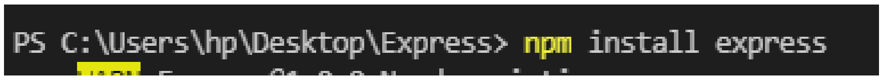
CRUD operations with Express Js

By- Shubhankar Shankar
Ashish Alok
Kunal Singh Chauhan

Step to follow:

Step-1 First we need to create a packet.json file in the folder.

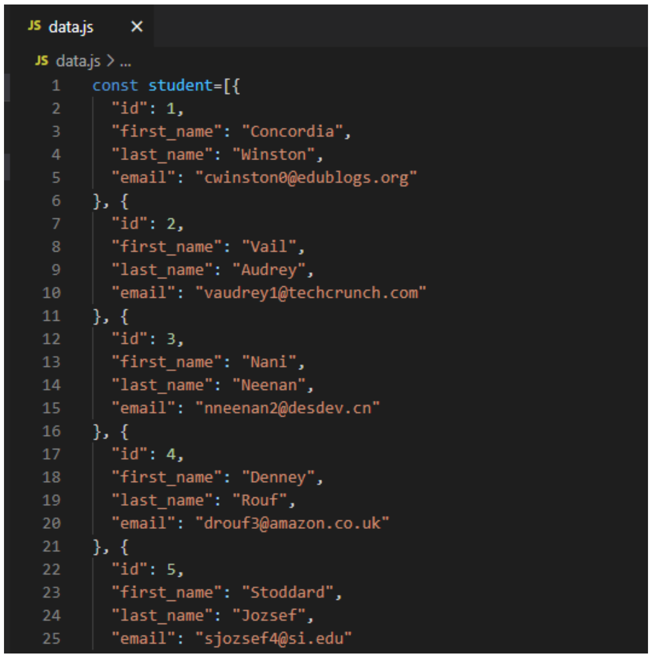
Step-2 Then we have to install Express Js using the command.



```
PS C:\Users\hp\Desktop\Express> npm install express
```

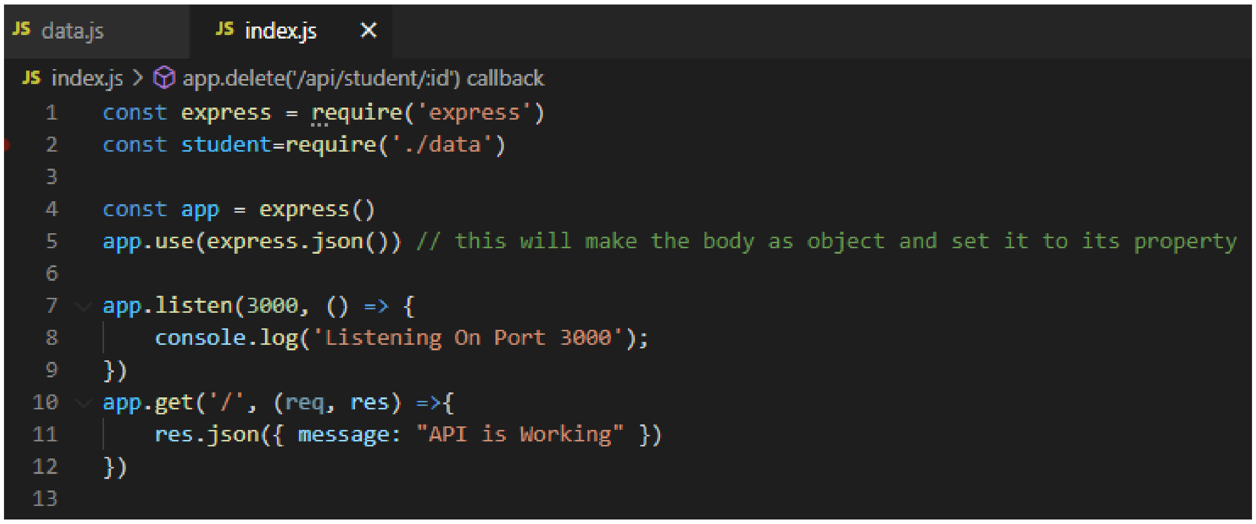
Step-3 Then we will create an index.js file in which we are going to write our javascript code for the operations.

Step-4 We will create one more file (data.js) in which we will store a dummy data so that we can use it as the API and perform operations on that.



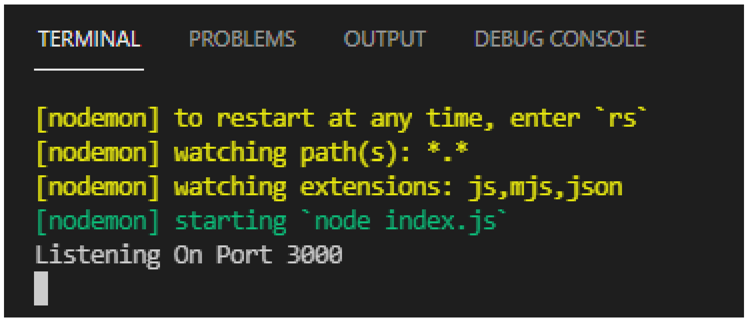
```
const student=[{
  "id": 1,
  "first_name": "Concordia",
  "last_name": "Winston",
  "email": "cwinston0@edublogs.org"
}, {
  "id": 2,
  "first_name": "Vail",
  "last_name": "Audrey",
  "email": "vaudrey1@techcrunch.com"
}, {
  "id": 3,
  "first_name": "Nani",
  "last_name": "Neenan",
  "email": "nneenan2@desdev.cn"
}, {
  "id": 4,
  "first_name": "Denney",
  "last_name": "Rouf",
  "email": "drouf3@amazon.co.uk"
}, {
  "id": 5,
  "first_name": "Stoddard",
  "last_name": "Jozsef",
  "email": "sjozsef4@si.edu"
```

Step-5 Then in the index.js file we have to import express and the data file so that we can use it.Then we will run it on a local host.

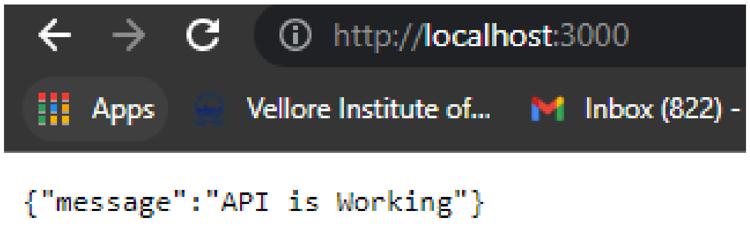


```
const express = require('express')
const student=require('./data')

const app = express()
app.use(express.json()) // this will make the body as object and set it to its property

app.listen(3000, () => {
    console.log('Listening On Port 3000');
})
app.get('/', (req, res) =>{
    res.json({ message: "API is Working" })
})
```

Output:

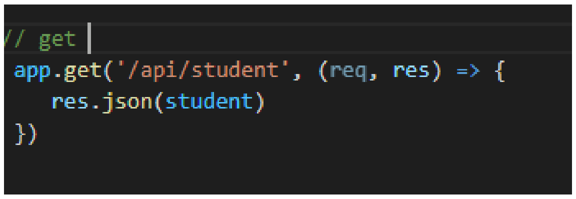


```
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Listening On Port 3000
```



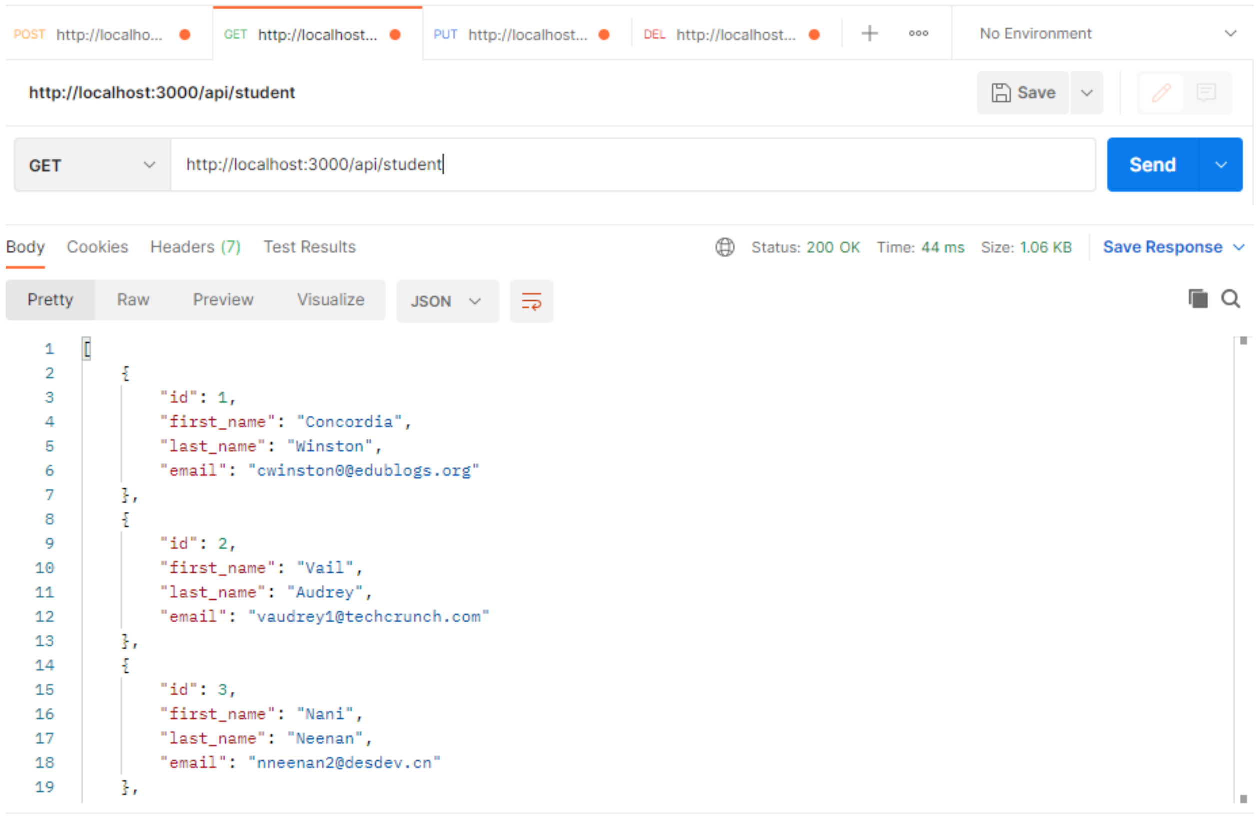
```
{"message":"API is Working"}
```

Step-6 **GET Operation**-Then we will write code for GET operation. By using this we will get all the data from the server.



```
// get
app.get('/api/student', (req, res) => {
    res.json(student)
})
```

For the output of this code we will use Postman to hit the API and get the data.



http://localhost:3000/api/student

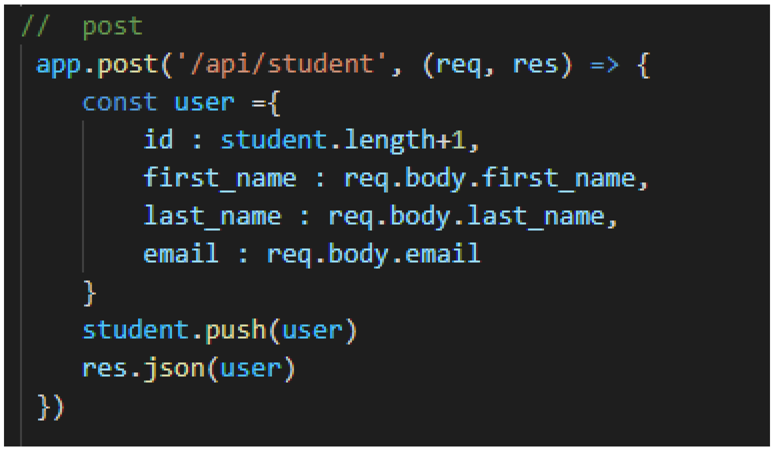
GET http://localhost:3000/api/student

Status: 200 OK Time: 44 ms Size: 1.06 KB

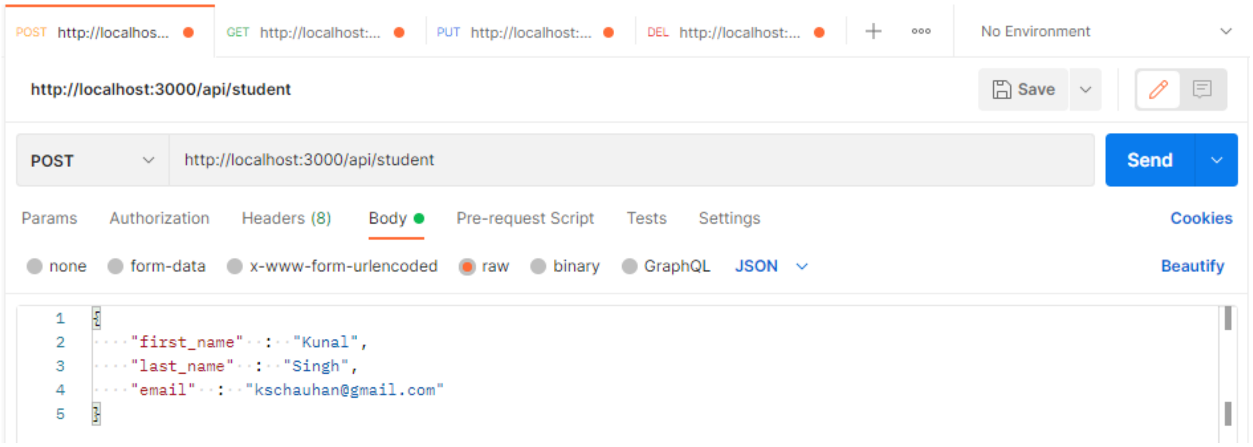
```
[
    {
        "id": 1,
        "first_name": "Concordia",
        "last_name": "Winston",
        "email": "cwinston0@edublogs.org"
    },
    {
        "id": 2,
        "first_name": "Vail",
        "last_name": "Audrey",
        "email": "vaudrey1@techcrunch.com"
    },
    {
        "id": 3,
        "first_name": "Nani",
        "last_name": "Neenan",
        "email": "nneenan2@desdev.cn"
    },
```

Here we can see that we are getting the data from the api by sending a GET request.

Step-7 **POST Operation:** Now we will do the POST operation using the following code we will post the data .


```
//  post
app.post('/api/student', (req, res) => {
   const user ={
       id : student.length+1,
       first_name : req.body.first_name,
       last_name : req.body.last_name,
       email : req.body.email
   }
   student.push(user)
   res.json(user)
})
```

Output:

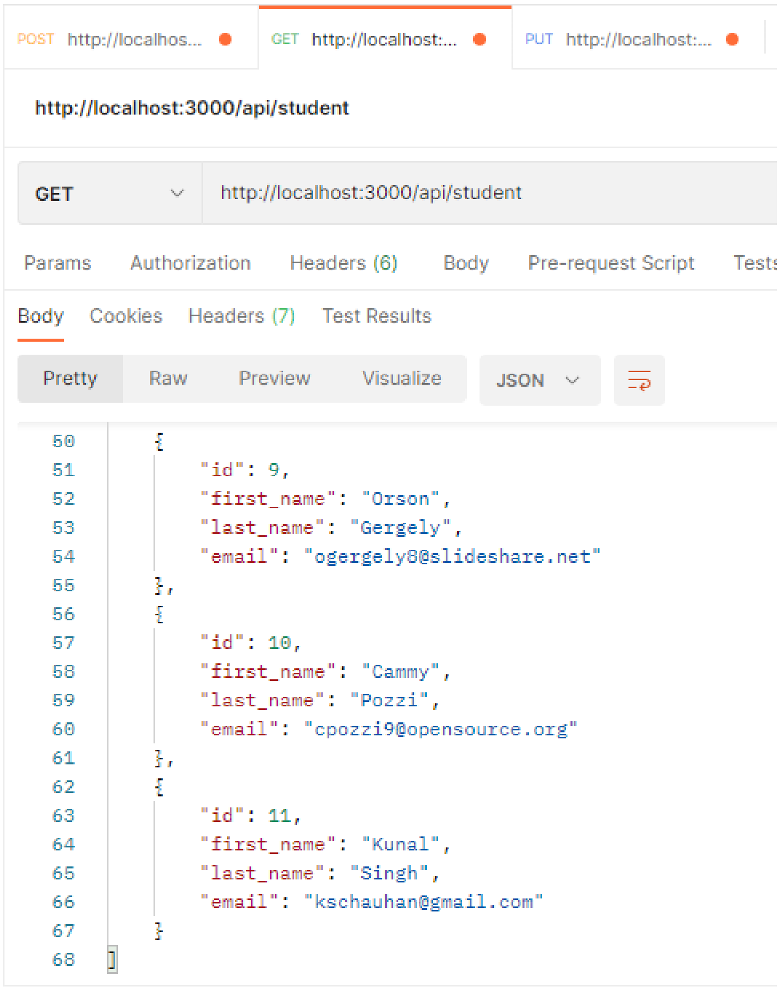


http://localhost:3000/api/student

POST http://localhost:3000/api/student

```
{
    "first_name"  :  "Kunal",
    "last_name"  :  "Singh",
    "email"  :  "kschauhan@gmail.com"
}
```

Here we can see that we are posting the data on url using the post request and let’s check whether the data posted or not.

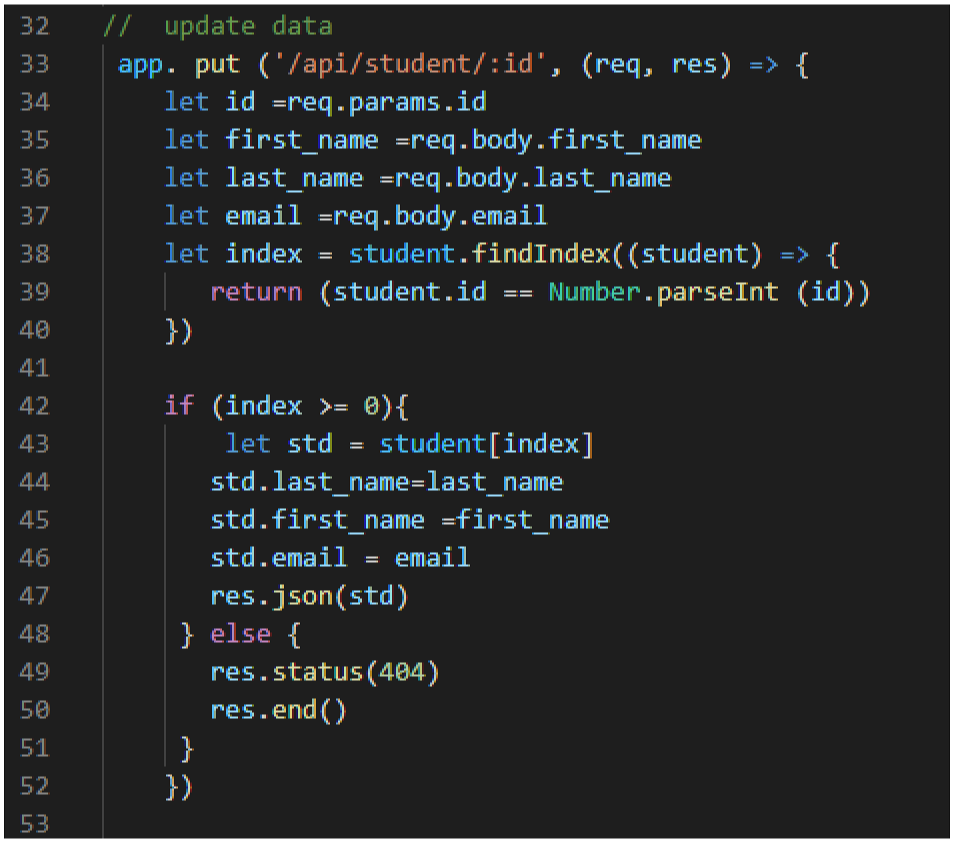


```
    {
        "id": 9,
        "first_name": "Orson",
        "last_name": "Gergely",
        "email": "ogergely8@slideshare.net"
    },
    {
        "id": 10,
        "first_name": "Cammy",
        "last_name": "Pozzi",
        "email": "cpozzi9@opensource.org"
    },
    {
        "id": 11,
        "first_name": "Kunal",
        "last_name": "Singh",
        "email": "kschauhan@gmail.com"
    }
]
```

Here we can see the data is posted in the json file.

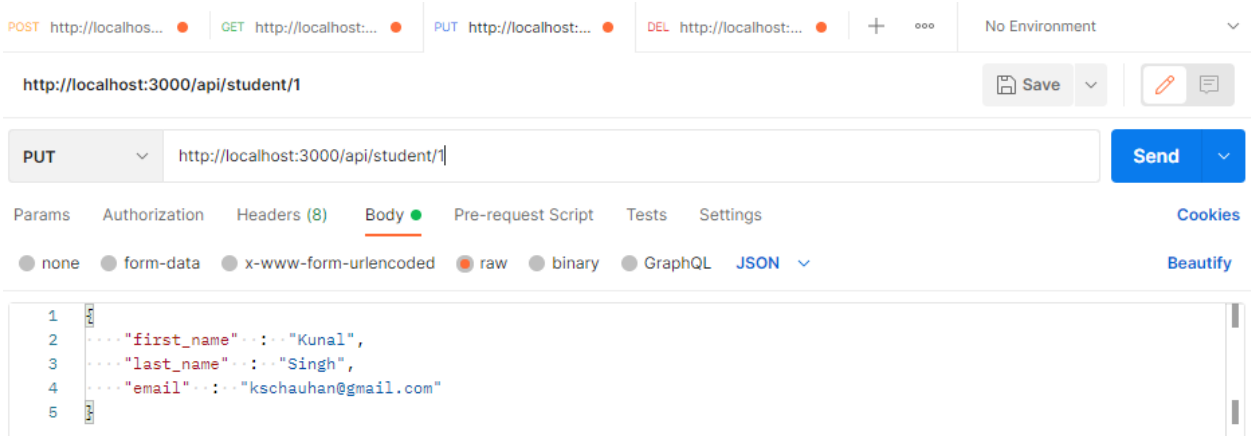
Step-8 **PUT Operation;** Now we will do PUT operation which is nothing but updating the data.Here we send the unique id with URL for specific id we want to update the data.

Code:



```
//  update data
app. put ('/api/student/:id', (req, res) => {
    let id =req.params.id
    let first_name =req.body.first_name
    let last_name =req.body.last_name
    let email =req.body.email
    let index = student.findIndex((student) => {
        return (student.id == Number.parseInt (id))
    })

    if (index >= 0){
        let std = student[index]
        std.last_name=last_name
        std.first_name =first_name
        std.email = email
        res.json(std)
    } else {
        res.status(404)
        res.end()
    }
    })
```

Output:

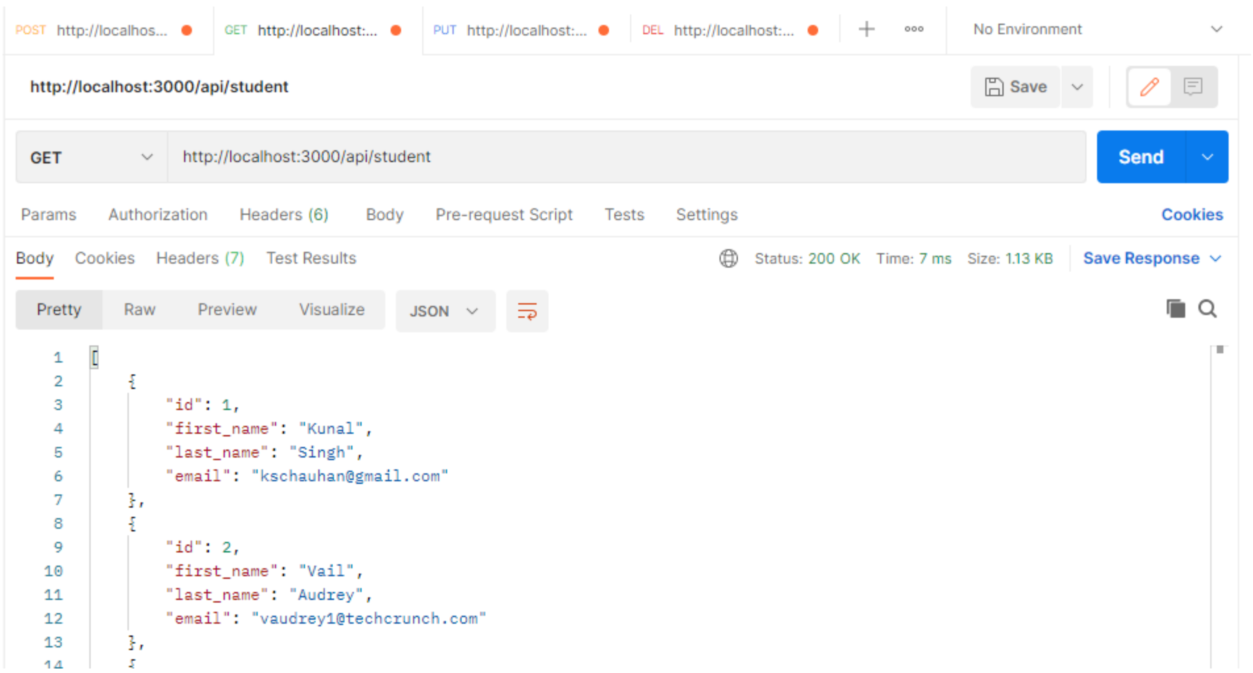


http://localhost:3000/api/student/1

PUT http://localhost:3000/api/student/1

```
{
    "first_name"  :  "Kunal",
    "last_name"  :  "Singh",
    "email"  :  "kschauhan@gmail.com"
}
```

Here we are passing the id =1 as we want update the data of id one with these given values.



http://localhost:3000/api/student

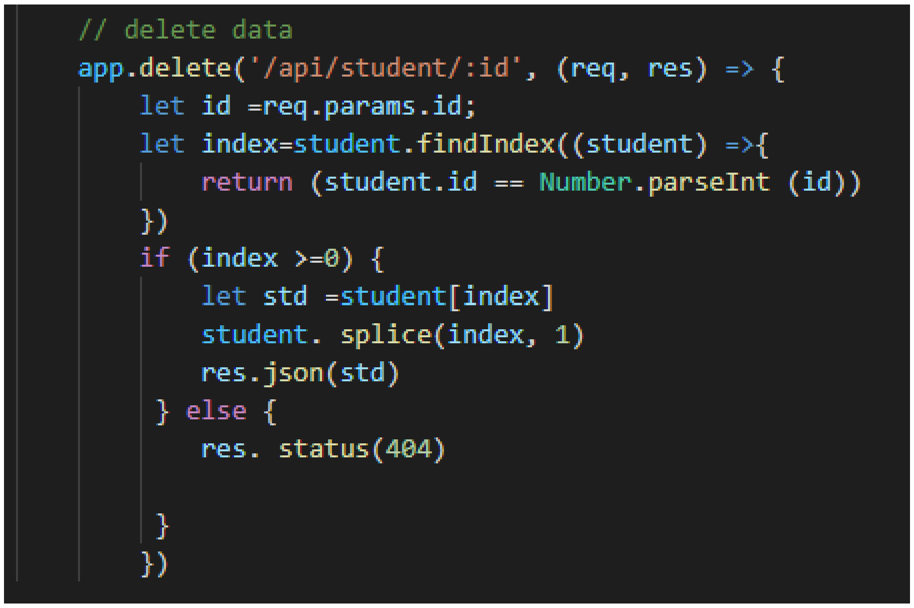
GET http://localhost:3000/api/student

Status: 200 OK Time: 7 ms Size: 1.13 KB

```
[
    {
        "id": 1,
        "first_name": "Kunal",
        "last_name": "Singh",
        "email": "kschauhan@gmail.com"
    },
    {
        "id": 2,
        "first_name": "Vail",
        "last_name": "Audrey",
        "email": "vaudrey1@techcrunch.com"
    },
    {
```

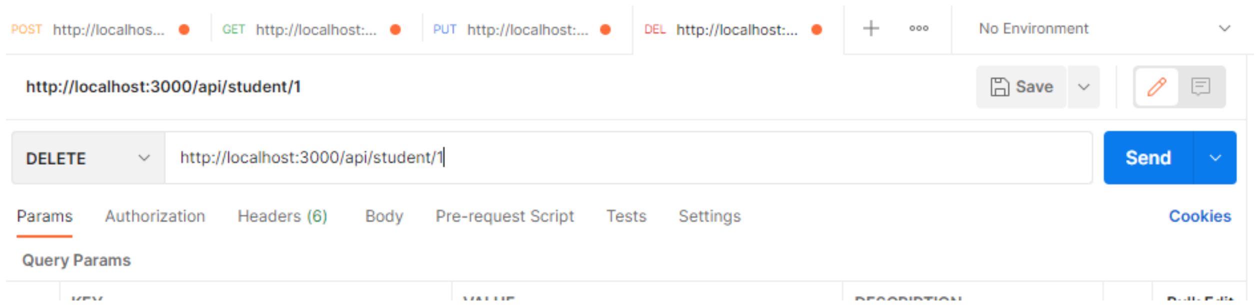
And here we can see that the data is updated for the id 1 with the value what we give.

Step-9 **Delete Operation:** Now we will move delete operation in which we can delete the data with the specific id.
Code:



```
// delete data
app.delete('/api/student/:id', (req, res) => {
    let id =req.params.id;
    let index=student.findIndex((student) =>{
        return (student.id == Number.parseInt (id))
    })
    if (index >=0) {
        let std =student[index]
        student. splice(index, 1)
        res.json(std)
     } else {
        res. status(404)

     }
    })
```

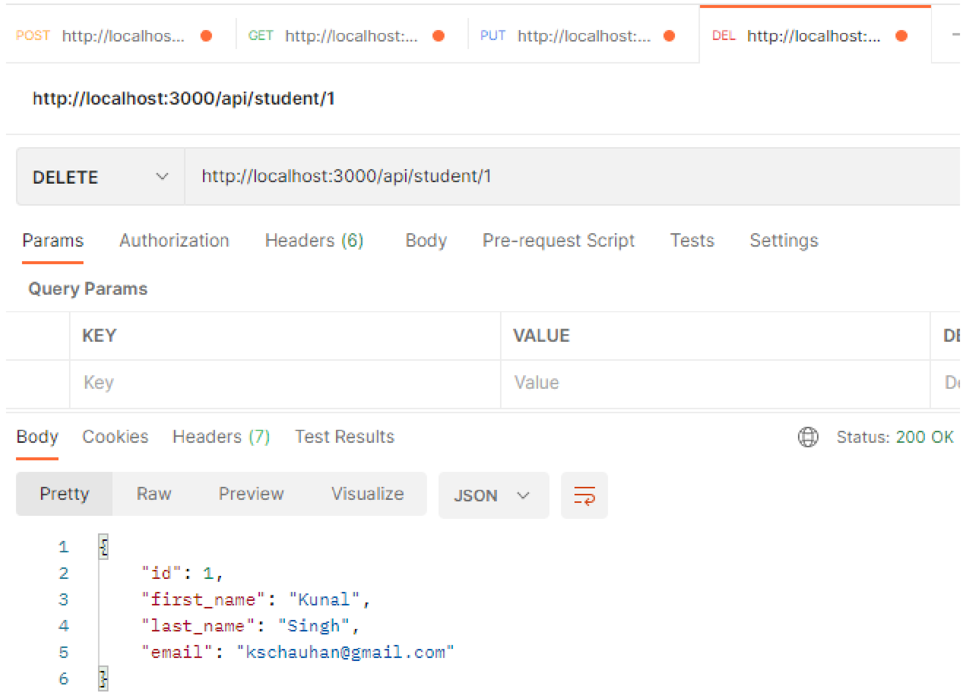
Output:



http://localhost:3000/api/student/1

DELETE http://localhost:3000/api/student/1

Here we are deleting the data associated with id 1 so we are sending /1 with the url and also getting a response that which data is deleted.



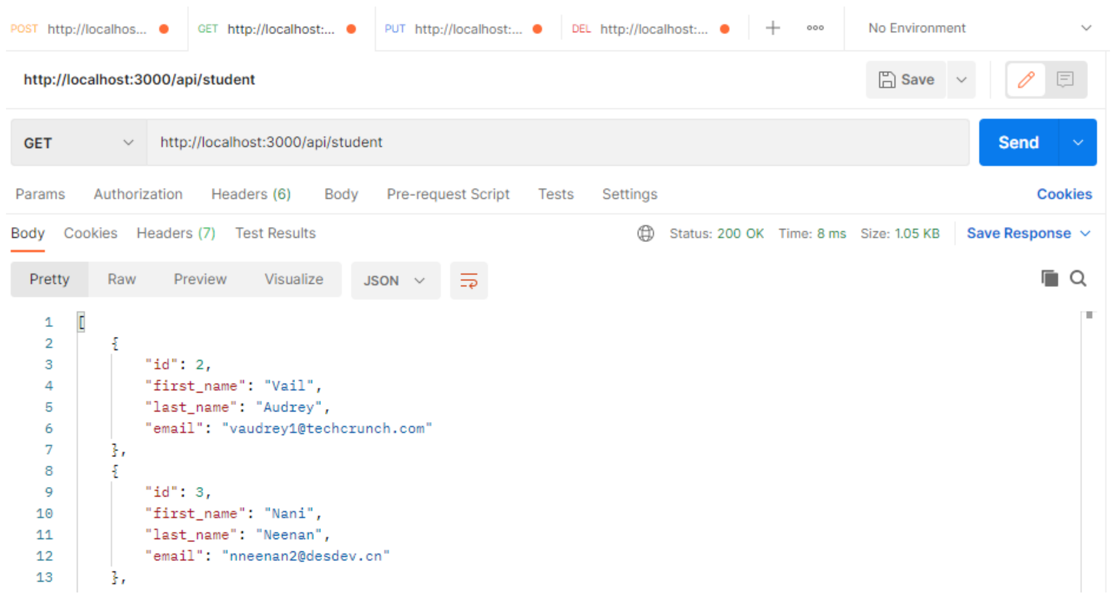
http://localhost:3000/api/student/1

DELETE http://localhost:3000/api/student/1

Status: 200 OK

```
{
    "id": 1,
    "first_name": "Kunal",
    "last_name": "Singh",
    "email": "kschauhan@gmail.com"
}
```

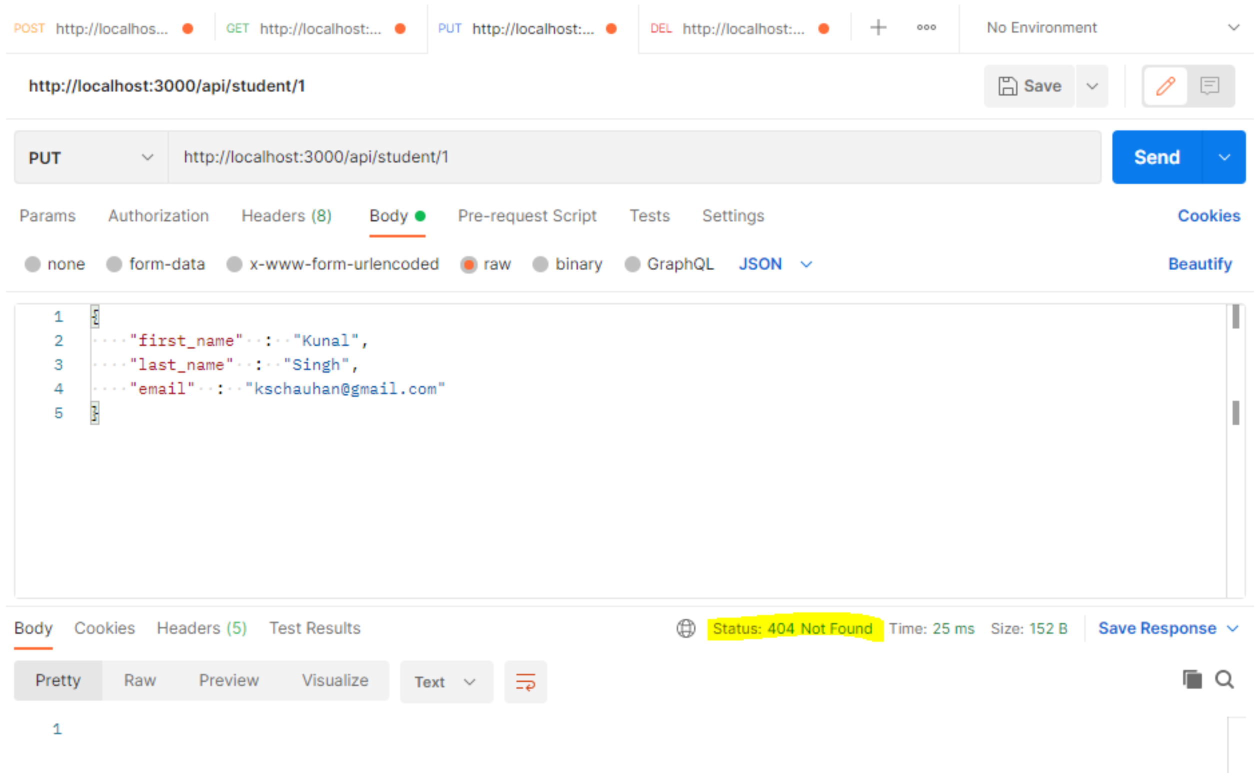
This is showing which data is deleted and now we check in the final file whether the data is deleted or not.



```
[
    {
        "id": 2,
        "first_name": "Vail",
        "last_name": "Audrey",
        "email": "vaudrey1@techcrunch.com"
    },
    {
        "id": 3,
        "first_name": "Nani",
        "last_name": "Neenan",
        "email": "nneenan2@desdev.cn"
    },
```

And here we can see that data is deleted for id =1.

And when we cross check for the data



```
{
    "first_name"  :  "Kunal",
    "last_name"  :  "Singh",
    "email"  :  "kschauhan@gmail.com"
}
```

It will show the error 404 Not found.

So this is all about the CRUD operation in Express Js.

Thank You.