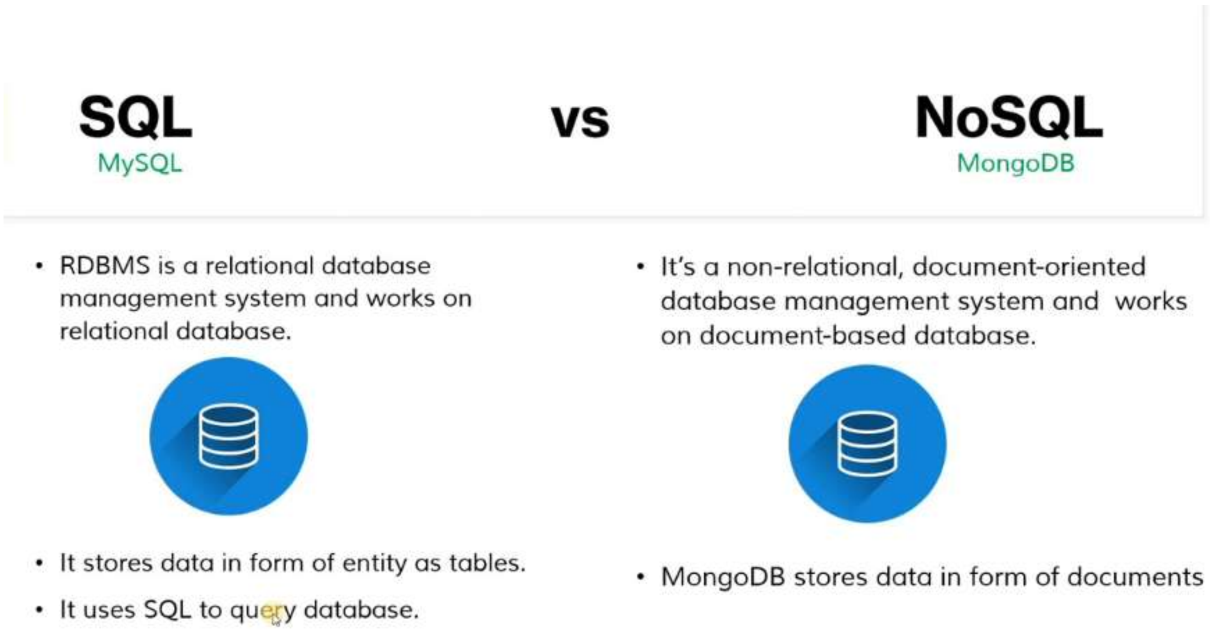
Express and MongoDB

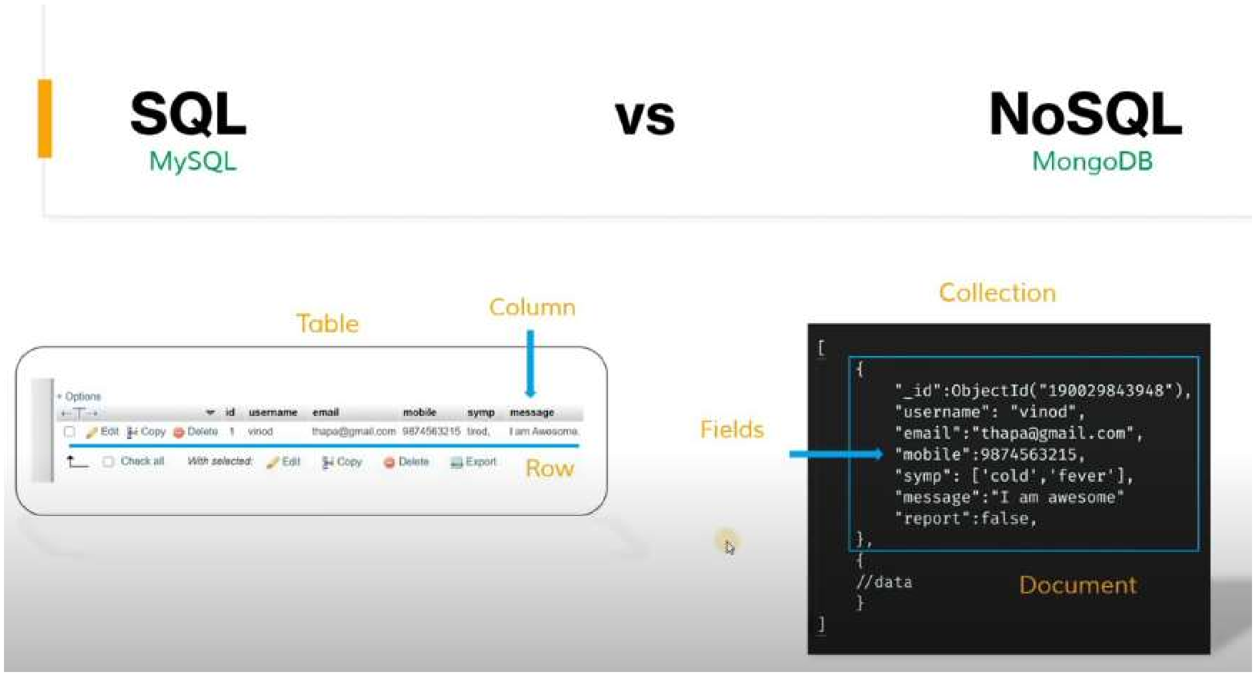
- Using Mongoose to make schemas in MongoDB.
- Making API end points using Express
- Doing CRUD on database MongoDB using Express
- Writing tests using mocha and chai.



SQL MySQL vs **NoSQL** MongoDB

- RDBMS is a relational database management system and works on relational database.
- It stores data in form of entity as tables.
- It uses SQL to query database.

- It's a non-relational, document-oriented database management system and works on document-based database.
- MongoDB stores data in form of documents



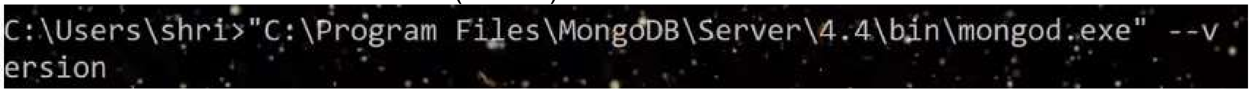
SQL MySQL vs **NoSQL** MongoDB

	id	username	email	mobile	symp	message
Edit Copy Delete	1	vinod	thapa@gmail.com	9874563215	tired,	I am Awesome.

```
[
  {
     "_id":ObjectId("190029843948"),
     "username": "vinod",
     "email":"thapa@gmail.com",
     "mobile":9874563215,
     "symp": ['cold','fever'],
     "message":"I am awesome"
     "report":false,
  },
  {
  //data
  }
]
```

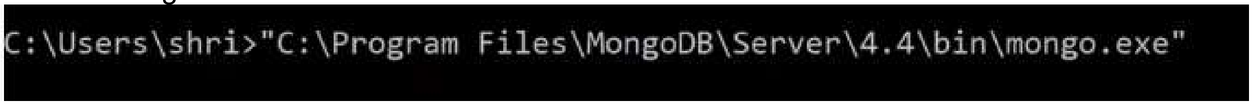
Mongodb.com→ software → Community
Data Directory (copy path) , Log Directory (copy path), Uncheck Mongodb compass

After installation to check version: (in cmd)



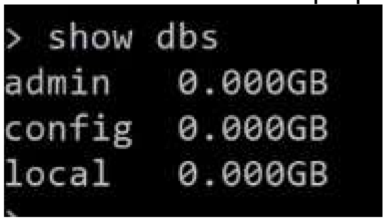
```
C:\Users\shri>"C:\Program Files\MongoDB\Server\4.4\bin\mongod.exe" --v
ersion
```

To Run Mongo:



```
C:\Users\shri>"C:\Program Files\MongoDB\Server\4.4\bin\mongo.exe"
```

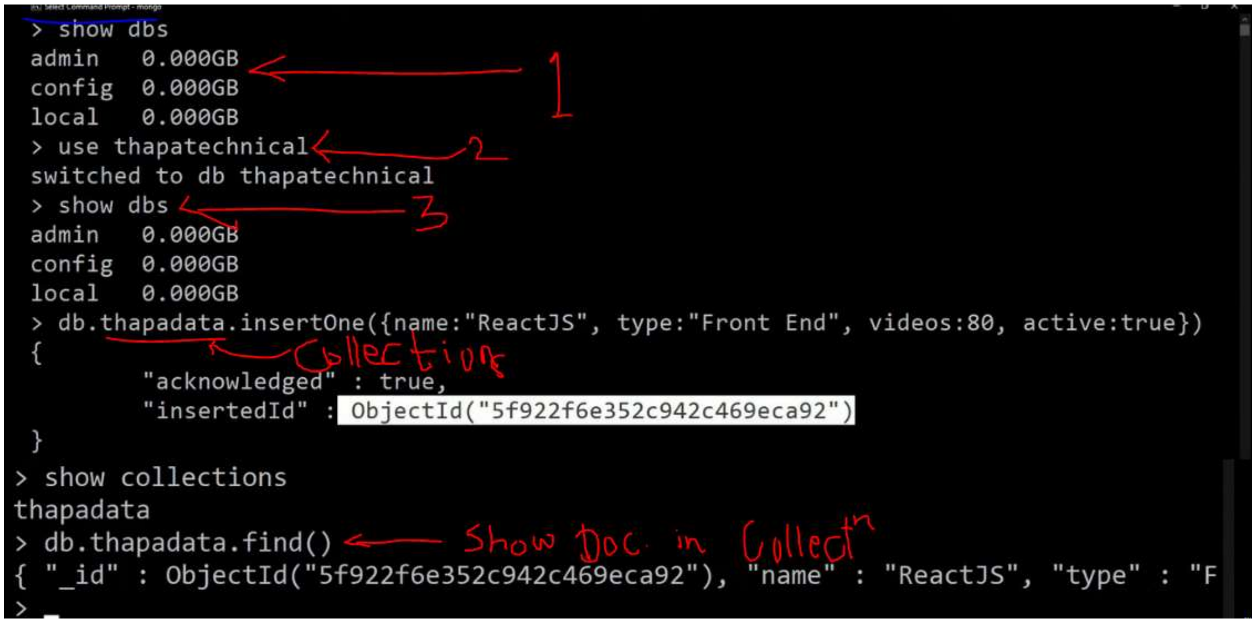
To Check whether Run properly:



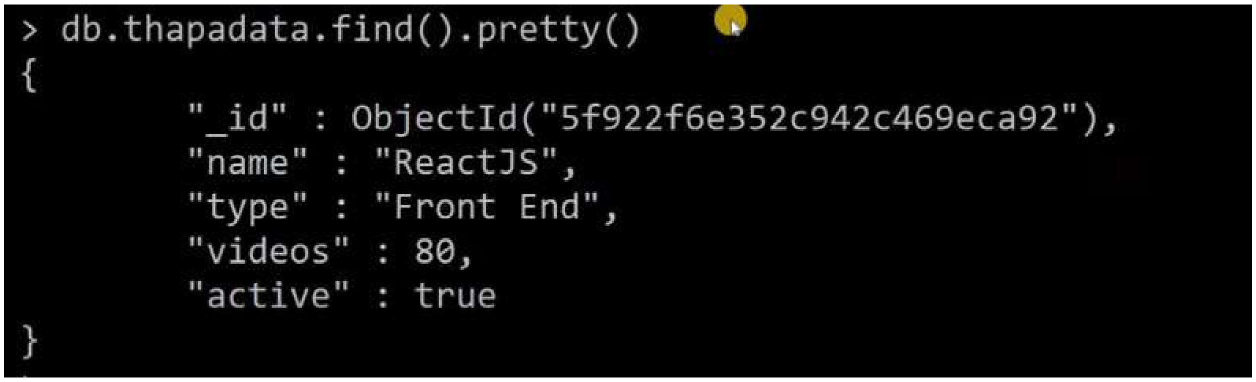
```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

Edit Environment Variable: Type "env" in search → Path → Edit → New → copy paste path (Data Directory path)

1. Create DB, Collections and Docs in MongoDB (CRUD Operations)

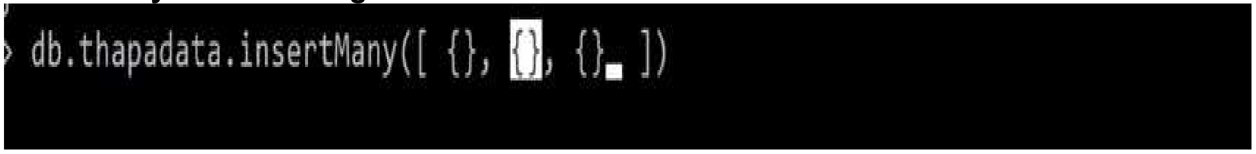


```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use thapatechnical
switched to db thapatechnical
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> db.thapadata.insertOne({name:"ReactJS", type:"Front End", videos:80, active:true})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f922f6e352c942c469eca92")
}
> show collections
thapadata
> db.thapadata.find()
{ "_id" : ObjectId("5f922f6e352c942c469eca92"), "name" : "ReactJS", "type" : "F
>
```



```
> db.thapadata.find().pretty()
{
        "_id" : ObjectId("5f922f6e352c942c469eca92"),
        "name" : "ReactJS",
        "type" : "Front End",
        "videos" : 80,
        "active" : true
}
```

Insert Many Docs in Mongo

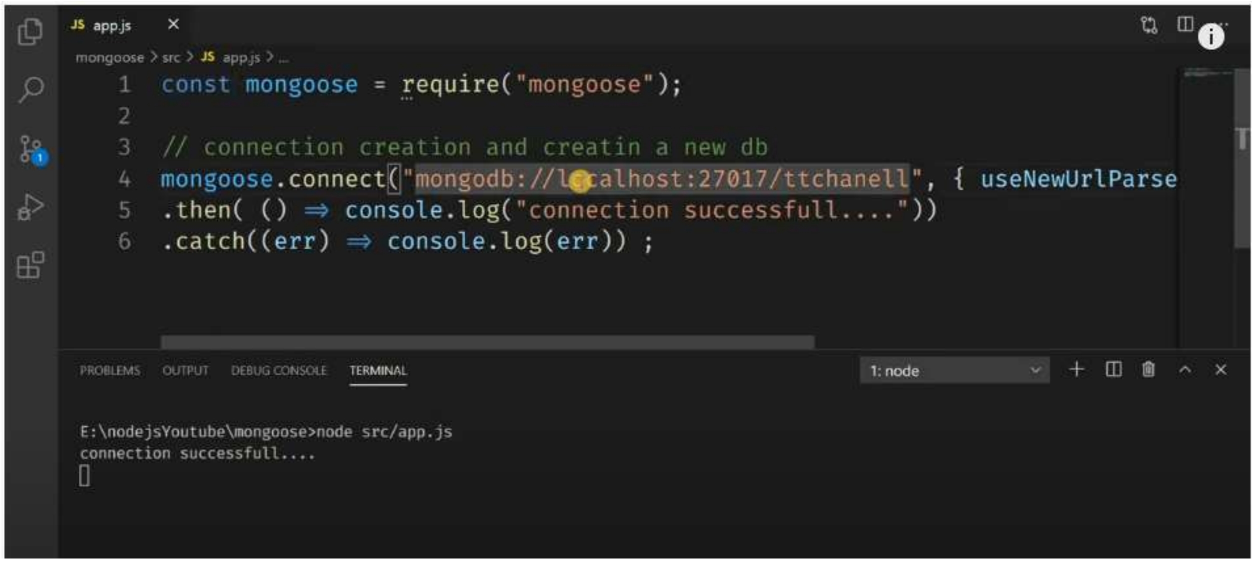


```
> db.thapadata.insertMany([ {}, {}, {} ])
```

2. Using Mongoose to make schemas in MongoDB.

It is an framework which Connect Mongo with Node JS (by Using "Mongoose")

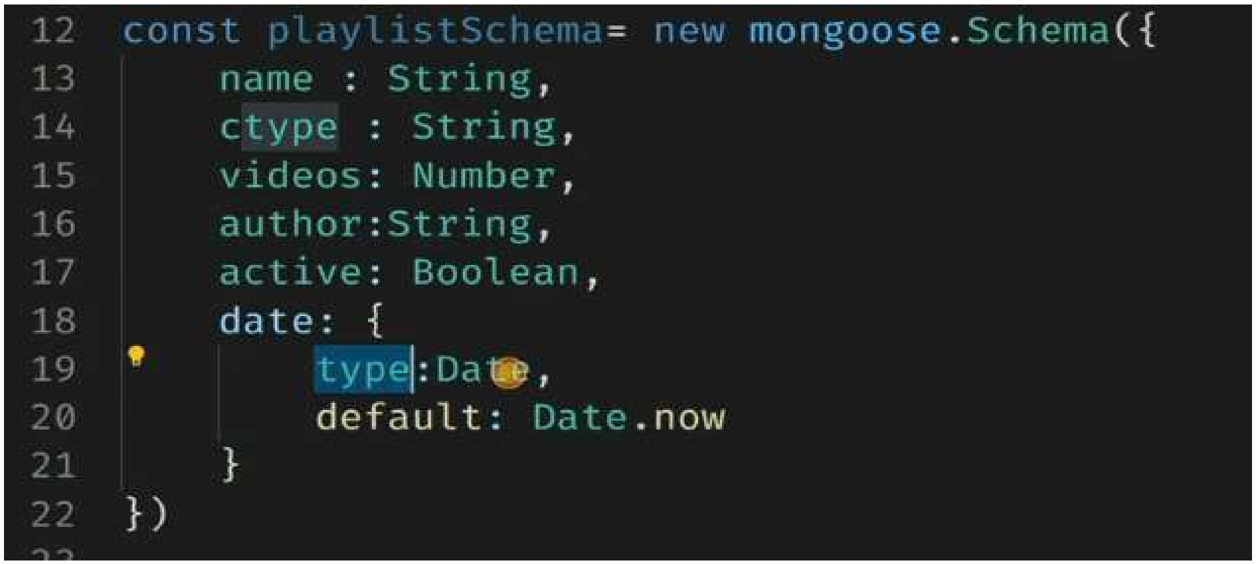
Mongoose installation: `npm i mongoose`
`npm init -y` …. For packages of development



```
const mongoose = require("mongoose");

// connection creation and creatin a new db
mongoose.connect("mongodb://localhost:27017/ttchanell", { useNewUrlParse
.then( () => console.log("connection successfull...."))
.catch((err) => console.log(err)) ;
```

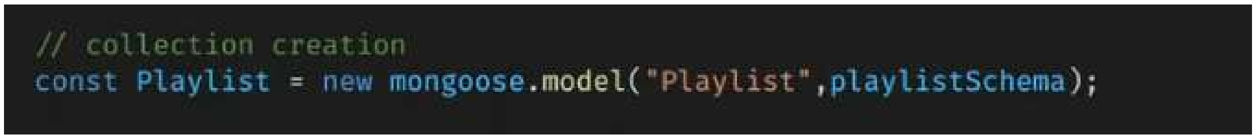
```
E:\nodejsYoutube\mongoose>node src/app.js
connection successfull....
```

Mongo Schema:
Definition:



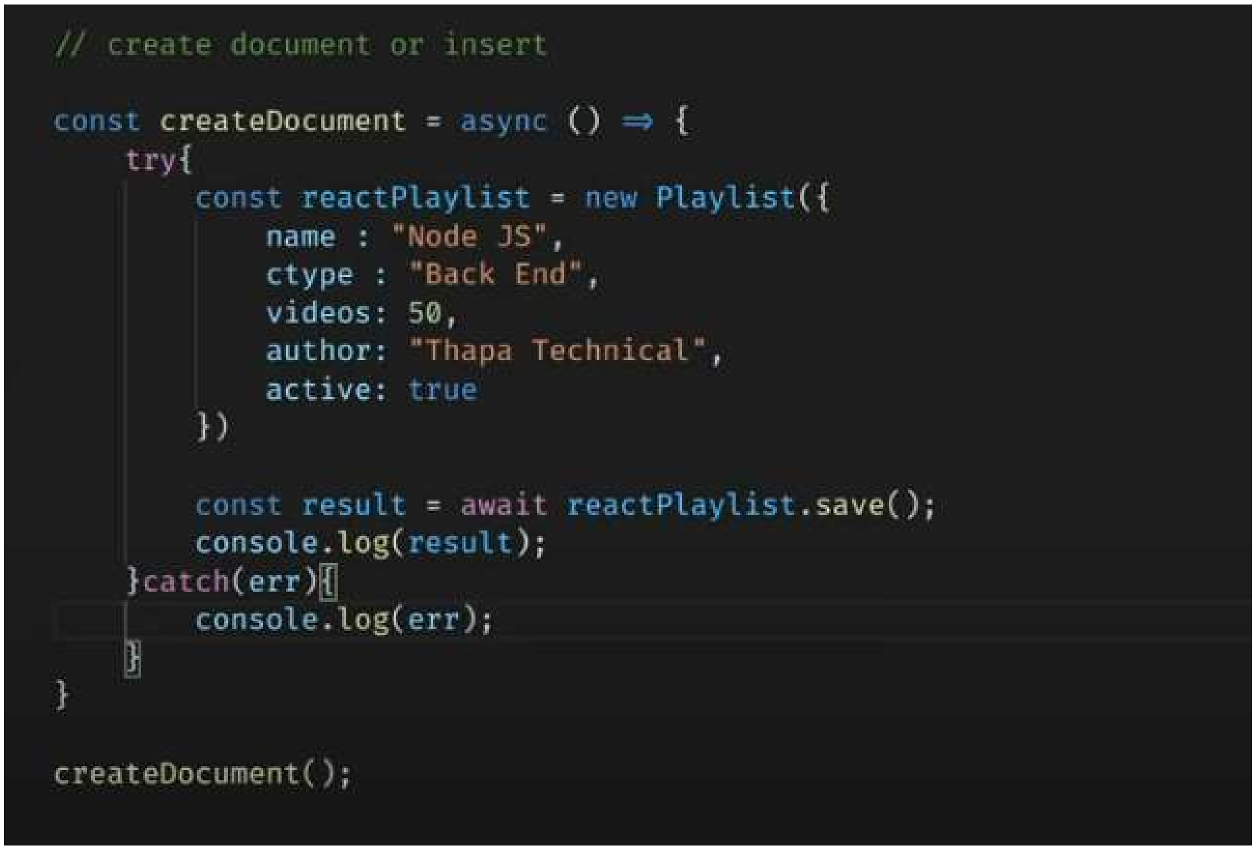
```
const playlistSchema= new mongoose.Schema({
    name : String,
    ctype : String,
    videos: Number,
    author:String,
    active: Boolean,
    date: {
        type:Date,
        default: Date.now
    }
})
```

Now we have to create Collection



```
// collection creation
const Playlist = new mongoose.model("Playlist",playlistSchema);
```

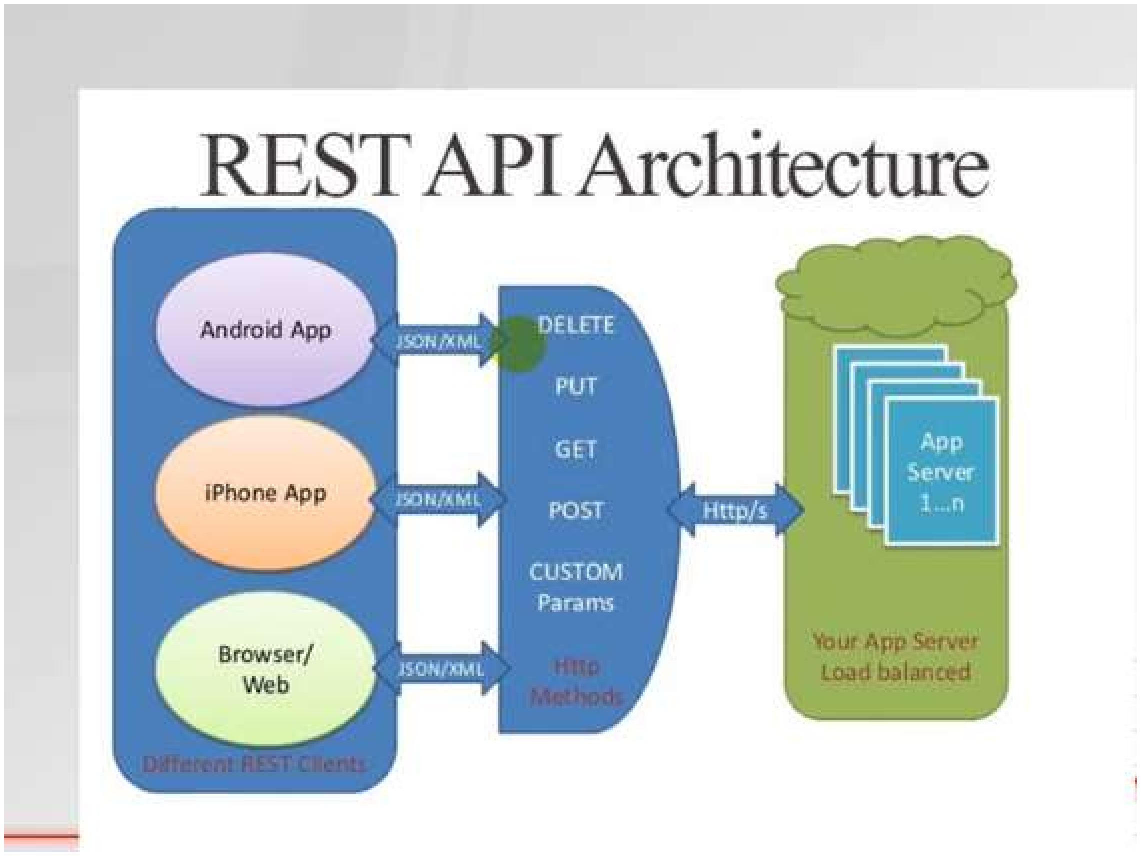
Now insert Document in the Collection



```
// create document or insert

const createDocument = async () ⇒ {
    try{
        const reactPlaylist = new Playlist({
            name : "Node JS",
            ctype : "Back End",
            videos: 50,
            author: "Thapa Technical",
            active: true
        })

        const result = await reactPlaylist.save();
        console.log(result);
    }catch(err){
        console.log(err);
    }
}

createDocument();
```

1. Making API end points using Express

An API is a set of definitions and protocols for building and integrating application software. Representational State Transfer.



Steps:

Initialize Node application → npm init (vs code terminal) …… it will create *package.json* file

Install Express → `npm install --save express`

Regular Update and Restart of server → `npm install --save nodemon`

Need to Install Middleware → `npm install --save body-parser`

Utility Required → `npm install --save underscore`

Step 2: Create a movies.json file and add following code. (Data Base Stored)

```
[{
    "Id": "1",
    "Title": "DDLJ",
    "Director": "Karan Johar"

   },
   {
    "Id": "2",
    "Title": "Sholey",
    "Director": "Sippi"

   },
   {
    "Id": "3",
    "Title": "Fashion",
    "Director": "MB"

   }]
```

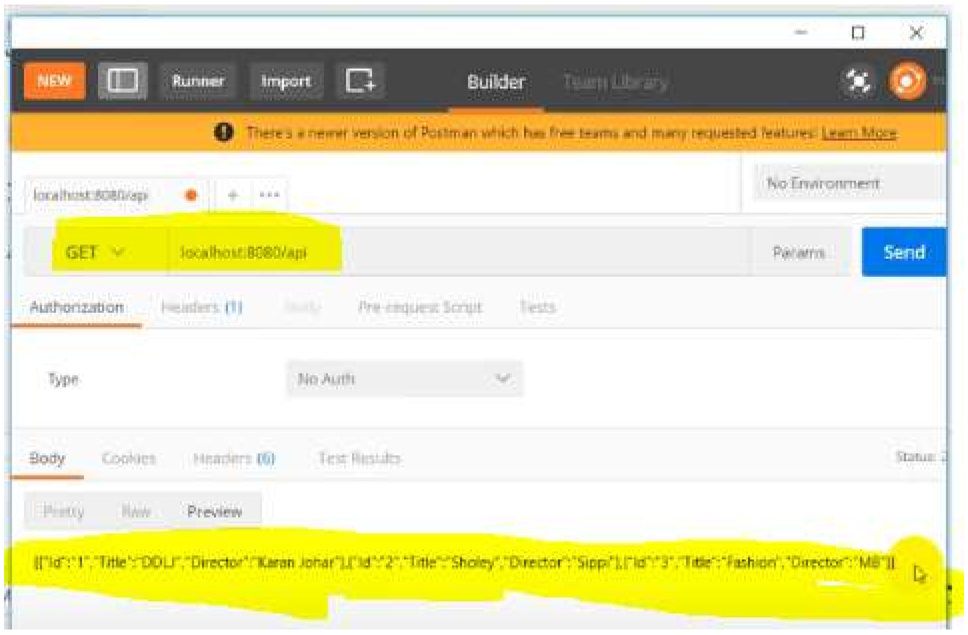
Step 3: Create new file index.js and add following code into index.js.

```
var express    = require('express');          // call express
var app        = express();  // define our app using express
var bodyParser = require('body-parser');
var jsondata= require('./movies.json');
var _und = require('underscore');

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

var port = process.env.PORT || 8080;

var router = express.Router();

router.get('/', function(req, res){
res.json(jsondata);

})
router.post('/postdata', function(req,res){
if(req.body.Id && req.body.Title)
{
jsondata.push(req.body);
res.json(jsondata);
}
else
{
    console.log('please put some values to insert');
```

```
}

})
router.put('/updatedata/:id', function(req,res){
if(req.params.id)
{
_und.each(jsondata , function(elem, index){
if(req.params.id === elem.Id){
    elem.Title = "Hello Brother";

    elem.Director = "xyz";
}

})
res.json(jsondata);
}
else
{
    console.log('Invalid Request');
}

})

router.delete('/deletedata/:id', function(req, res) {
    getindextodelete = -1;
    if(req.params.id){

        _und.each(jsondata, function(elem,index){
if(elem.Id === req.params.id){
    getindextodelete  = index;


}


        })
        if(getindextodelete > -1)
        {
            jsondata.splice(getindextodelete ,1);
        }

    res.json(jsondata);
    }
    else{
        console.log('Please pass body elements with id');
    }
});

app.use('/api', router);
app.listen(port);
```

Step 4: Run Program and test it on postman.
nodemon index.js

```
Test it on Postman
```



GET localhost:8080/api

Type: No Auth

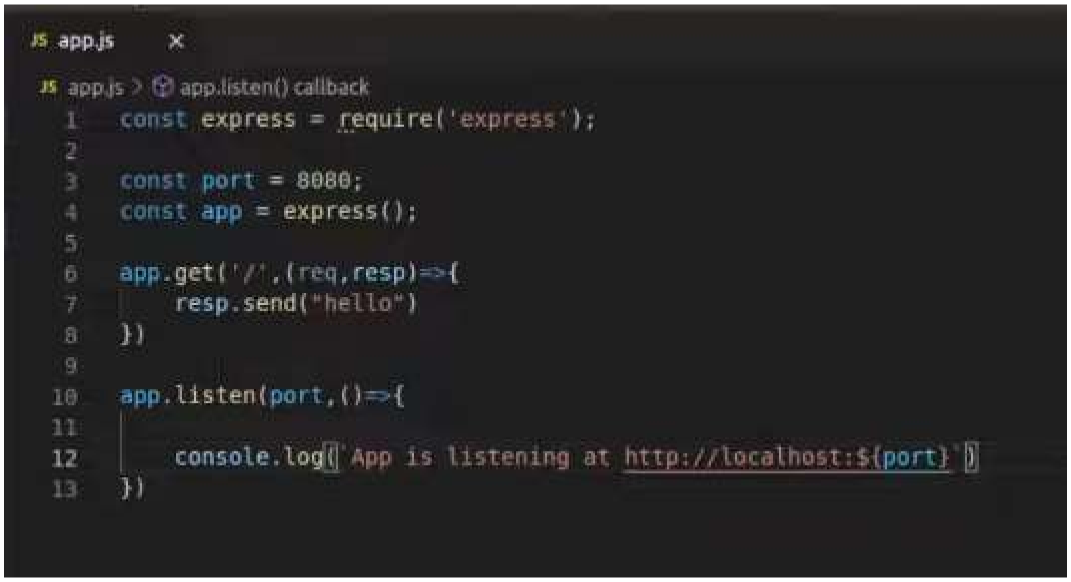
```
[{"id":"1","Title":"DDLJ","Director":"Karan Johar"},{"id":"2","Title":"Sholey","Director":"Sippi"},{"id":"3","Title":"Fashion","Director":"MB"}]
```

2. Writing tests using mocha and chai

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing *simple* and *fun*. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. Hosted on GitHub.

Chai is a BDD / TDD assertion library for node and the bro wser that can be delightfully paired with any javascript testi ng framework.

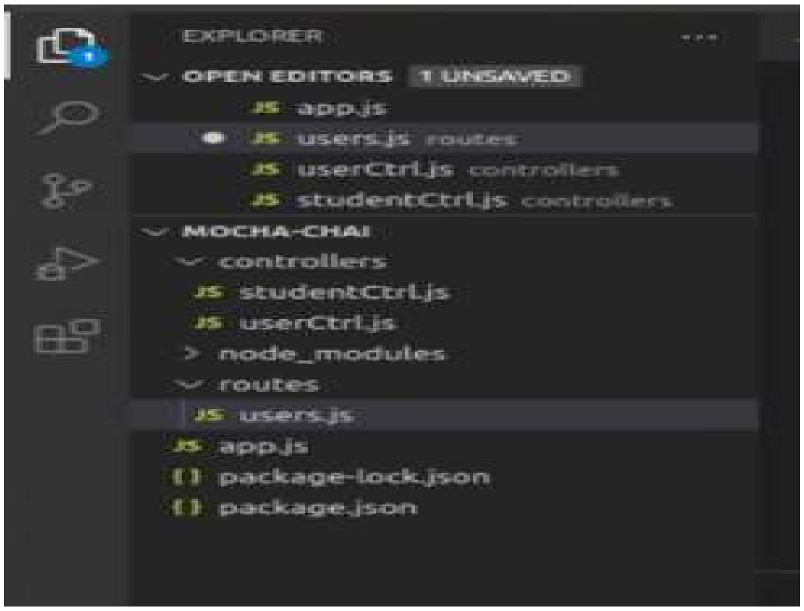
1. npm init –y (for package file)
2. npm i express nodemon



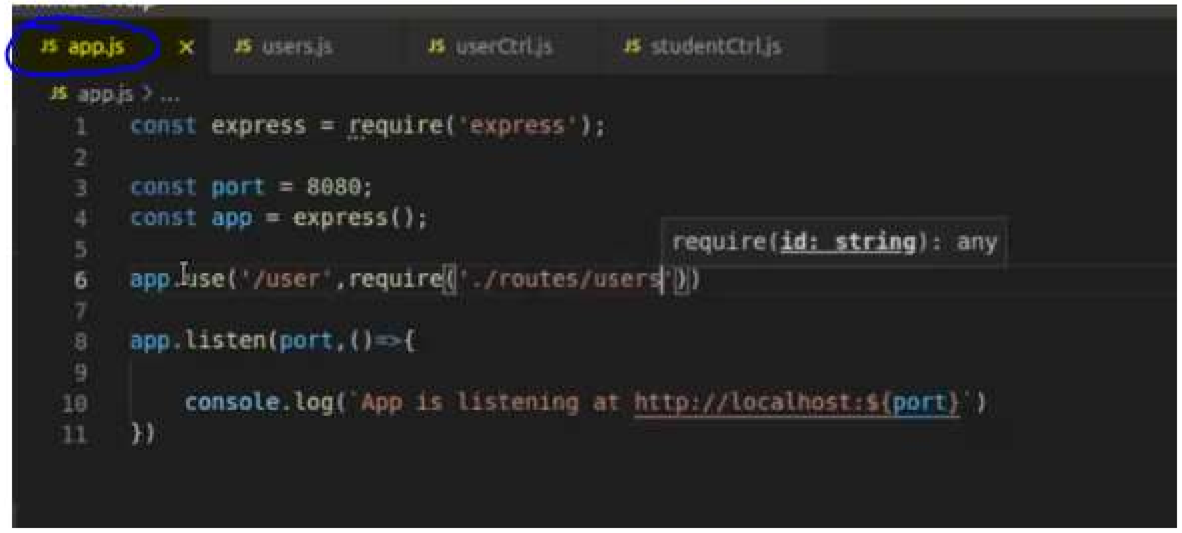
```
const express = require('express');

const port = 8080;
const app = express();

app.get('/',(req,resp)=>{
    resp.send("hello")
})

app.listen(port,()=>{

    console.log(`App is listening at http://localhost:${port}`)
})
```

nodemon app.js RUN on browser→ *localhost:8080*

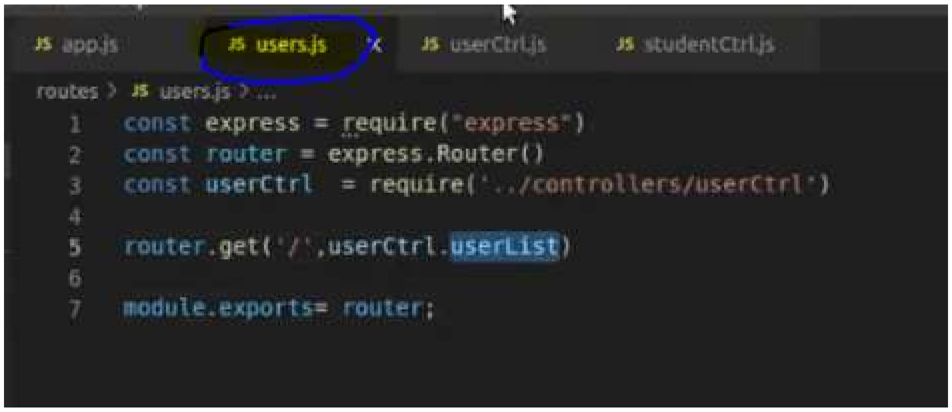
FILES Required for Project Setup:



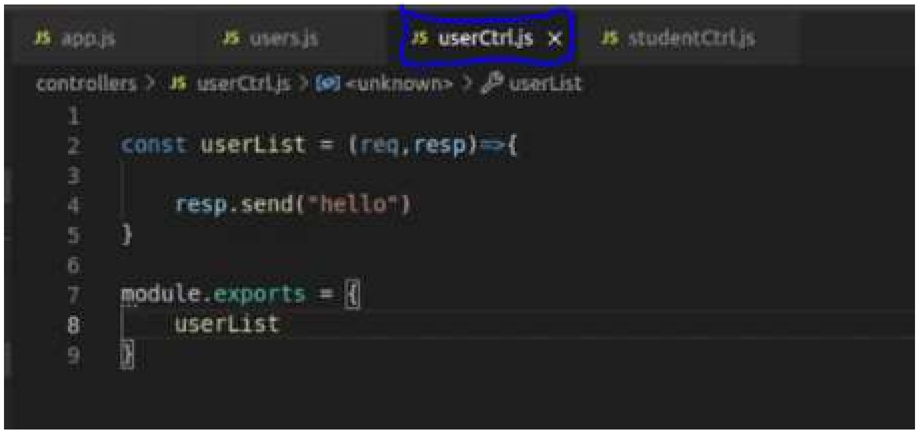
```
MOCHA-CHAI
  controllers
    studentCtrl.js
    userCtrl.js
  node_modules
  routes
    users.js
  app.js
  package-lock.json
  package.json
```



```
const express = require('express');

const port = 8080;
const app = express();

app.use('/user',require('./routes/users'))

app.listen(port,()=>{

    console.log(`App is listening at http://localhost:${port}`)
})
```



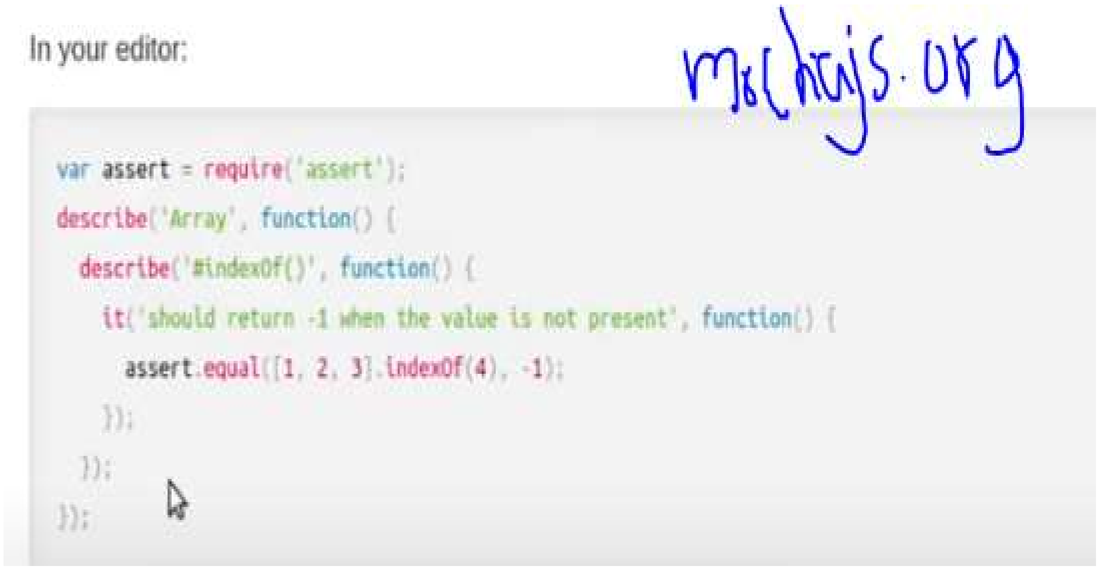
```
const express = require("express")
const router = express.Router()
const userCtrl  = require('../controllers/userCtrl')

router.get('/',userCtrl.userList)

module.exports= router;
```



```

const userList = (req,resp)=>{

    resp.send("hello")
}

module.exports = {
    userList
}
```

(Now we have to check it on Browser → localhost: 8080) … will not run because path set for User. So in Browser → localhost: 8080/user

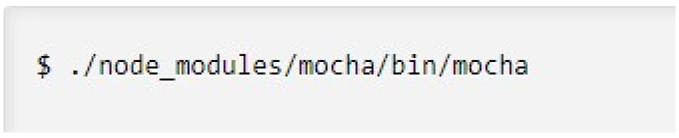
Now install Mocha → npm i mocha - -save-dev
Create folder test and in that folder create file first.spec.js and copy following code


In your editor:

mochajs.org

```
var assert = require('assert');
describe('Array', function() {
  describe('#indexOf()', function() {
    it('should return -1 when the value is not present', function() {
      assert.equal([1, 2, 3].indexOf(4), -1);
    });
  });
});
```

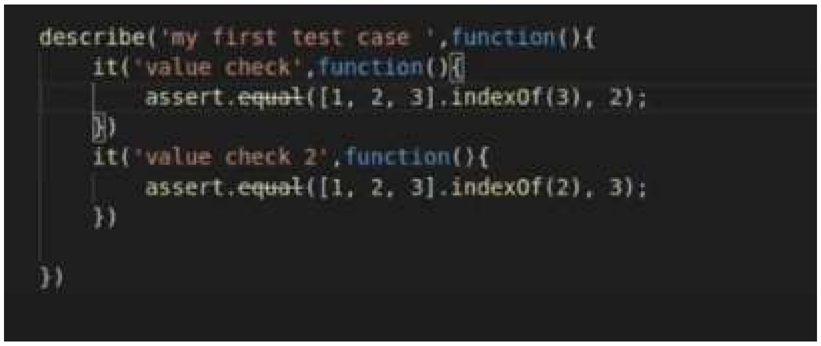
*Following command to Test by using **mocha** in terminal →*



```
$ ./node_modules/mocha/bin/mocha
```

*To avoid long command path : Go to **package.json** → "test" : "mocha"*

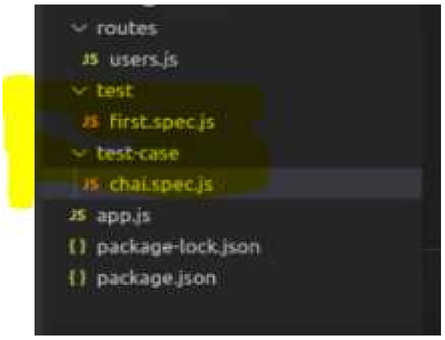
*Added Two More Test cases in **first.spec.js***



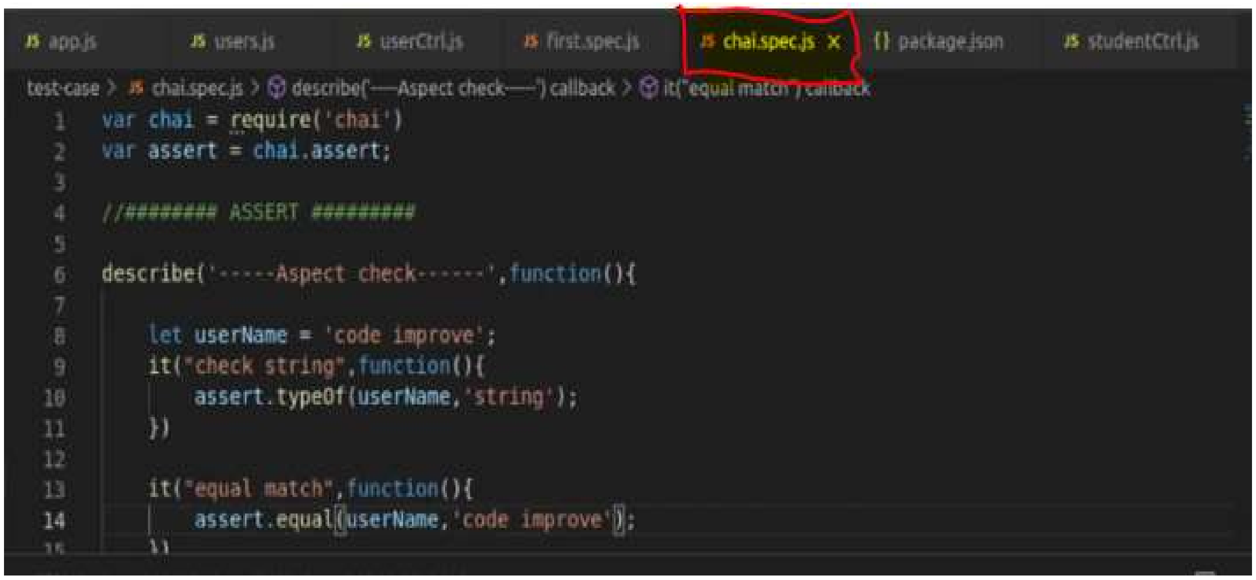
```
describe('my first test case ',function(){
    it('value check',function(){
        assert.equal([1, 2, 3].indexOf(3), 2);
    })
    it('value check 2',function(){
        assert.equal([1, 2, 3].indexOf(2), 3);
    })

})
```

Now for chai :

npm I chai - -save-dev

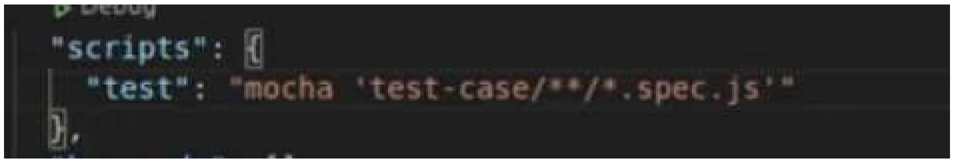


```
routes
  users.js
test
  First.spec.js
test-case
  chai.spec.js
app.js
package-lock.json
package.json
```


app.js
users.js
userCtrl.js
first.spec.js
chai.spec.js
package.json
studentCtrl.js
test-case > chai.spec.js > describe('-----Aspect check------') callback > it("equal match") callback

```
var chai = require('chai')
var assert = chai.assert;

//######## ASSERT #########

describe('-----Aspect check------',function(){

    let userName = 'code improve';
    it("check string",function(){
        assert.typeOf(userName,'string');
    })

    it("equal match",function(){
        assert.equal(userName,'code improve');
    })
```

While using Chai, Changes in *package.json*



```
"scripts": {
  "test": "mocha 'test-case/**/*.spec.js'"
},
```

For Testing → npm test