Module 5: REST APIs And GraphQL

Demo Document 1

edureka!

edureka!

© Brain4ce Education Solutions Pvt. Ltd.

Working of mongodb API

Step 1: Go inside your application folder and generate package.json file

```
code — npm TERM_PROGRAM=Apple_Terminal SHELL=/bin/bash — 81×22
Last login: Mon May 13 08:16:51 on ttys001
Avyaans-MacBook-Pro:~ avi$ cd Desktop/folder/EdurekaApp/module5/code/
Avyaans-MacBook-Pro:code avi$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (code) apiwithmongo
version: (1.0.0)
description: api using mongodb
entry point: (index.js)
test command:
git repository:
keywords: node ejs mongo
author: Edureka
license: (ISC)
```

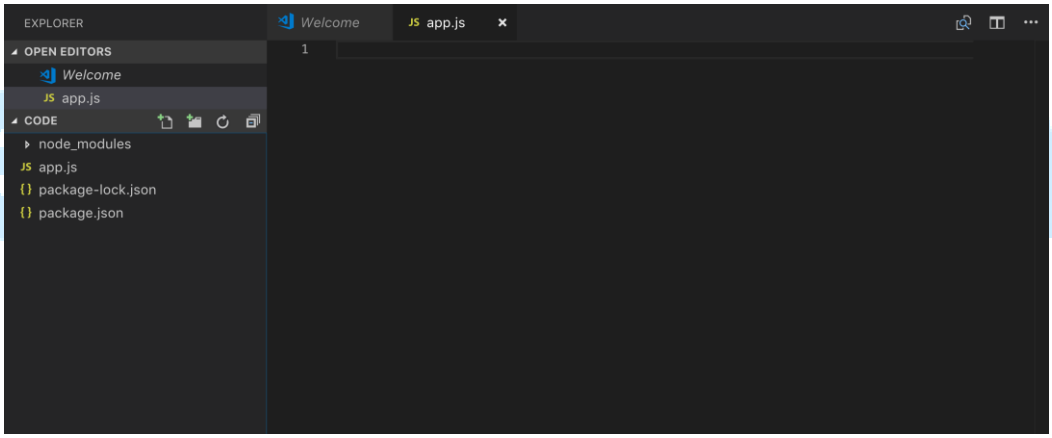
Step 2: After adding all details type “yes” and it will generate package.json in folder

```
code — -bash — 81×22
license: (ISC)
About to write to /Users/avi/Desktop/folder/EdurekaApp/module5/code/package.json:

{
  "name": "apiwithmongo",
  "version": "1.0.0",
  "description": "api using mongodb",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "node",
    "ejs",
    "mongo"
  ],
  "author": "Edureka",
  "license": "ISC"
}


Is this OK? (yes) yes
```

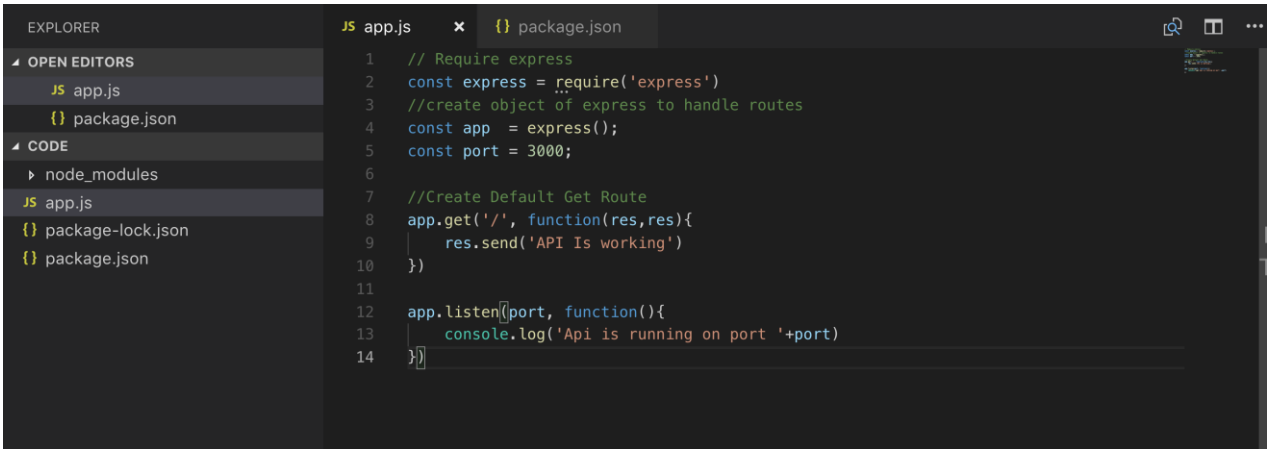
Step 3: Install “MongoDB” to connect with the database, “express” for server and “body-parser” to post data from form.

```
code — -bash — 81×22
Avyaans-MacBook-Pro:code avi$ npm install express mongodb body-parser
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN apiwithmongo@1.0.0 No repository field.

+ body-parser@1.19.0
+ mongodb@3.2.4
+ express@4.16.4
added 70 packages from 43 contributors and audited 164 packages in 7.897s
found 0 vulnerabilities
```

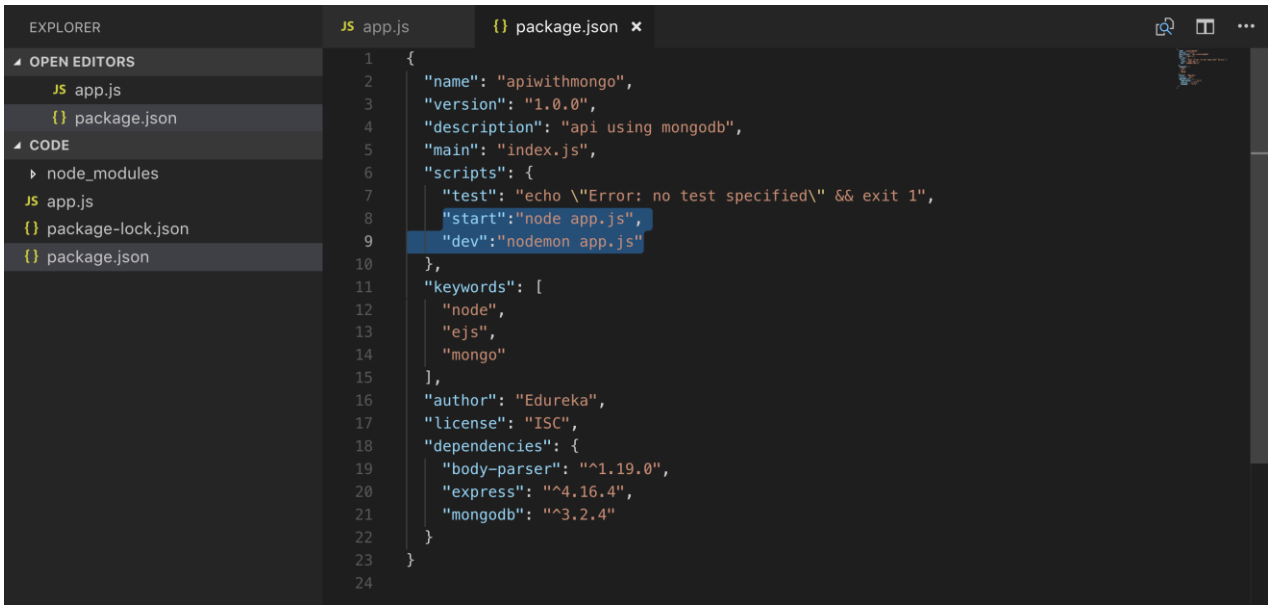
Step 4: Create a folder structure



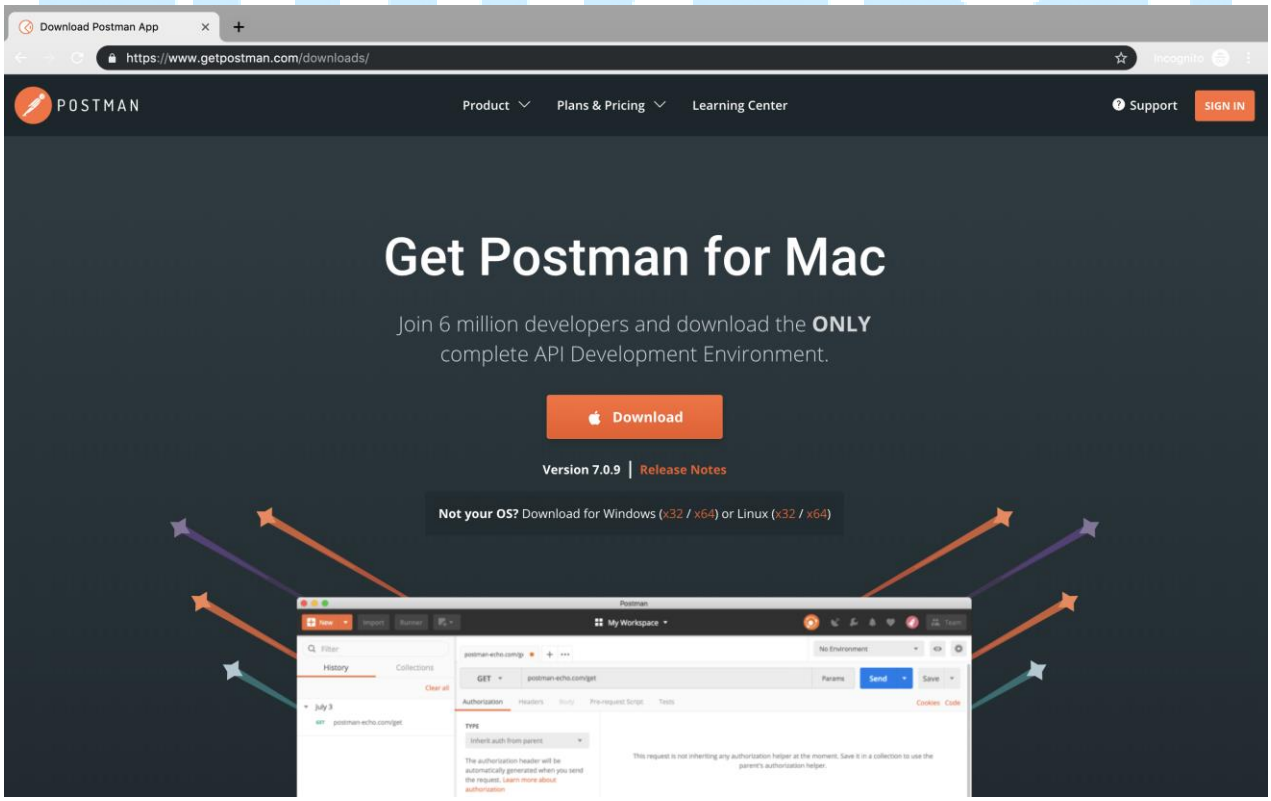
Step 5: Create basic express server with default get route.



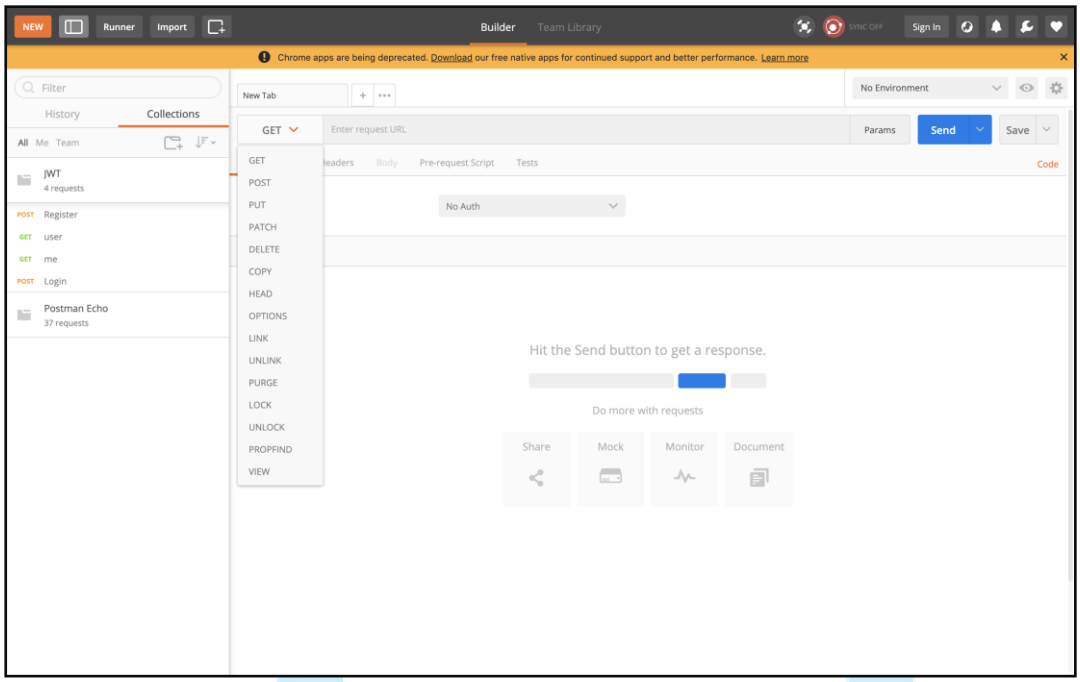
Step 6: Add command start application in both development and production mode



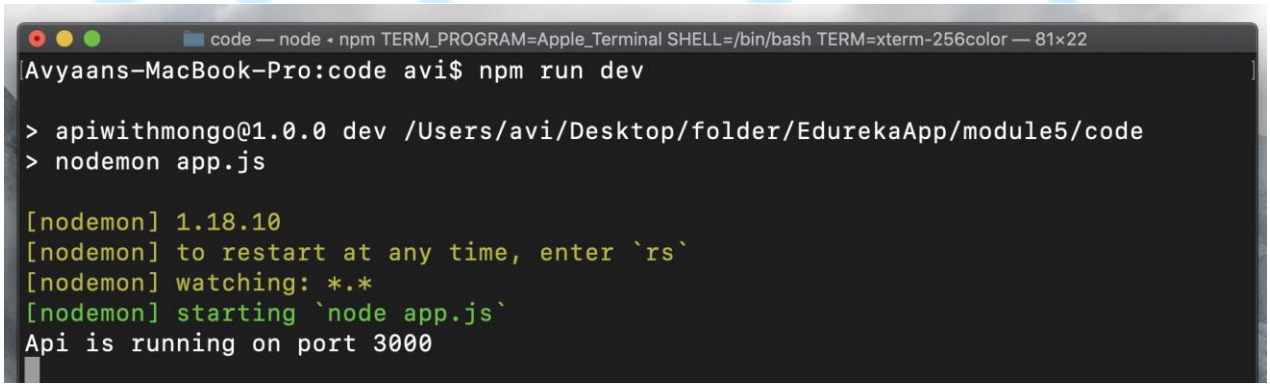
Step 7: Install Postman either as chrome extension or App to test end point of API.



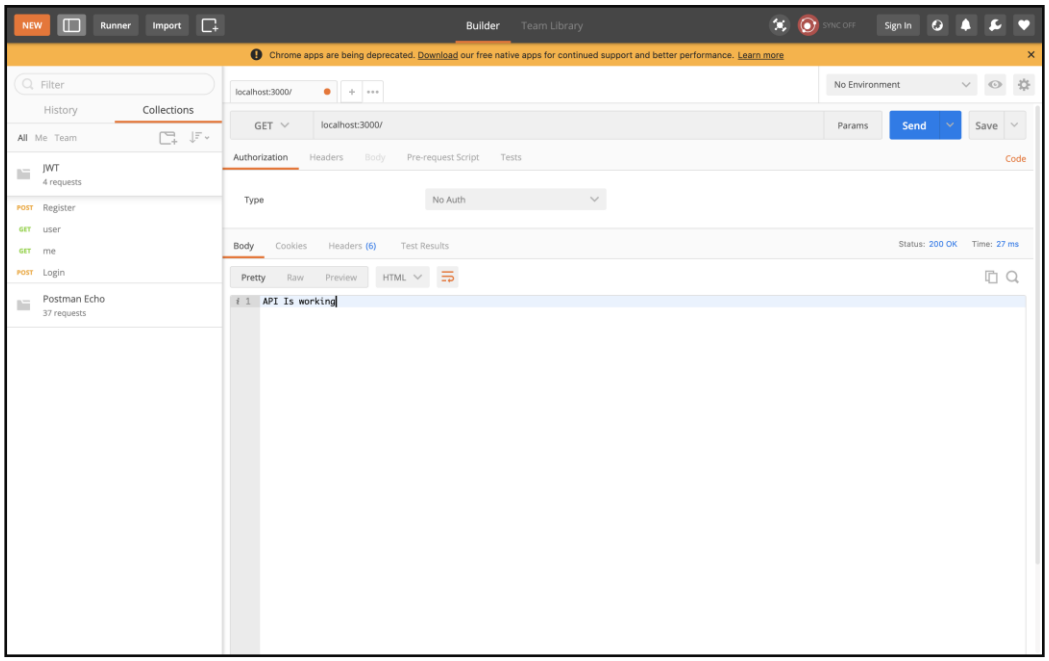
Step 8: In Postman we can test all GET, POST, PUT and DELETE api.



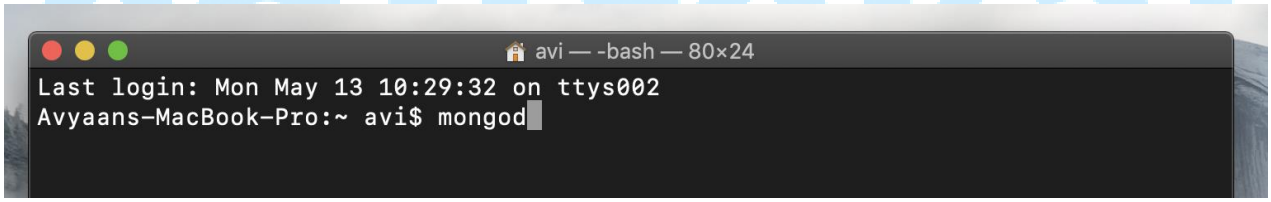
Step 9: Start App in development mode using "npm run dev"



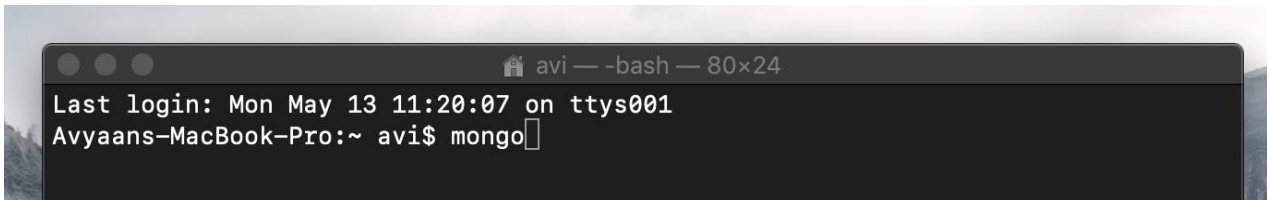
Step 10: Test the API endpoint using postman



Step 11: Run mongo DB server as we have so that we can connect Api with mongoDB



Step 12: In Another Command Prompt run mongo client to run database query on console



Step 13: List all database using “show dbs” in mongo client

```
avi — mongo — 80×24
interfaces. If this behavior is desired, start the
2019-05-13T11:20:24.819+0100 I CONTROL  [initandlisten] **          server with
--bind_ip 127.0.0.1 to disable this warning.
2019-05-13T11:20:24.819+0100 I CONTROL  [initandlisten]
> show dbs
MEANStackDB      0.000GB
admin            0.000GB
apr_dashboard    0.000GB
apr_node_oo      0.000GB
aprlogin         0.000GB
classdatabase    0.000GB
classpractice    0.000GB
config           0.000GB
curd             0.000GB
dashboard        0.000GB
graphql          0.000GB
local            0.000GB
marchNode        0.000GB
march_dashboard  0.000GB
marchang         0.000GB
ts_crud          0.000GB
user_crud        0.000GB
userlogin        0.000GB
>
```

Step 14: We are using Class practice db throughout the demo

```
avi — mongo — 80×24
MEANStackDB      0.000GB
admin            0.000GB
apr_dashboard    0.000GB
apr_node_oo      0.000GB
aprlogin         0.000GB
classdatabase    0.000GB
classpractice    0.000GB
config           0.000GB
curd             0.000GB
dashboard        0.000GB
graphql          0.000GB
local            0.000GB
marchNode        0.000GB
march_dashboard  0.000GB
marchang         0.000GB
ts_crud          0.000GB
user_crud        0.000GB
userlogin        0.000GB
> use classpractice
switched to db classpractice
> show collections
movies
mydata
>
```

Step 15: Connect Nodejs App with Mongodb running on local system to run CRUD operations

```
EXPLORER                    app.js   package.json
OPEN EDITORS
  app.js
  package.json
CODE
  node_modules
  app.js
  package-lock.json
  package.json

// Require express
const express = require('express')
//create object of express to handle routes
const app  = express();
const port = 3000;
const MongoClient = require('mongodb').MongoClient;
let db;
//pass mongodb url
const mongourl = 'mongodb://127.0.0.1:27017/'
const col_name = 'userlist';

//Create Default Get Route
app.get('/', function(res,res){
    res.send('API Is working')
})

MongoClient.connect(mongourl,{ useNewUrlParser: true }, function(err,client) {
    if(err) throw err;
    //database name here
    db = client.db('userdata')
    app.listen(port, ()=> {
        console.log('Api is running on port '+port)
    })
})
```

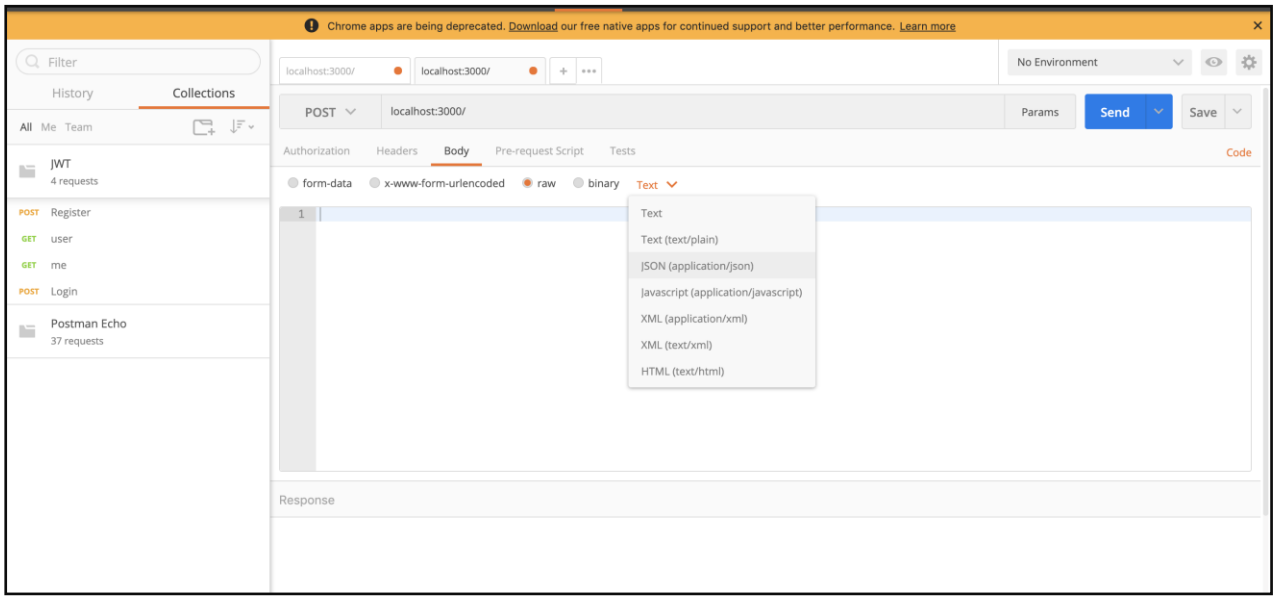
Step 16: Add body parser to grab data from the form we need to add bodyParser as middleware and add add url-encoded with json

```
EXPLORER                    app.js   package.json
OPEN EDITORS
  app.js
  package.json
CODE
  node_modules
  app.js
  package-lock.json
  package.json

// Require express
const express = require('express')
//create object of express to handle routes
const app  = express();
const port = 3000;
const MongoClient = require('mongodb').MongoClient;
// to parse data from body
const bodyParser = require('body-parser');
let db;
//pass mongodb url
const mongourl = 'mongodb://127.0.0.1:27017/'
const col_name = 'userlist';
// Adding encoding using bodyparser
app.use(bodyParser.urlencoded({extended:true}));
app.use(bodyParser.json());


//Create Default Get Route
app.get('/', function(res,res){
    res.send('API Is working')
})


MongoClient.connect(mongourl,{ useNewUrlParser: true }, function(err,client) {
    if(err) throw err;
    //database name here
    db = client.db('userdata')
    app.listen(port, ()=> {
        console.log('Api is running on port '+port)
    })
})

OUTLINE
master*  7↓ 9↑  0 0                Ln 17, Col 1  Spaces: 4  UTF-8  LF  JavaScript
```

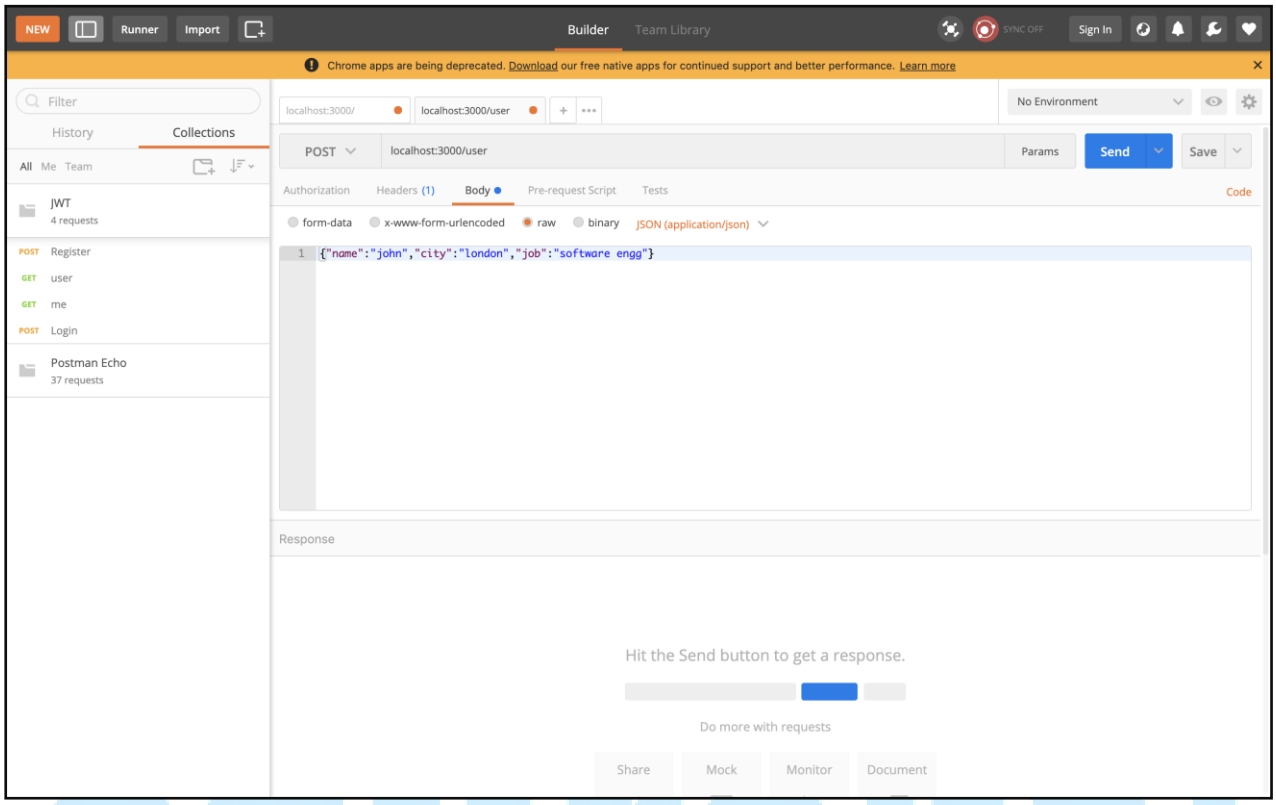
Step 17: Add post endpoint to add data in mongodb as specified in line 11and add the need endpoint on line 18

```
1  // Require express
2  const express = require('express')
3  //create object of express to handle routes
4  const app  = express();
5  const port = 3000;
6  const MongoClient = require('mongodb').MongoClient;
7  // to parse data from body
8  const bodyParser = require('body-parser');
9  let db;
10 //pass mongodb url
11 const mongourl = 'mongodb://127.0.0.1:27017/'
12 const col_name = 'userlist';
13 // Adding encoding using bodyparser
14 app.use(bodyParser.urlencoded({extended:true}));
15 app.use(bodyParser.json());
16
17 //route to add user
18 app.post('/user', (req,res) => {
19     db.collection(col_name)
20         // getting data in body for post call
21         .insert(req.body, (err,result) => {
22             if(err) throw err;
23             res.send('data.inserted');
24         })
25 })
26
27 //Create Default Get Route
28 app.get('/', function(res,res){
29     res.send('API Is working')
30 })
31
```

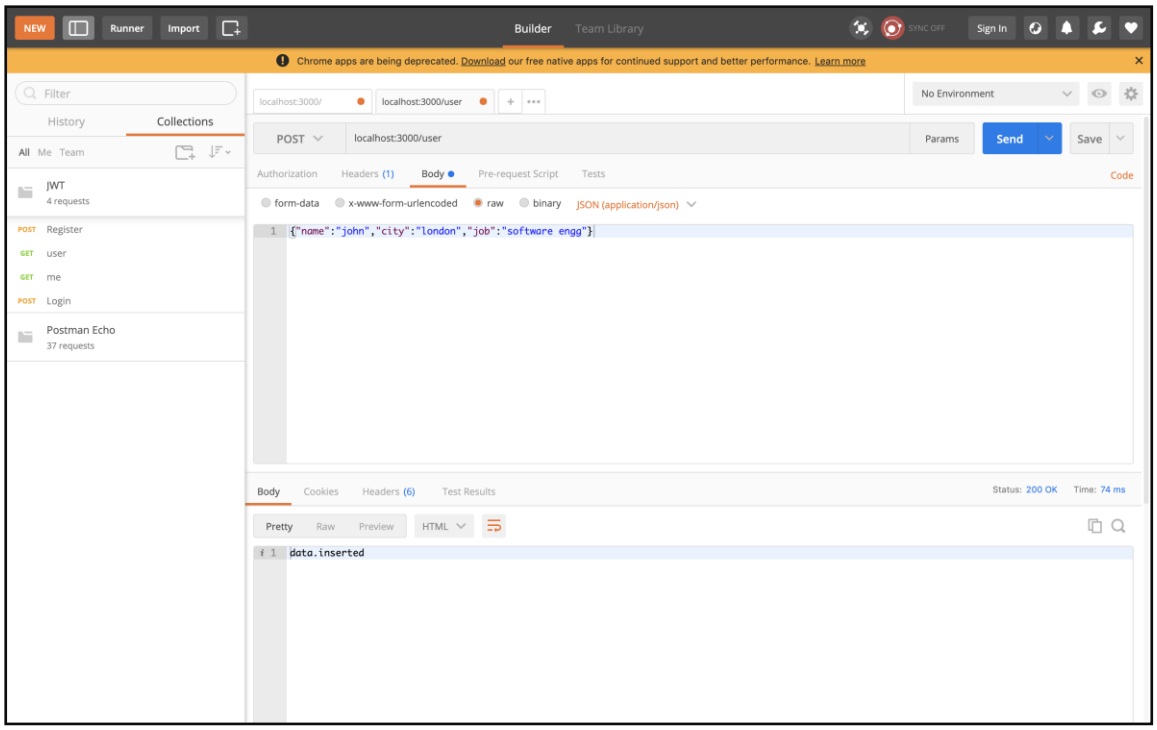
Step 18: With postman to post, select the method as POST



Step 19: In Post call we need to send data in post body as json



Step 20: After clicking send on successful insertion data will be inserted in DB and we will get



Step 21: We can verify insert from mongodb console and we can see data is inserted successful in DB

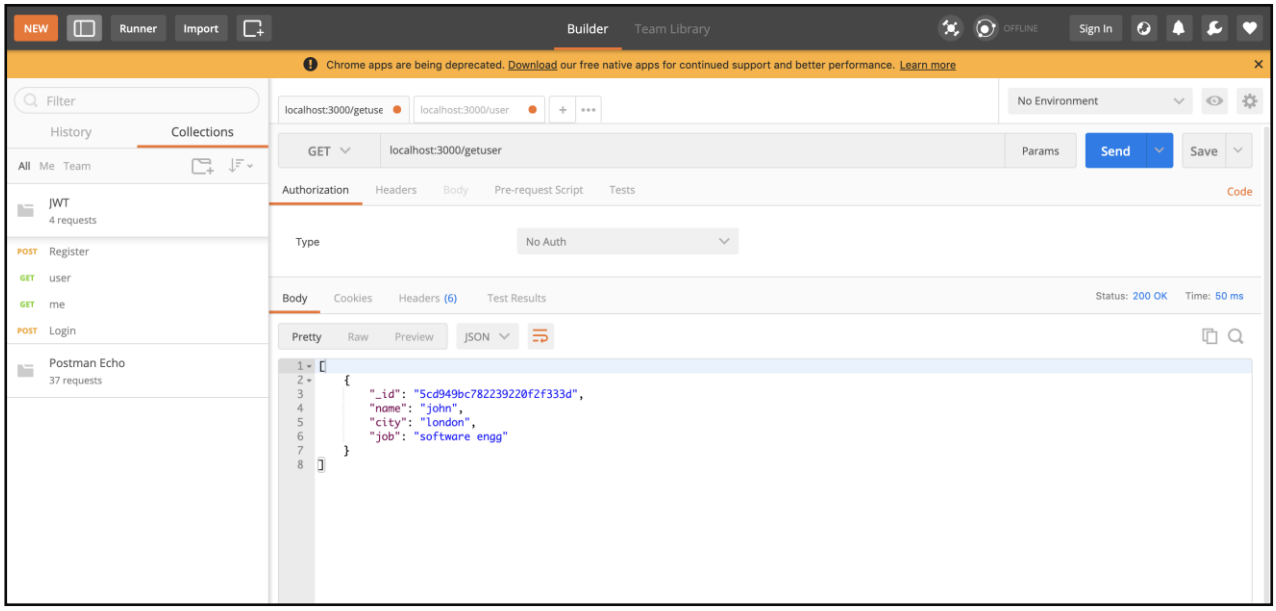
```
> use userdata
switched to db userdata
> show collections
userlist
> db.userlist.find().pretty()
{
        "_id" : ObjectId("5cd949bc782239220f2f333d"),
        "name" : "john",
        "city" : "london",
        "job" : "software engg"
}
>
```

Step 22: Add Get Route to get data from database to fetch data from DB '/getUser'

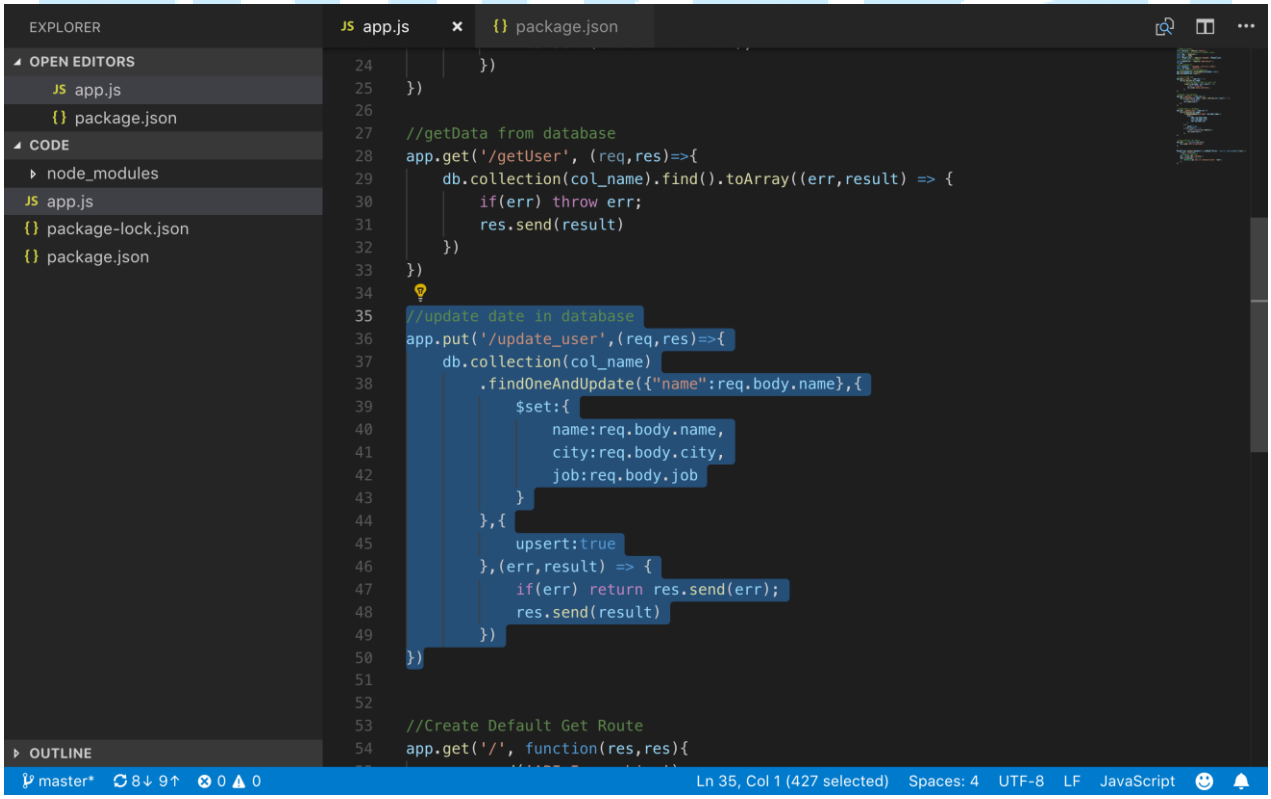
```
EXPLORER
OPEN EDITORS
    app.js
    package.json
CODE
    node_modules
    app.js
    package-lock.json
    package.json
OUTLINE
```

```
let db;
//pass mongodb url
const mongourl = 'mongodb://127.0.0.1:27017/'
const col_name = 'userlist';
// Adding encoding using bodyparser
app.use(bodyParser.urlencoded({extended:true}));
app.use(bodyParser.json());

//route to add user
app.post('/user', (req,res) => {
    db.collection(col_name)
        // getting data in body for post call
        .insert(req.body, (err,result) => {
            if(err) throw err;
            res.send('data.inserted');
        })
})

//getData from database
app.get('/getUser', (req,res)=>{
    db.collection(col_name).find().toArray((err,result) => {
        if(err) throw err;
        res.send(result)
    })
})

//Create Default Get Route
app.get('/', function(res,res){
    res.send('API Is working')
})
```

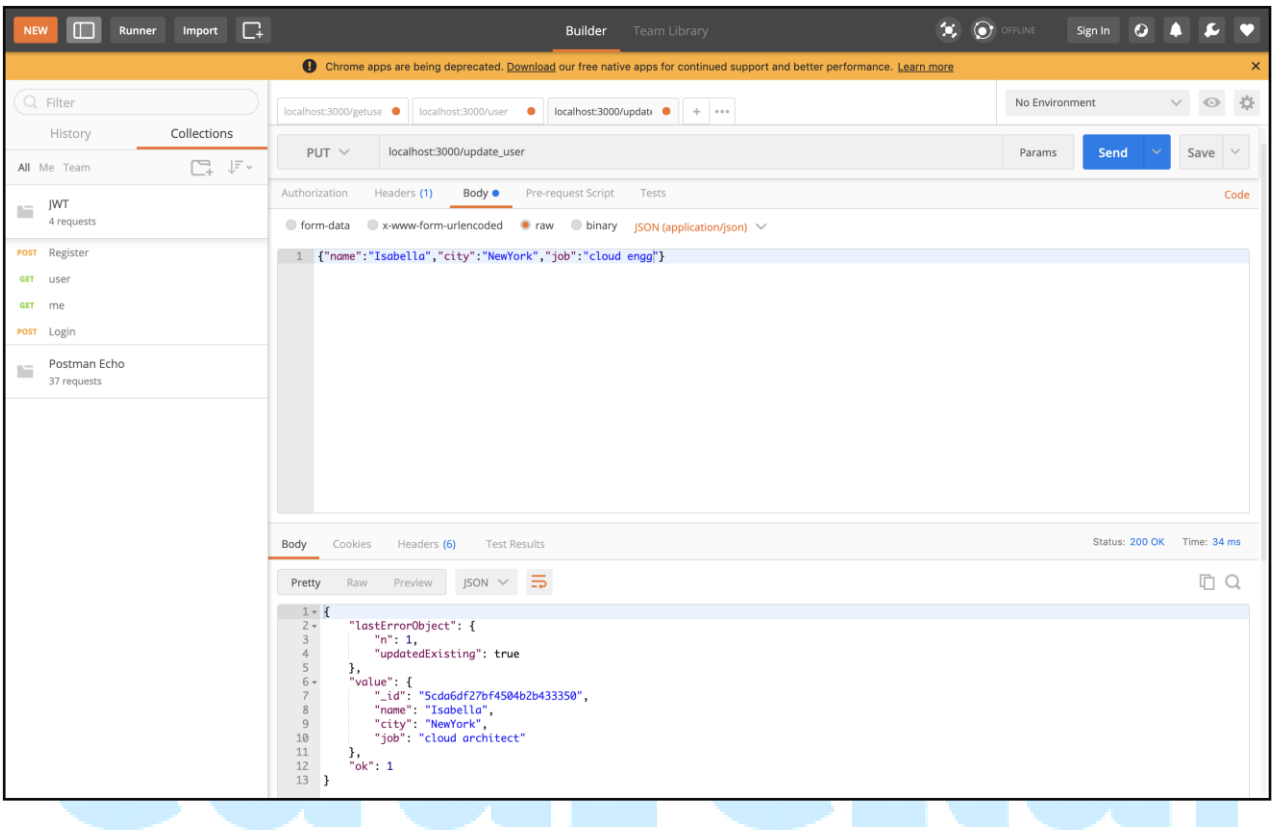
Step 23: Test '/getuser' using Postman on clicking Send we get json response, the one we inserted using POST request



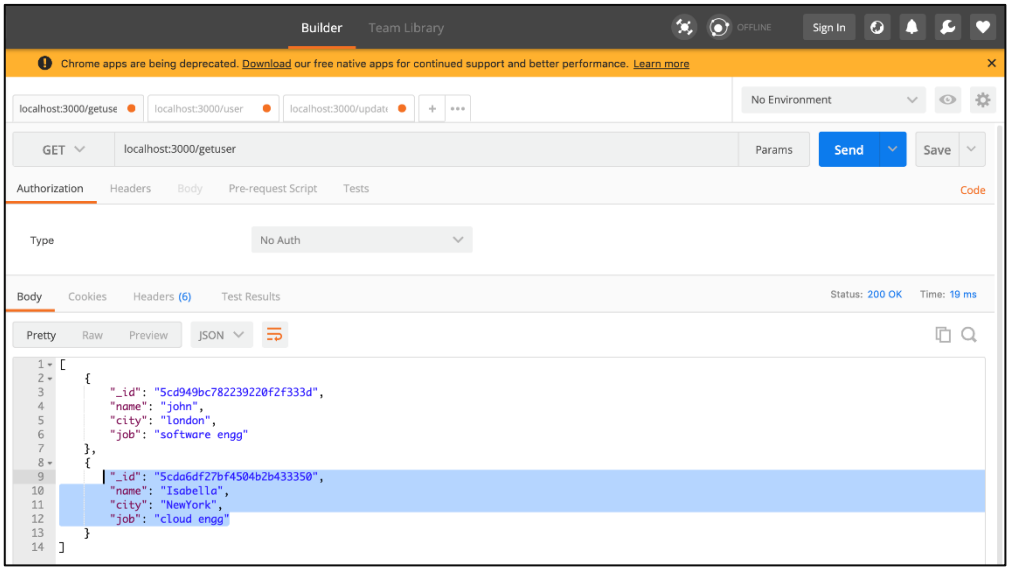
Step 24: To update data we need to make PUT request and use mongodb update query to update data in records



Step 25: Using the PUT request from postman we can send updated data and in response we will get updated value with _object key



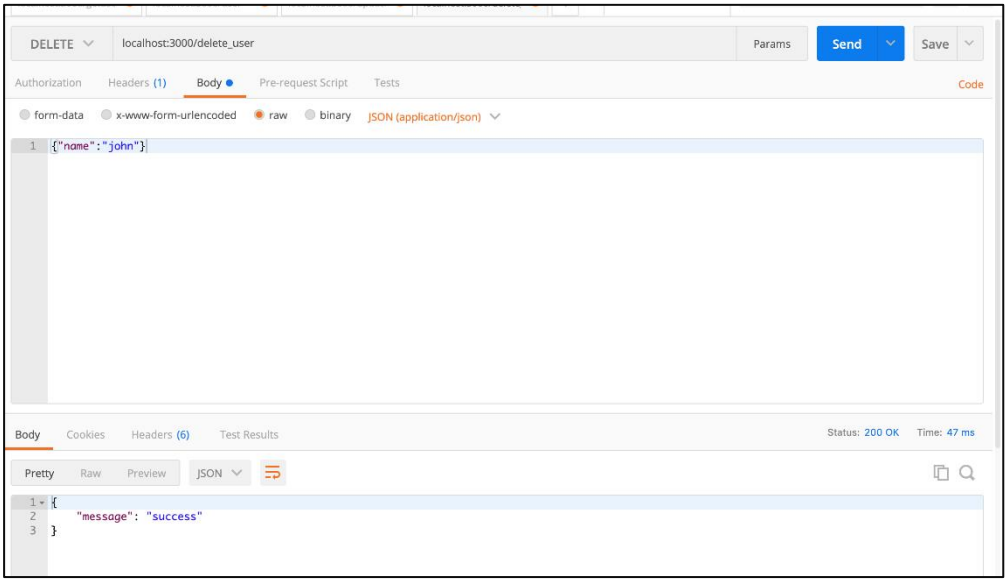
Step 26: You see the updated data while typing '/getuser'



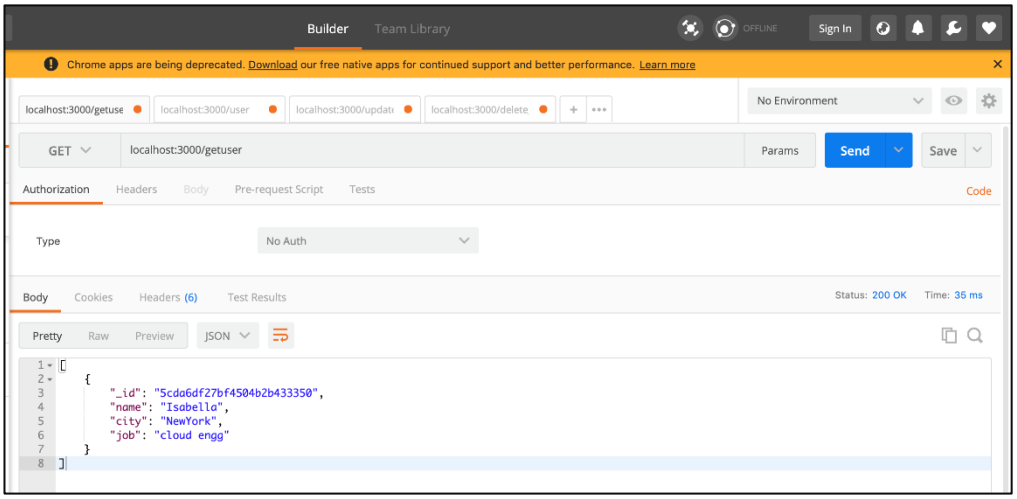
Step 27: To delete the data we need to make delete request and use mongodb delete query to delete the data in records

```
EXPLORER
OPEN EDITORS
    app.js
    package.json
CODE
    node_modules
    app.js
    package-lock.json
    package.json
OUTLINE

app.js   package.json

    })
})

//update date in database
app.put('/update_user',(req,res)=>{
    db.collection(col_name)
        .findOneAndUpdate({"name":req.body.name},{
            $set:{
                name:req.body.name,
                city:req.body.city,
                job:req.body.job
            }
        },{
            upsert:true
        },(err,result) => {
            if(err) return res.send(err);
            res.send(result)
        })
})

//Delete date in database
app.delete('/delete_user',(req,res) => {
    db.collection(col_name).findOneAndDelete({
        "name":req.body.name
    },(err,result) => {
        if (err) return res.send(500,err)
        res.send({message: 'success'})
    })
})

//Create Default Get Route
```

Step 28: Using the DELETE request from postman



Step 29: Verify whether it is deleted through '/getuser'



Thus, we have performed CRUD operations through the mongodb API