the array in which cmd argument are stored
// The array can be accessed by process.argv

// Demonstration of command line arguments using Default inbuilt method

console.log('\nInbuilt command line arguments using process.argv')

var arguments = process.argv;
for (i = 0; i < arguments.length; i++) {
  console.log(`Argument[${i}] -> ${arguments[i]}`);
}

// output
// E:\Sem 6\Cap 919 Node js\ca material\assignment 1>node question1.js "Nitesh khatri"  11813
971
// Argument[0] -> C:\Program Files\nodejs\node.exe
// Argument[1] -> E:\Sem 6\Cap 919 Node js\ca material\assignment 1\question1.js
// Argument[2] -> Nitesh khatri
// Argument[3] -> 11813971
// argv[0] and argv[1] will always be present in the argv array even if no arguments is passe
d
```

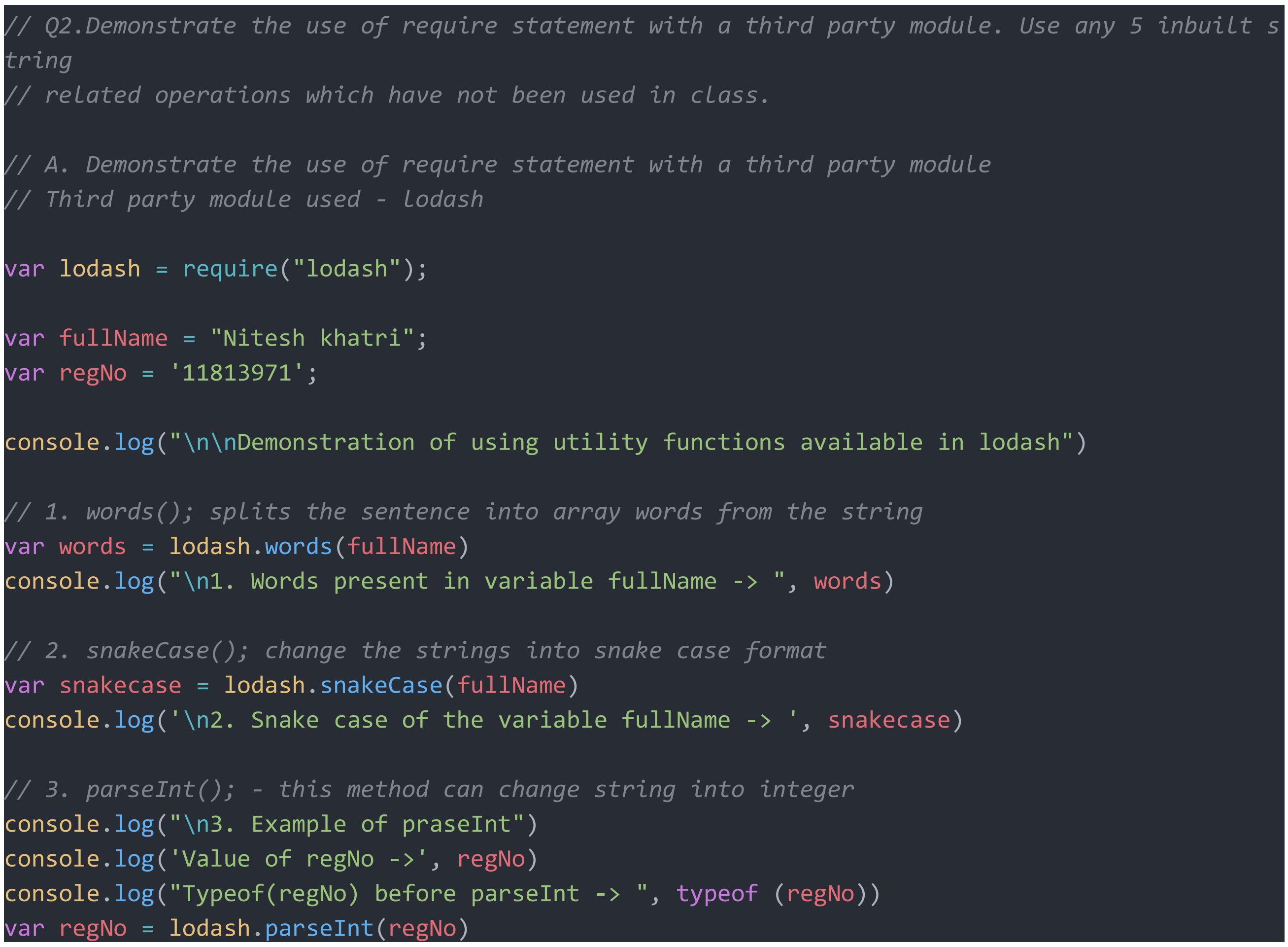
```
// These two are passed by default where [0] argument is the node.exe location and [1] is the
 path of the file

// -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
-_-_-_-_-_-_-_-_-_ //

// Command line arguments using third party module yargs
// yargs is little advance method to take user input through command line
// It stores the arguments in a object which contain first key:value pair as _:[] a array of
arguments
// second key:value pair is '$0':'Name of the executed file'
// Yargs also provide us the option to create or define our own commands(flags)
// we can also pass arguments like - node question1.js "Normal argument" --user="Nitesh khatr
i" --reg="11813971"
// output ->
// {
// _: ['Normal argument'],
//   user: 'Nitesh khatri',
//   reg: 11813971,
//   '$0': 'question1.js'
// }

// Demonstration

console.log('\nCommands line arguments using Yargs module')

const Yargs = require("yargs");
const yargsArguments = Yargs.argv;

console.log("Argument array -> _:", yargsArguments._);

for (var i = 0; i < yargsArguments._.length; i++) {
  console.log(`Argument[${i}]  ->  ${yargsArguments._[i]}`);
}

console.log(`Name of the file -> ${yargsArguments.$0}`);
console.log(yargsArguments)

// Difference between default and yargs method is their functionality and their way of storin
g the arguments
// in default method arguments are stored in a array with 2 default arguments and in yargs al
so it is stored
// in a array but that array is present inside a object and array only contain arguments pass
ed by the user.
// Another difference is that yargs provide additional features like defining our own command
s which is not
// available in default method
```

**Output**



```
E:\private repos\CAP919\assignment 1>node question1.js "Nitesh Khatri" --regNo=11813971

Inbuilt command line arguments using process.argv
Argument[0] -> C:\Program Files\nodejs\node.exe
Argument[1] -> E:\private repos\CAP919\assignment 1\question1.js
Argument[2] -> Nitesh Khatri
Argument[3] -> --regNo=11813971

Commands line arguments using Yargs module
Argument array -> _: [ 'Nitesh Khatri' ]
Argument[0]  ->  Nitesh Khatri
Name of the file -> question1.js
{ _: [ 'Nitesh Khatri' ], regNo: 11813971, '$0': 'question1.js' }
```

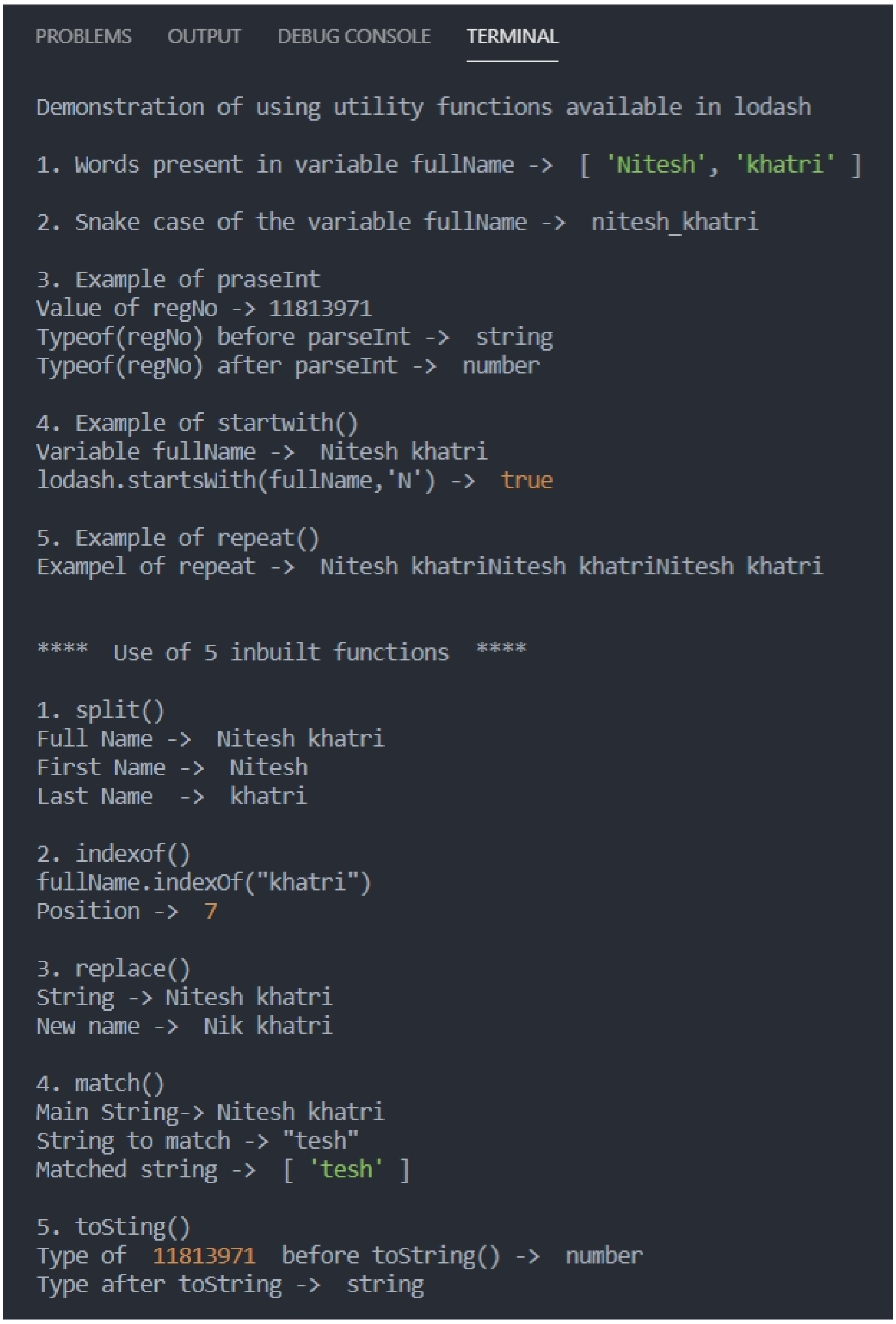
**Q2. Demonstrate the use of require statement with a third party module. Use any 5 inbuilt string related operations which have not been used in class. (Try to keep it unique) [10].**



```
// Q2.Demonstrate the use of require statement with a third party module. Use any 5 inbuilt s
tring
// related operations which have not been used in class.

// A. Demonstrate the use of require statement with a third party module
// Third party module used - lodash

var lodash = require("lodash");

var fullName = "Nitesh khatri";
var regNo = '11813971';

console.log("\n\nDemonstration of using utility functions available in lodash")

// 1. words(); splits the sentence into array words from the string
var words = lodash.words(fullName)
console.log("\n1. Words present in variable fullName -> ", words)

// 2. snakeCase(); change the strings into snake case format
var snakecase = lodash.snakeCase(fullName)
console.log('\n2. Snake case of the variable fullName -> ', snakecase)

// 3. parseInt(); - this method can change string into integer
console.log("\n3. Example of praseInt")
console.log('Value of regNo ->', regNo)
console.log("Typeof(regNo) before parseInt -> ", typeof (regNo))
var regNo = lodash.parseInt(regNo)
```

```
console.log("Typeof(regNo) after parseInt -> ", typeof (regNo))

// 4. startWith(); - returns true if a string starts with the passed string
console.log('\n4. Example of startwith()')
console.log('Variable fullName -> ', fullName)
var startbool = lodash.startsWith(fullName, "N")
console.log("lodash.startsWith(fullName,'N') -> ", startbool)

// 5. repeat(); -> String will get repeated the numbers of times according to the value passe
d
console.log('\n5. Example of repeat()')
var repeated = lodash.repeat(fullName, 3)
console.log('Exampel of repeat -> ', repeated)

// B. 5 inbuilt string function operations available in node js

console.log("\n\n****  Use of 5 inbuilt functions  ****")

// 1. split(); - Split a string using a separator
var firstName = fullName.split(" ")[0];
var lastName = fullName.split(" ")[1];
console.log('\n1. split()')
console.log('Full Name -> ', fullName)
console.log("First Name -> ", firstName);
console.log("Last Name  -> ", lastName);

// 2. indexof(); - Finds the index of sub string from a string
console.log('\n2. indexof()')
console.log('fullName.indexOf("khatri")')
var position = fullName.indexOf("khatri");
console.log('Position -> ', position);

// 3. replace(); - Replace a string with another string
console.log('\n3. replace()')
console.log("String ->", fullName)
var newName = fullName.replace("Nitesh", "Nik");
console.log("New name -> ", newName);

// 4. match(); - return a matched string using regex. return null if not matched
console.log('\n4. match()')
console.log('Main String->', fullName, '\nString to match -> "tesh"')
var stringMatch = fullName.match(/tesh/g);

console.log('Matched string -> ', stringMatch);

// 5. toString(); - will convert Number to string type
console.log('\n5. toSting()')
console.log("Type of ", regNo, " before toString() -> ", typeof (regNo))
var reg = regNo.toString();
console.log('Type after toString -> ', typeof (reg));
```

# Output



```
Demonstration of using utility functions available in lodash

1. Words present in variable fullName ->  [ 'Nitesh', 'khatri' ]

2. Snake case of the variable fullName ->  nitesh_khatri

3. Example of praseInt
Value of regNo -> 11813971
Typeof(regNo) before parseInt ->  string
Typeof(regNo) after parseInt ->  number

4. Example of startwith()
Variable fullName ->  Nitesh khatri
lodash.startsWith(fullName,'N') ->  true

5. Example of repeat()
Exampel of repeat ->  Nitesh khatriNitesh khatriNitesh khatri


****  Use of 5 inbuilt functions  ****

1. split()
Full Name ->  Nitesh khatri
First Name ->  Nitesh
Last Name  ->  khatri

2. indexof()
fullName.indexOf("khatri")
Position ->  7

3. replace()
String -> Nitesh khatri
New name ->  Nik khatri

4. match()
Main String-> Nitesh khatri
String to match -> "tesh"
Matched string ->  [ 'tesh' ]

5. toSting()
Type of  11813971  before toString() ->  number
Type after toString ->  string
```

**Q:3 Note App based Application: Create an application which adds registration details for an event, cancels the registration and list all the registrations for a given event.**

## App.js

```
// Node Event Management Application:
// Q3. Create an application which adds registration details for an event, cancels the regist
ration and list all
// the registrations for a given event.

// Commands -
// Add Registration( Register for the event ),
// Cancel the registration of a user
// List all the registration

// Events -> Hackaton Run
// categories -> 1. coding sprint, 2.datathon, 3. hackathon
// Type -> 1. online, 2. offline

// Register User ✅ -> node app.js add --id="" --user="" --category="" --course="" --type=""
// --id is optional and can be skipped only in case of Register command

// Cancel registration ❌ -> node app.js cancel --id=""
// --id="" is mandatory here

// List  node app.js list --category=""
// --category is optional. if provide all the registration from that category will be listed

const commands = require("./commands")
const eventapp = require("./event")
const object = commands.obj

var command = object._[0]

if (command == "add") {
    message = eventapp.register(object.id, object.user, object.category, object.course, objec
t.type)
    console.log(message)
}
else if (command == "cancel") {
    message = eventapp.cancel(object.id,object.category)
    console.log(message)
}
else if(command == "list")
{
    message = eventapp.list(object.category)
}
else{
    console.log('Entered command not recognized')
```

## Commands.js

```
// This file contain all the commands and their options
const yargs = require('yargs')
const obj = yargs.argv

// creating options for the commands

const Category = {
    describe: "Categories of competition",
    demand: true,
    type: "string",
    alias: "ctg"
}

const Course = {
    describe: "Course in which student is studying",
    demand: true,
    type: "string",
    alias: "crs"
}

const Userid = {
    describe: "Id of the registered user",
    demand: true,
    type: "string",
    alias: "id"
}

const User = {
    describe: "Name of the registered user",
    demand: true,
    type: "string",
    alias: "u"
}

const Type = {
    describe: "Online or offline",
    demand: true,
    type: "string",
    alias: "t"
}

// Register user command
yargs.command(
    {
        command:"add",
        id: {
            describe: "Register a new id",
            demand: false,
            type: "string",
        },
        user: User,
        category: Category,
        course: Course,
        type: Type
    }

)

// Cancel registeration command
yargs.command(
    {
        command: "cancel",
        builder: {
            userId: Userid,
        }
    }
)

// list commands
yargs.command(
    {
        command: "list",
        builder: {
            category: {
                describe: "Categories of competition",
                demand: false,
                type: "string",
            }
        }
    }
)
.help().argv

// exporting all the required modules
module.exports =
{
    yargs,
    obj
}
```

## Event.js

```
const fs = require("fs");
const { toNumber } = require("lodash");
let check = false;

// This function will read the json file and then parse the json data into object
function loadjson() {
```

```javascript
    try {
        const data = fs.readFileSync('eventDetails.json');
        return JSON.parse(data); // returning the parsed json data
    }
    catch (e) {
        return []
    }
}

// This function will write the object to the json file
function saveDetails(object) {
    fs.writeFileSync('eventDetails.json', JSON.stringify(object, null, 4))
}

// This function will register a user to the event
function register(id, user, category, course, type) {

    var eventJson = loadjson();
    ++eventJson['eventDetail']['idcounter'];
    ++eventJson['eventDetail']['totalregistrations'];
    var idcounter = eventJson['eventDetail']['idcounter'];

    if (id === undefined) // if id is not defined by the user, one will be generated
    {
        id = (user.split(' ')[0]) + '-' + idcounter;
    }
    else { // if id is already passed by the user
        for (const key in eventJson['registrations']) {

            // Making sure if the passed id is unique or not
            if (id === eventJson['registrations'][key]["id"]) {
                return `
                -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

                  Can not register user with this id
                  Id already exist  ❌

                -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
                `;
            }
        }
    }

    var user = {
        id: id,
        name: user,
        category: category,
        course: course,
        type: type
    }

    // Adding new user details to the object
    eventJson["registrations"][idcounter] = user
    saveDetails(eventJson) // Writing the new details to json file
    return `
    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

        User registered successfully ✅

    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
    `;
}

// function to cancel a existing registration
function cancel(id) {

    var eventJson = loadjson();
    // Deleting the user registration by matching its id
    for (const key in eventJson['registrations']) {
        if (id === eventJson['registrations'][key]["id"]) {

            delete eventJson['registrations'][key]; // Removing the user from the list
            --eventJson['eventDetail']['totalregistrations'];
            saveDetails(eventJson)
            return `
            _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

             Registeration canceled successfully ✅

            _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
    `
        }
    }
    return `
    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

      User with id ${id} Not found ❌

    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
    `;
}

// This function will list all the registered user in the event
// This function will list all the user if optinal parameter --category is not passed
// If the --category value is passed then only that category will be displayed
function list(category) {
    var eventJson = loadjson();
    total = eventJson['eventDetail']['idcounter'];
    current = eventJson['eventDetail']['totalregistrations']
    console.log('\n         *** Hackathon Run ***        \n');
    console.log('Total registered users     -> ', total)
    console.log('Current total users        -> ', current);
    console.log('Total No of Deregistration -> ', total - current)
    console.log('\nHackathon categories');
    for (const key in eventJson['eventDetail']['categories']) {
        console.log(toNumber(key) + 1, eventJson['eventDetail']['categories'][key])
    }
    console.log(`\nType of hackathon category
1. ${eventJson['eventDetail']['type'][0]},
2. ${eventJson['eventDetail']['type'][1]}`)

    console.log('\nParticipants details')

    if (category === undefined) { // checking if the category value is passed by the user
        for (const key in eventJson['registrations']) {
            print(key, eventJson)
        }
        return 0;
    }
    else {
        for (const key in eventJson['registrations']) {
            if (category === eventJson['registrations'][key]['category']) {
                print(key, eventJson)
                check = true;
            }
        }}
    if (check == false) // if id will not match to any existing id in the json file
    {
        console.log(`
    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

      No category found with name '${category}' ❌

    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
    `);
    }
}
// To print the user details
function print(key, object) {
    let obj = object['registrations'][key];
    console.log(`
Id       -> ${obj['id']}
Name     -> ${obj['name']}
Category -> ${obj['category']}
Course   -> ${obj['course']}
type     -> ${obj['type']}
`)
}
// Exporting all the required module
module.exports = {
    register,
    cancel,
    list
}
```

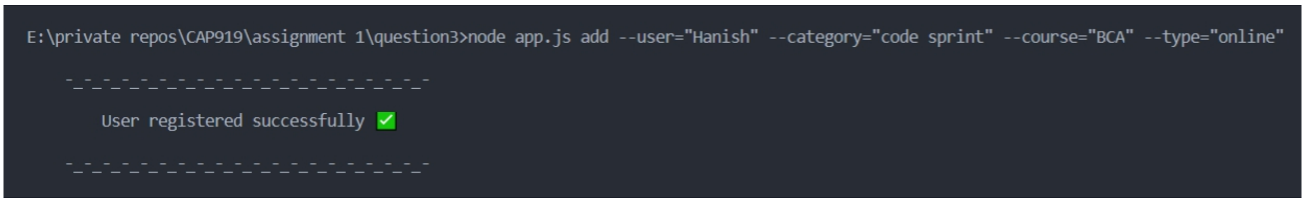
## eventDetails.json

```json
{
    "eventDetail": {
        "eventName": "Hackathon Run",
        "categories": [
            "coding sprint",
            "datathon",
            "hackathon" ],
        "type": [
            "online",
            "offline" ],
        "totalregistrations": 6,
        "idcounter": 7
    },
    "registrations": {
        "1": {
            "id": "Nitesh-1",
            "name": "Nitesh Khatri",
            "category": "hackathon",
            "course": "BCA",
            "type": "online" },
        "2": {
            "id": "Deepak-2",
            "name": "Deepak Khatri",
            "category": "datathon",
            "course": "BCA",
            "type": "online" },
        "3": {
            "id": "Manmohan-3",
            "name": "Manmohan singh",
            "category": "coding sprint",
            "course": "BCA",
            "type": "offline" },
        "4": {
            "id": "Ramandeep-4",
            "name": "Ramandeep singh",
            "category": "datathon",
            "course": "BCA",
            "type": "offline" },
        "5": {
            "id": "Ajit-5",
            "name": "Ajit singh",
            "category": "coding sprint",
            "course": "BCA",
            "type": "online" },
        "6": {
            "id": "Aman-6",
            "name": "Aman",
            "category": "hackathon",
            "course": "BCA",
            "type": "online" }
    }
}
```

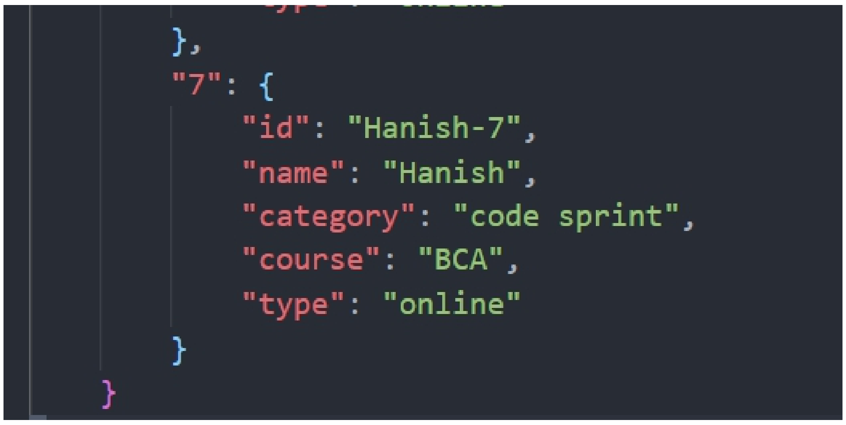
**Output**

If id is not passed during the registration it will be generated automatically

## Adding new registration

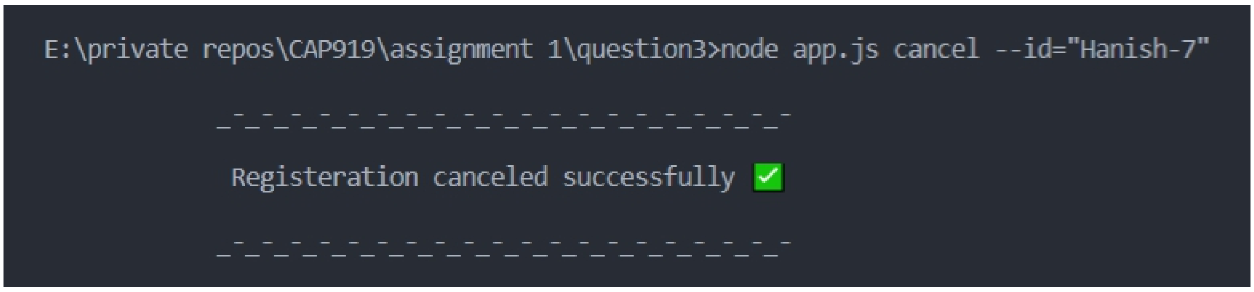


```
E:\private repos\CAP919\assignment 1\question3>node app.js add --user="Hanish" --category="code sprint" --course="BCA" --type="online"

    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

        User registered successfully ✅

    -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
```

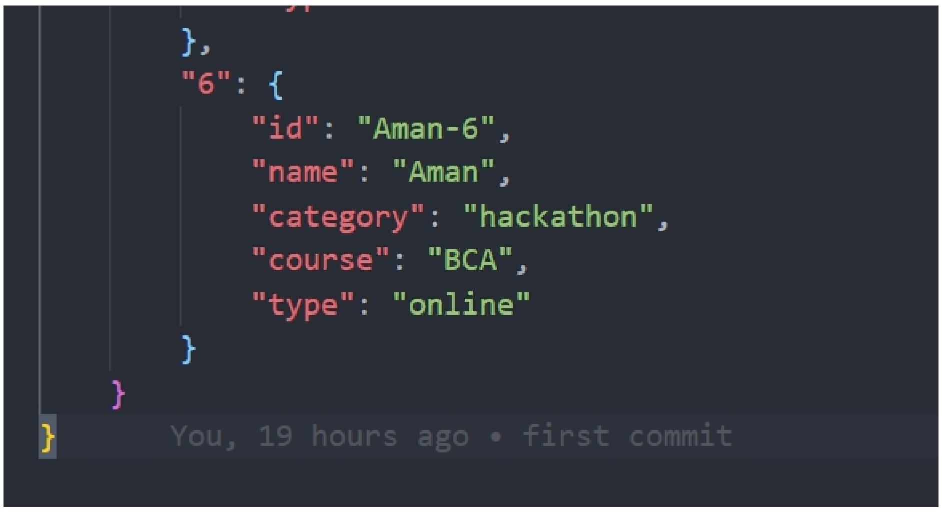


```
        },
        "7": {
            "id": "Hanish-7",
            "name": "Hanish",
            "category": "code sprint",
            "course": "BCA",
            "type": "online"
        }
    }
```

## Canceling a registration

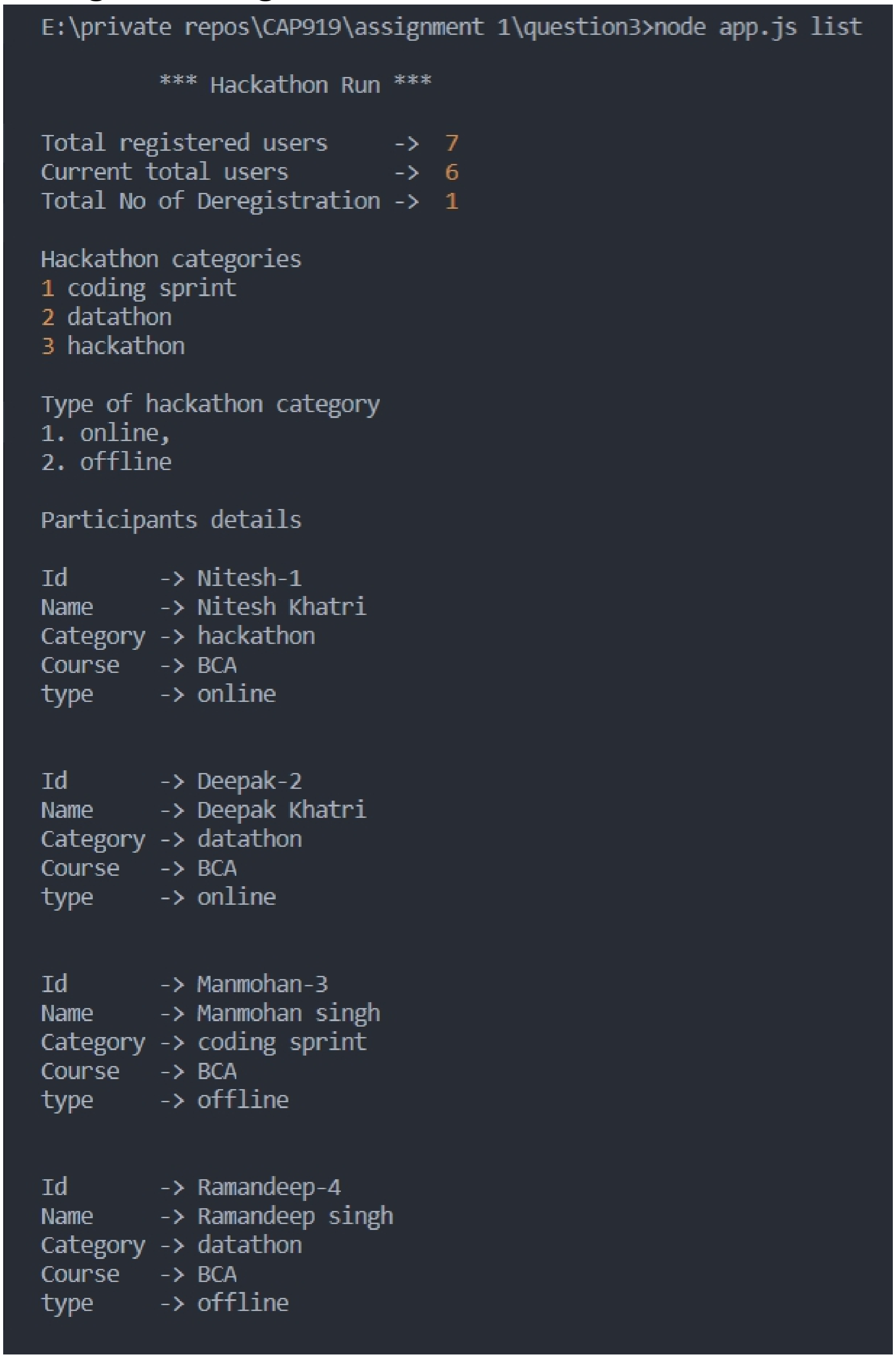


```
E:\private repos\CAP919\assignment 1\question3>node app.js cancel --id="Hanish-7"

            -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

             Registeration canceled successfully ✅

            -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
```

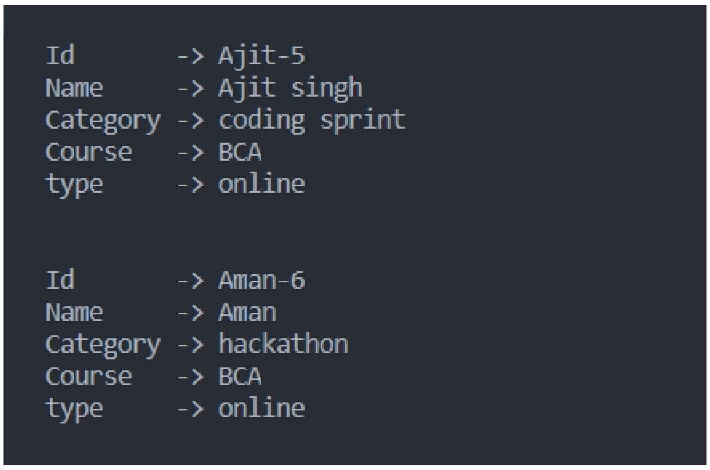


```
        },
        "6": {
            "id": "Aman-6",
            "name": "Aman",
            "category": "hackathon",
            "course": "BCA",
            "type": "online"
        }
    }
}       You, 19 hours ago • first commit
```

## Listing all the registration

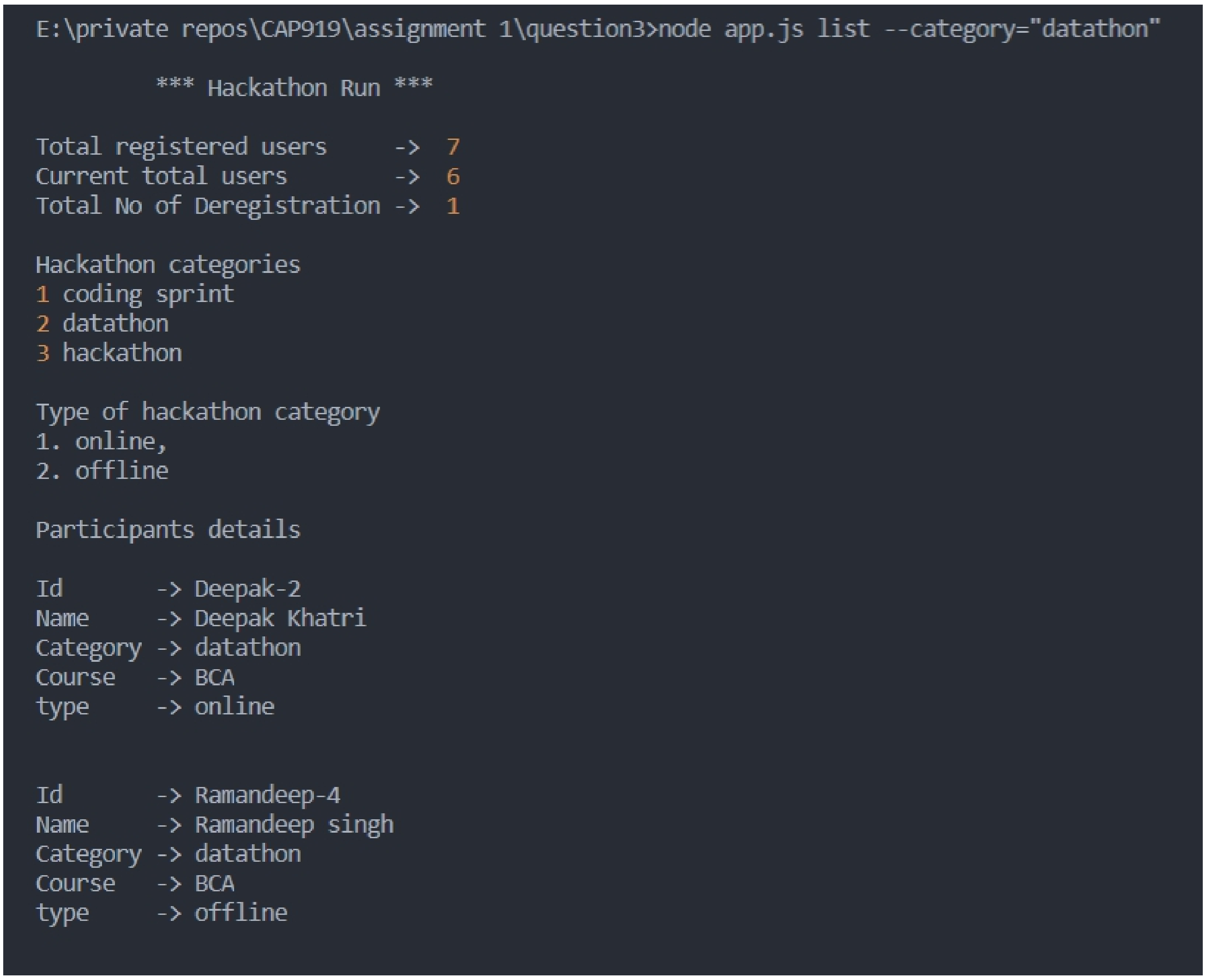


```
E:\private repos\CAP919\assignment 1\question3>node app.js list

         *** Hackathon Run ***

Total registered users      ->  7
Current total users         ->  6
Total No of Deregistration ->  1

Hackathon categories
1 coding sprint
2 datathon
3 hackathon

Type of hackathon category
1. online,
2. offline

Participants details

Id       -> Nitesh-1
Name     -> Nitesh Khatri
Category -> hackathon
Course   -> BCA
type     -> online


Id       -> Deepak-2
Name     -> Deepak Khatri
Category -> datathon
Course   -> BCA
type     -> online


Id       -> Manmohan-3
Name     -> Manmohan singh
Category -> coding sprint
Course   -> BCA
type     -> offline


Id       -> Ramandeep-4
Name     -> Ramandeep singh
Category -> datathon
Course   -> BCA
type     -> offline
```



```
Id       -> Ajit-5
Name     -> Ajit singh
Category -> coding sprint
Course   -> BCA
type     -> online


Id       -> Aman-6
Name     -> Aman
Category -> hackathon
Course   -> BCA
type     -> online
```

## List category wise



```
E:\private repos\CAP919\assignment 1\question3>node app.js list --category="datathon"

         *** Hackathon Run ***

Total registered users      ->  7
Current total users         ->  6
Total No of Deregistration  ->  1

Hackathon categories
1 coding sprint
2 datathon
3 hackathon

Type of hackathon category
1. online,
2. offline

Participants details

Id       -> Deepak-2
Name     -> Deepak Khatri
Category -> datathon
Course   -> BCA
type     -> online


Id       -> Ramandeep-4
Name     -> Ramandeep singh
Category -> datathon
Course   -> BCA
type     -> offline
```