REST API

#Node JS Notes

What is API ?

- A web service is a collection of open protocols and standards used for exchanging data between applications or systems.
- Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.

What is API ?

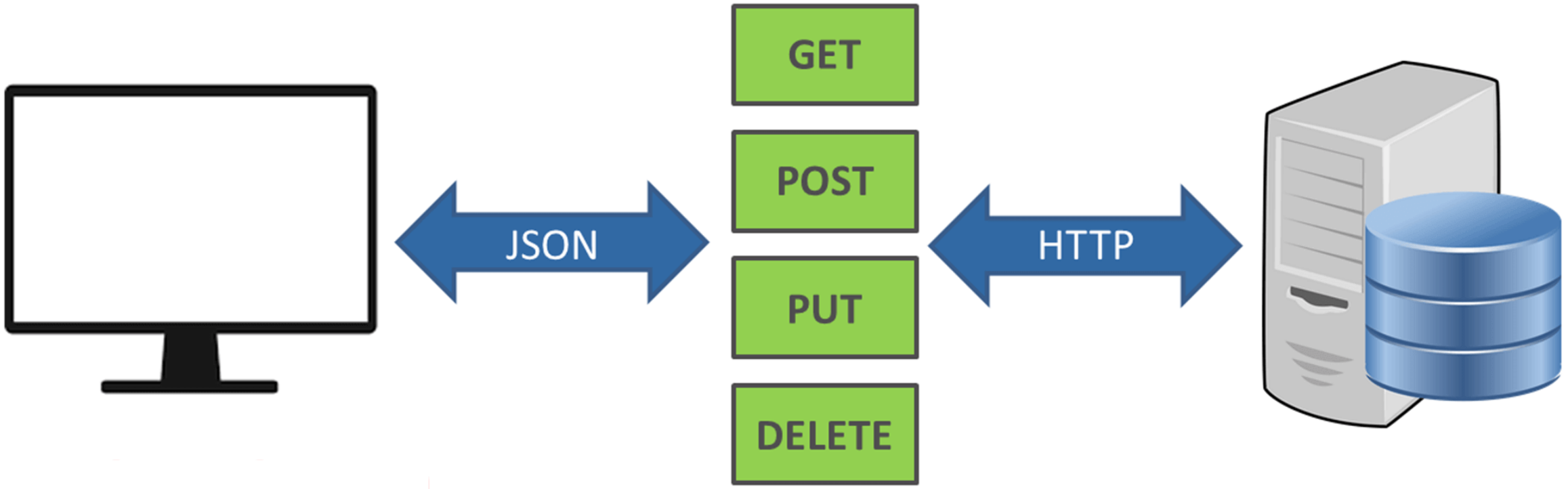
- **API** (Application Program Interface) is an agreed way to send and receive data between computers.
- For example, if you want to display Google Maps on your site, but the maps are on Google's servers, you need a way to ask Google to provide you with the maps.
- The way to ask Google to send you the requested maps is through an API provided by Google that tells you to which web addresses should you send the requests to get the data.
- In a more formal language, you need to send a request to the remote server to get a response.

What is Rest?

- **REST** (Representational State Transfer) is an API that defines a set of functions that programmers can use to send requests and receive responses using the HTTP protocol methods such as GET and POST.

What is REST API ?

- **REST API** can be used by any site or application no matter what language it is written in because the requests are based on the universal HTTP protocol, and the information is usually returned in the JSON format that almost all of the programming languages can read.



Client sends a **request**

HTTP methods

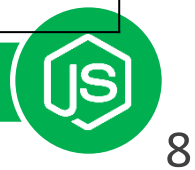
Server sends a **response**

HTTP methods

- Following four HTTP methods are commonly used in REST based architecture.
- **GET** – Retrieves data from a remote server. It can be a single resource or a list of resources.
- **POST** – Creates a new resource on the remote server.
- **PUT** – Updates the data on the remote server.
- **DELETE** – Deletes data from the remote server.

HTTP status codes

Code	Message	Description
200	**OK**	**The data was received and the operation was performed.**
201	Created	The data was received and a new resource was created. The response needs to return the data in the payload.
204	No content	The operation was successful but no data is returned in the response body. This is useful when deleting a resource.
301	Moved permanently	This and all the future requests should be redirected to a new URL.
302	Moved temporarily	The resource moved temporarily to another URL.
400	Bad request	The server cannot accept the request because something is wrong with the client or the request that it sent.
403	Forbidden	The client is not allowed to use the resource.
404	**Not found**	**The computer is not able to find the resource.**
405	Method not allowed	For example, when sending a DELETE request to a server that doesn't support the method.
500	**Internal server error**	**Service unavailable due to error on the server side.**



- For example, to create a new article on our blog, we should send a request to a remote server using the POST HTTP method.
- To view a single article or a list of articles, we use the GET method. The PUT method can be used to edit an existing article, and the DELETE method to delete.

Method	Url	Action
GET	users	fetch all users
GET	user/1	fetch user with id =1
POST	user	add new user
PUT	user/1	update user by id = 1
DELETE	user/1	delete user by id = 1

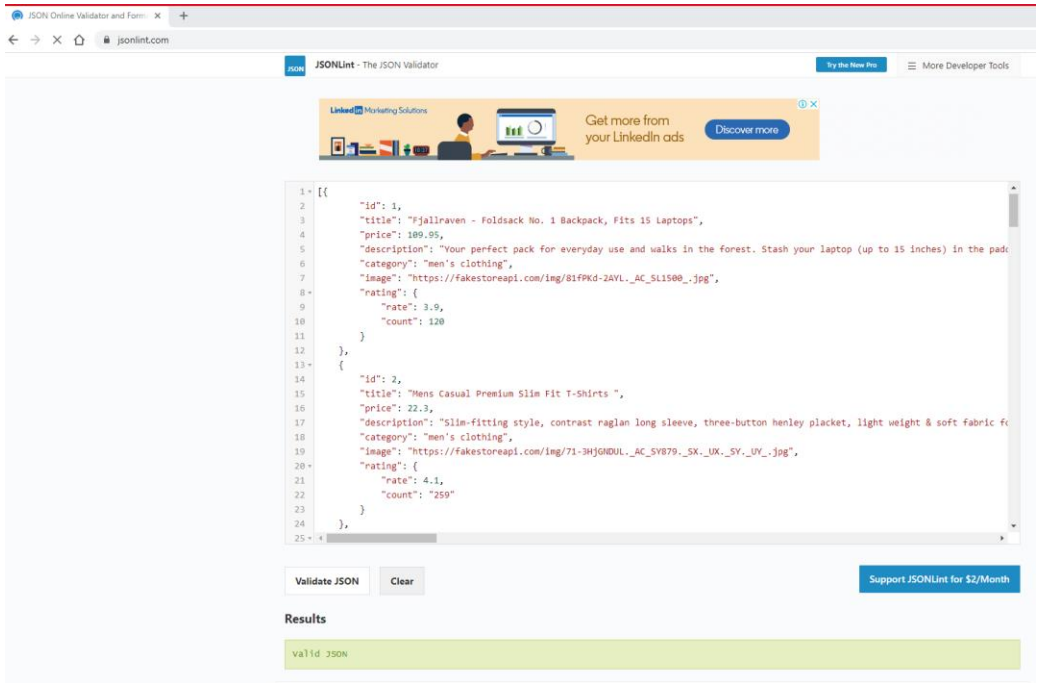
- **GET** /photo — Retrieve all photos
- **GET** /photo/:id — Retrieve a photo by its ID
- **POST** /photo — Create a new photo
- **PUT** /photo/:id — Update properties of a photo by its ID
- **DELETE** /photo/:id — Delete a photo by its ID

Software to Test API

- PostMan
  - https://www.postman.com/downloads/
- VS Code (Thunder Client)
  - https://marketplace.visualstudio.com/items?itemName=rangav.vscode-thunder-client
  - https://www.thunderclient.io/
- JSON Viewer Chrome Extenstion (Get Method)
  - https://chrome.google.com/webstore/detail/json-viewer/gbmdgpbipfallnflgajpaliibnhdgobh

JSON Validator

- https://jsonlint.com/
- https://jsonformatter.org/
- https://jsonformatter.curiousconcept.com/
- https://elmah.io/tools/json-formatter/
- https://codebeautify.org/jsonvalidator



Fake JSON Data

- https://jsonplaceholder.typicode.com/
- https://www.mockaroo.com/
- https://fakestoreapi.com/
- https://www.npmjs.com/package/json-server
- https://www.mockachino.com/
- https://dummy.restapiexample.com/
- https://gorest.co.in/

JSON DATA Pass

- **Simple Json Data Convert**

```
res.send(JSON.stringify(rows));
```

- **Json Data with Status Code**

```
res.status(200).send(JSON.stringify(rows));
```

- **Json Data with Status Code Error**

```
res.status(500).send(JSON.stringify({"msg":"Error"}));
```

- Json Data with Object and Array

```
res.send(JSON.stringify({"status": 200,"flag": 1, "message": "Data Fetch", "data": rows}));

res.send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));
```

- Json Data with Status Code a Object and Array

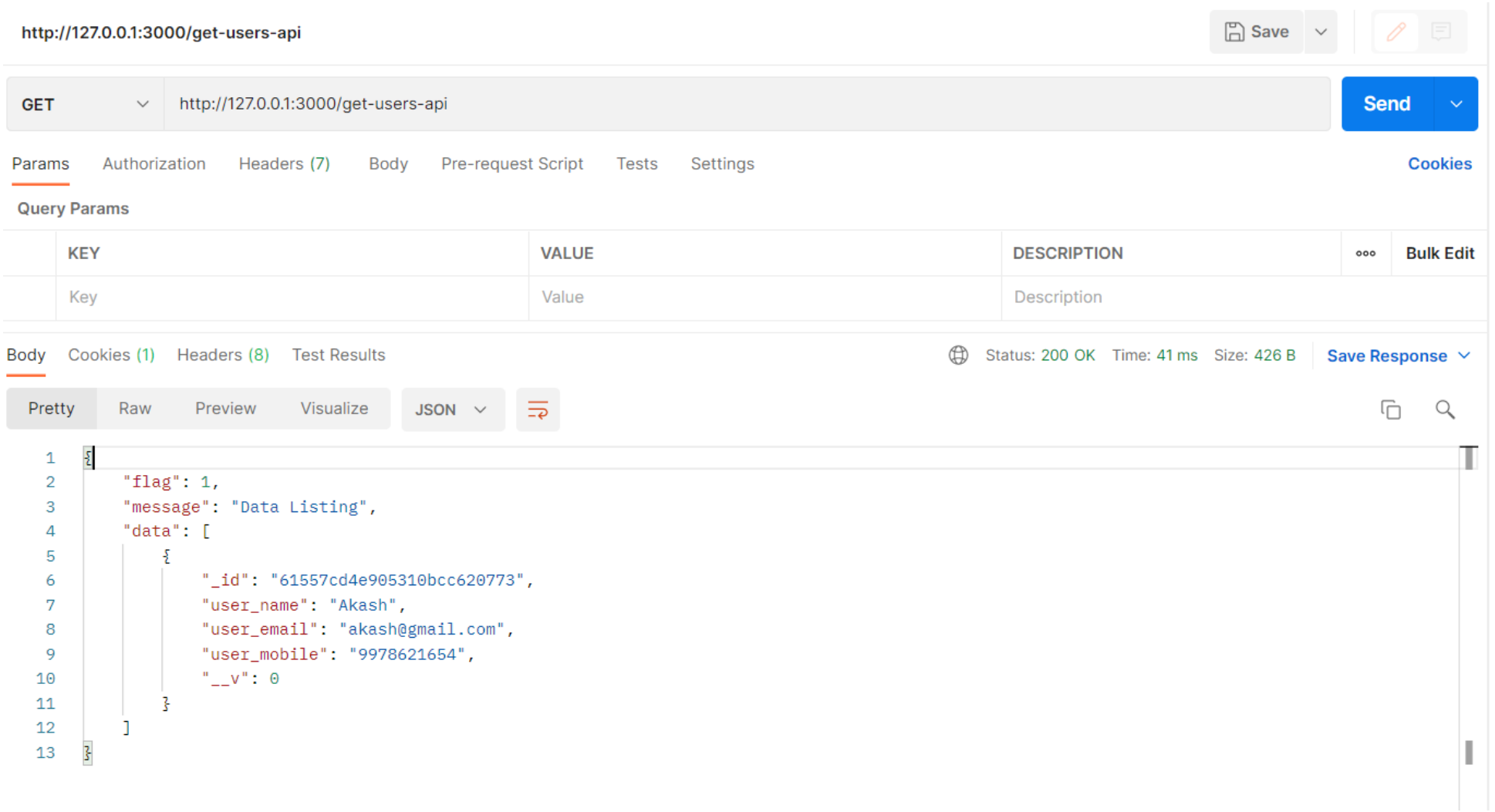
```
res.status(500).send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));

res.status(200).send(JSON.stringify({"status": 200,"flag": 1, "message": "Data Deleted", "data": ''}));
```

MongoDB

Display

```
//Get All Data API
router.get('/get-users-api', function(req, res, next) {
    UsersModel.find({}, function(err, mydata) {
        if (err) {
            res.send(JSON.stringify({'flag':0,'message':'Error in API','err' : err}));
        } else {
            res.send(JSON.stringify({'flag':1,'message':'Data Listing','data':mydata}));
        }
    });
});
```

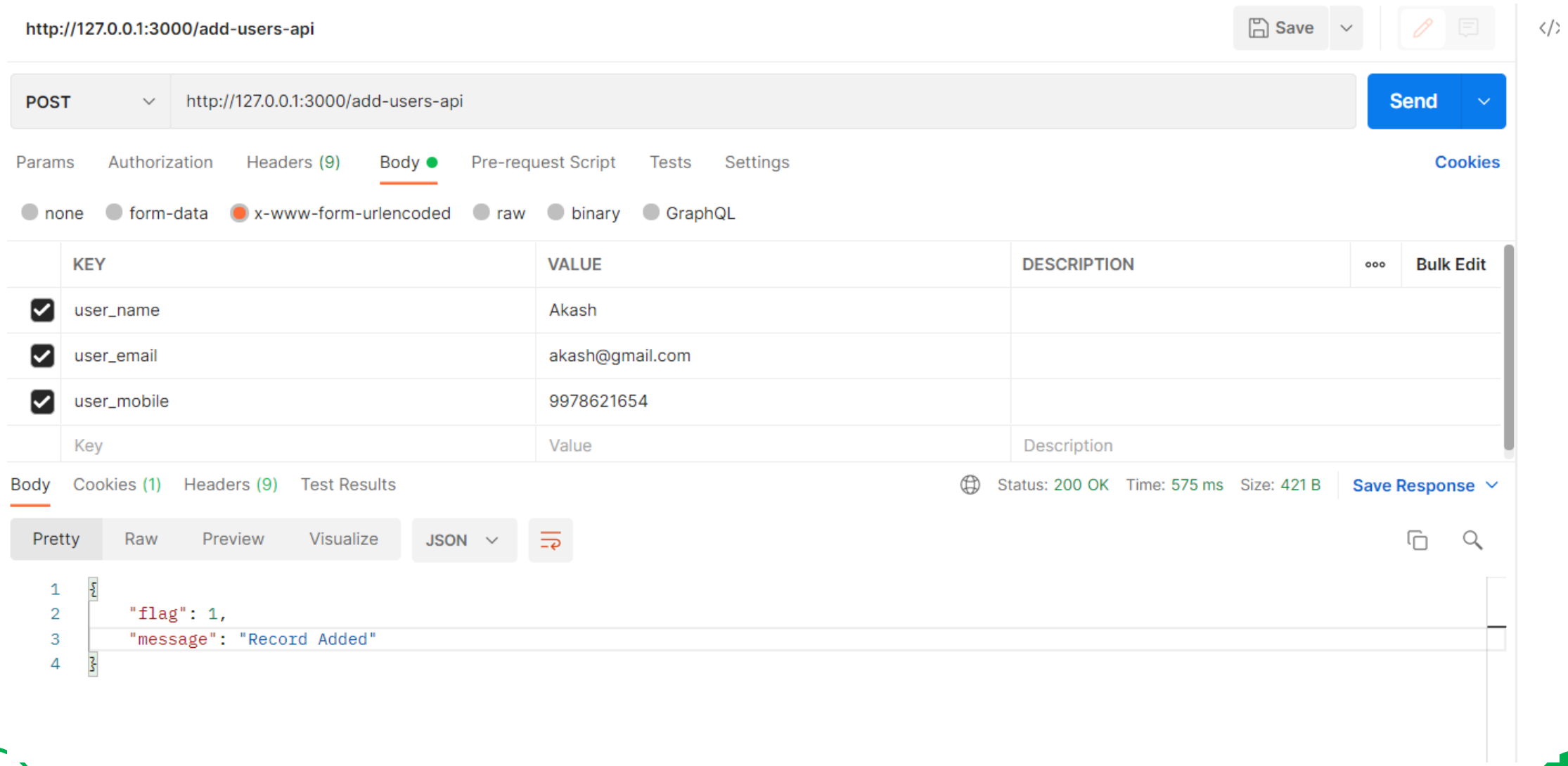
Display


http://127.0.0.1:3000/get-users-api
Save
GET
http://127.0.0.1:3000/get-users-api
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Key
Value
Description
Body
Cookies (1)
Headers (8)
Test Results
Status: 200 OK
Time: 41 ms
Size: 426 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"flag": 1,
"message": "Data Listing",
"data": [
{
"_id": "61557cd4e905310bcc620773",
"user_name": "Akash",
"user_email": "akash@gmail.com",
"user_mobile": "9978621654",
"__v": 0
}
]
}

Add Record

```
//Add Data API
router.post('/add-users-api', function(req, res, next) {
    console.log(req.body);

    const mybodydata = {
        user_name: req.body.user_name,
        user_email: req.body.user_email,
        user_mobile: req.body.user_mobile
    }
    var data = UsersModel(mybodydata);

    data.save(function(err) {
        if (err) {
            res.send(JSON.stringify({'flag':0,'message':'Error in API','err' : err}));
        } else {
            res.send(JSON.stringify({'flag':1,'message':'Record Added'}));
        }
    })
});
```

Add Record


http://127.0.0.1:3000/add-users-api
Save
POST
http://127.0.0.1:3000/add-users-api
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
user_name
Akash
user_email
akash@gmail.com
user_mobile
9978621654
Key
Value
Description
Body
Cookies (1)
Headers (9)
Test Results
Status: 200 OK
Time: 575 ms
Size: 421 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"flag": 1,
"message": "Record Added"
}

Single Record

```
/* GET SINGLE Data BY ID */
router.get('/get-users-details-api/:id', function(req, res, next) {
  UsersModel.findById(req.params.id, function (err, mydata) {
    if(err){
        res.send(JSON.stringify({'flag':0,'message':'Error in API','err' : err}));

  }else{
    res.send(JSON.stringify({'flag':1,'message':'Data Listing','data':mydata}));

  }
  });
});
```

http://127.0.0.1:3000/get-users-details-api/61557cd4e905310bcc620773

Save

GET http://127.0.0.1:3000/get-users-details-api/61557cd4e905310bcc620773 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	ooo	Bulk Edit
Key	Value	Description		

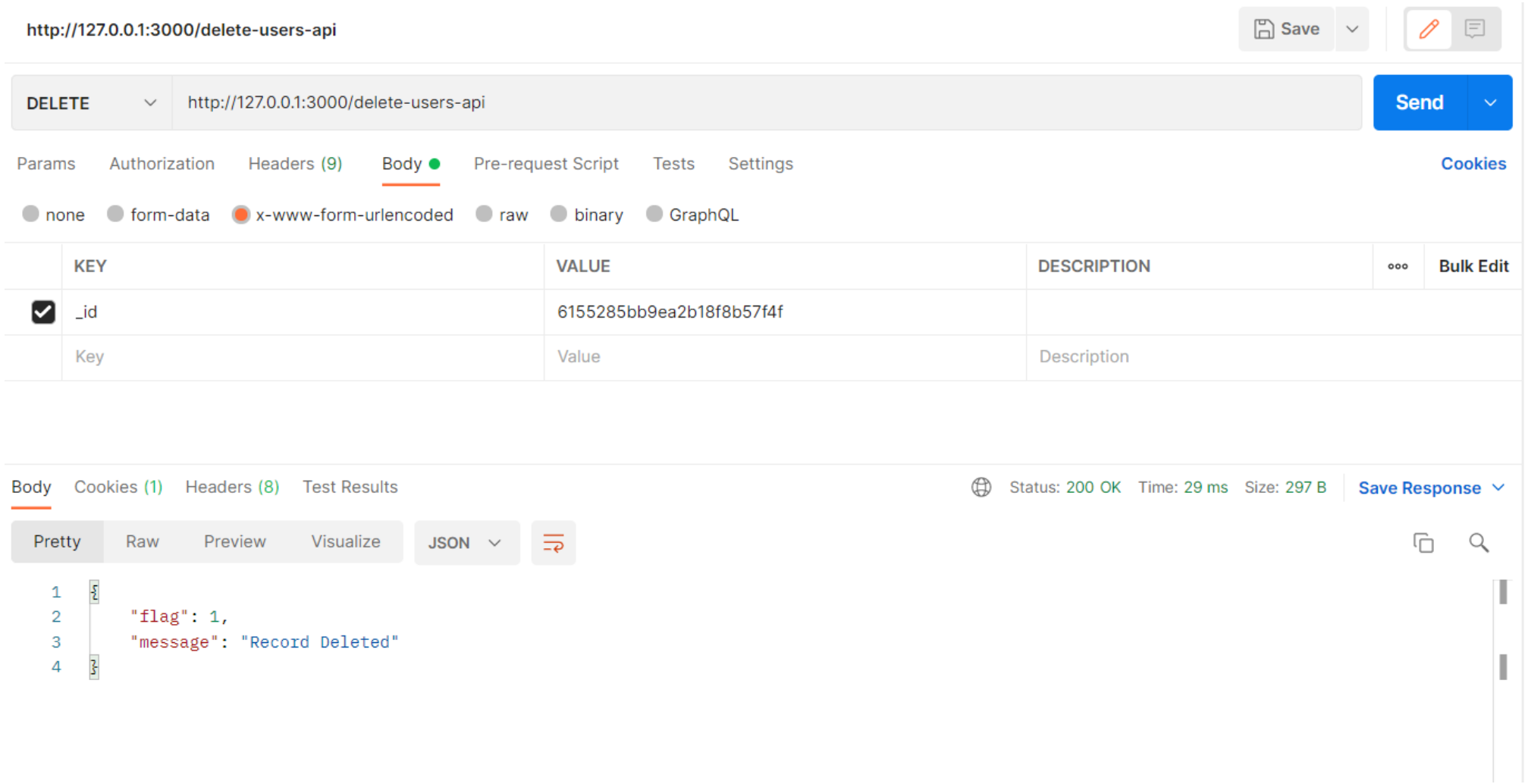
Body Cookies (1) Headers (8) Test Results Status: 200 OK Time: 22 ms Size: 424 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "flag": 1,
    "message": "Data Listing",
    "data": {
        "_id": "61557cd4e905310bcc620773",
        "user_name": "Akash",
        "user_email": "akash@gmail.com",
        "user_mobile": "9978621654",
        "__v": 0
    }
}
```

Delete

```
/* DELETE Data BY ID */
router.delete('/delete-users-api', function(req, res, next) {
  UsersModel.findByIdAndRemove(req.body._id, function (err, post) {
    if (err) {
        res.send(JSON.stringify({'flag':0,'message':'Error in API','err' : err}));

    } else {
        res.send(JSON.stringify({'flag':1,'message':'Record Deleted'}));
    }
  });
});
```


http://127.0.0.1:3000/delete-users-api
Save
DELETE
http://127.0.0.1:3000/delete-users-api
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
_id
6155285bb9ea2b18f8b57f4f
Key
Value
Description
Body
Cookies (1)
Headers (8)
Test Results
Status: 200 OK
Time: 29 ms
Size: 297 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"flag": 1,
"message": "Record Deleted"
}

Update

```
/* UPDATE Data API */
router.put('/update-users-api/:id', function(req, res, next) {
  console.log(req.params.id);
  UsersModel.findByIdAndUpdate(req.params.id, req.body, function (err, post) {
  if (err) {
    res.send(JSON.stringify({'flag':0,'message':'Error in API','err' : err}));
  } else {
    res.send(JSON.stringify({'flag':1,'message':'Record Updated'}));
  }
});
});
```

http://127.0.0.1:3000/update-users-api/61557cd4e905310bcc620773

Save

PUT http://127.0.0.1:3000/update-users-api/61557cd4e905310bcc620773 Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION	•••	Bulk Edit
user_name	Akash Padhiyar			
user_email	akashpadhiyar@gmail.com			
user_mobile	9978621654			
Key	Value	Description		

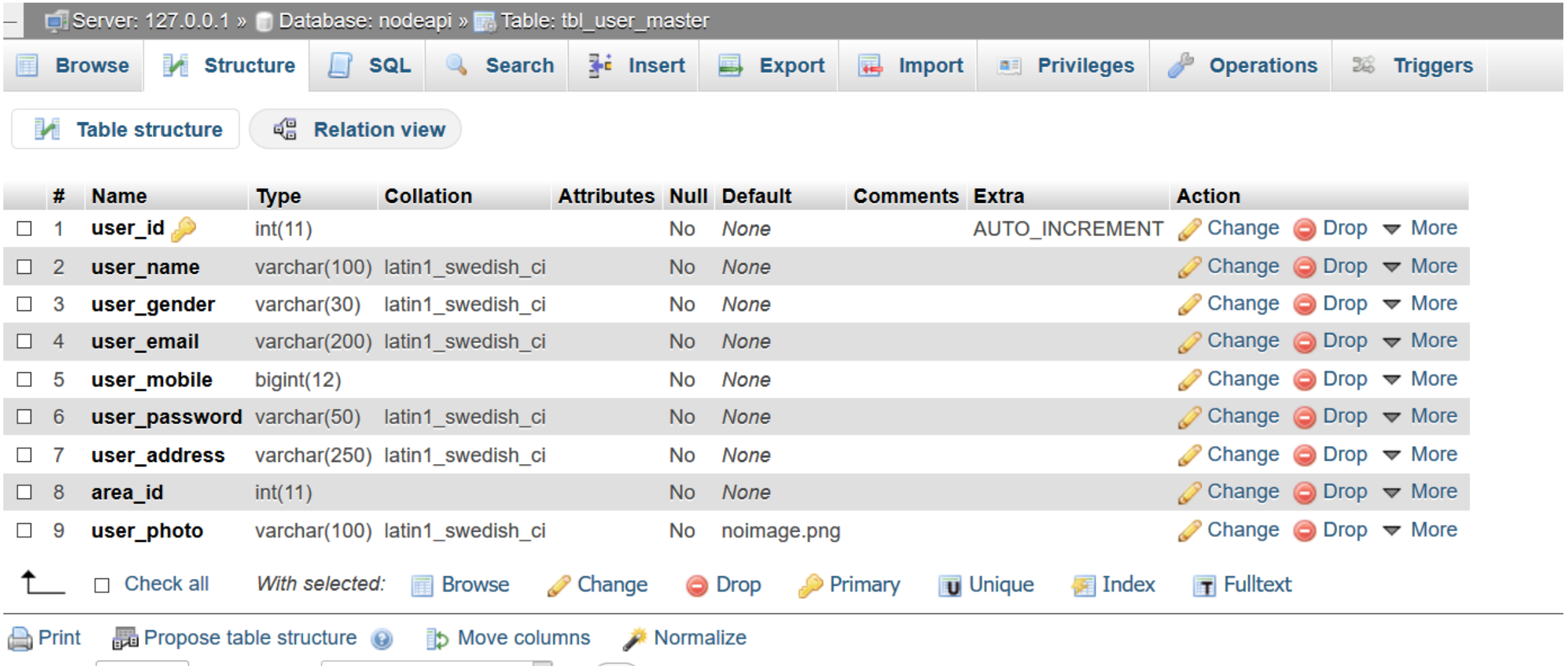
Body Cookies (1) Headers (8) Test Results Status: 200 OK Time: 43 ms Size: 297 B Save Response

Pretty Raw Preview Visualize JSON

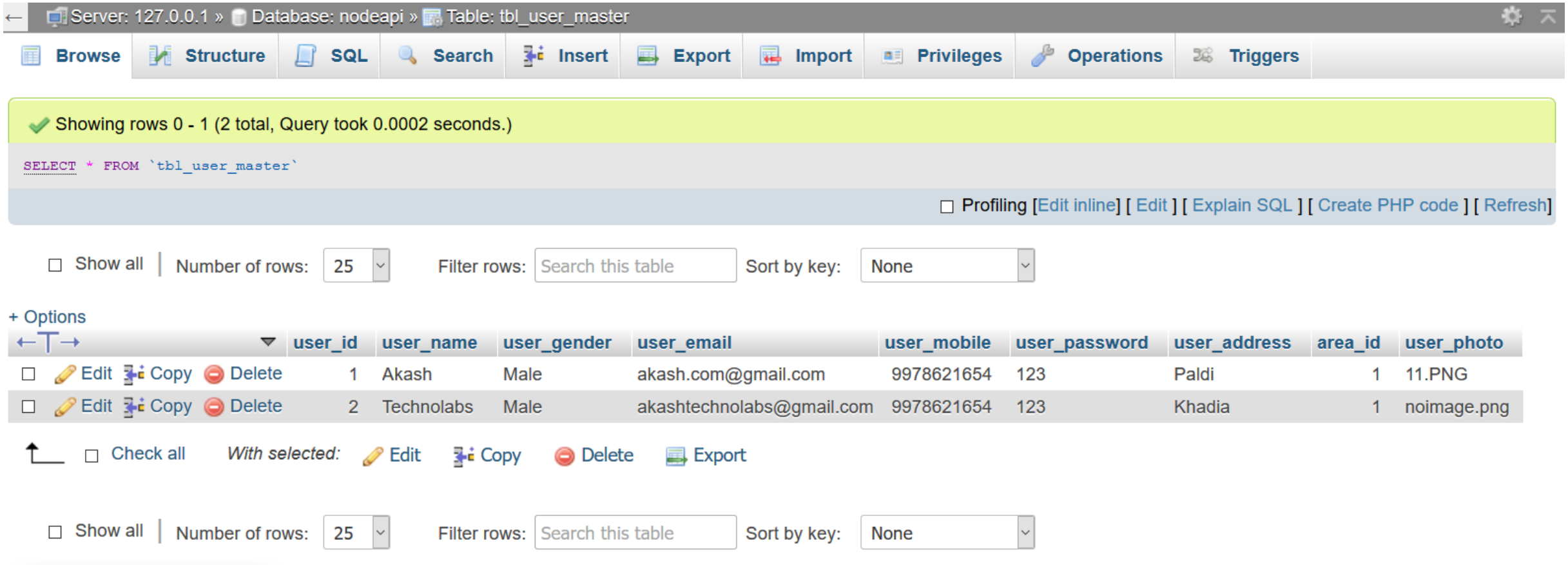
```
{
    "flag": 1,
    "message": "Record Updated"
}
```

API Create using Mysql

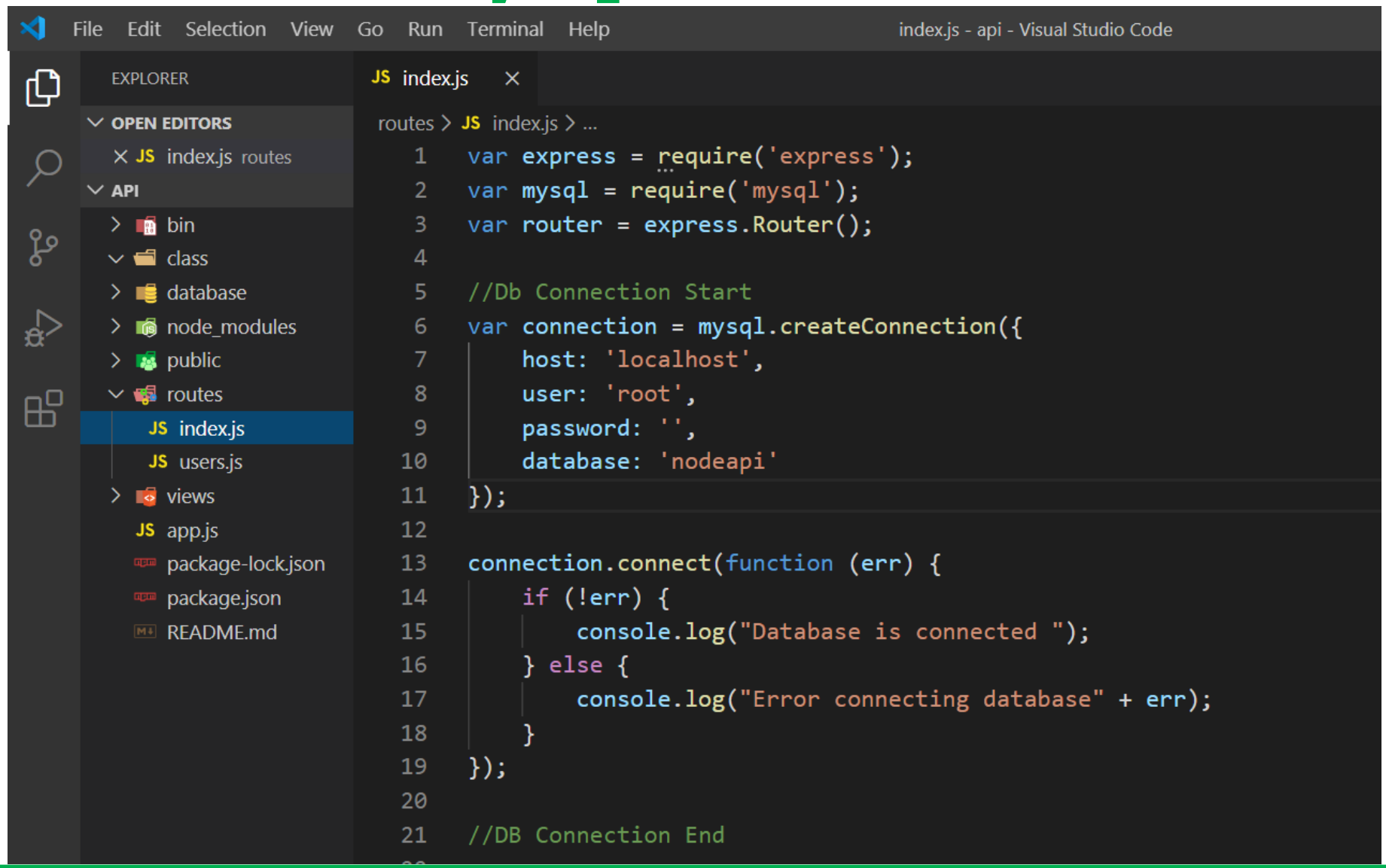
Table Structure



Record


Server: 127.0.0.1 » Database: nodeapi » Table: tbl_user_master
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
Showing rows 0 - 1 (2 total, Query took 0.0002 seconds.)
SELECT * FROM `tbl_user_master`
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
user_id user_name user_gender user_email user_mobile user_password user_address area_id user_photo
Edit Copy Delete 1 Akash Male akash.com@gmail.com 9978621654 123 Paldi 1 11.PNG
Edit Copy Delete 2 Technolabs Male akashtechnolabs@gmail.com 9978621654 123 Khadia 1 noimage.png
Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Mysql Connection



Get all User Data

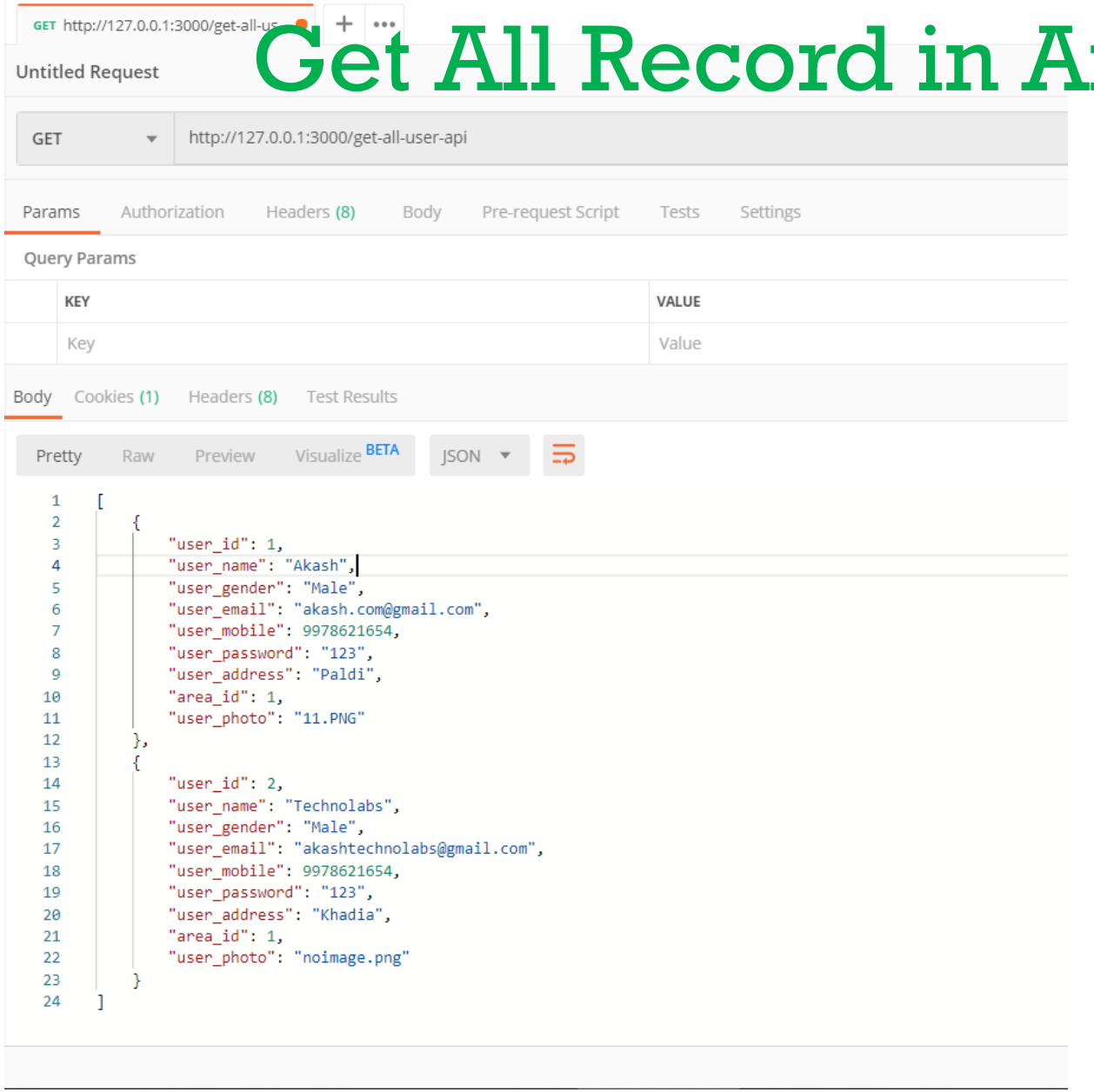
File Edit Selection View Go Run Terminal Help index.js - api - Visual Studio Code

index.js

routes > index.js > ...

```
22
23
24  //Display Home Page (Index Page)
25  router.get('/', function (req, res, next) {
26      res.render('index');
27  });
28
29  //Display all records API
30  router.get('/get-all-user-api', function (req, res, next) {
31
32      connection.query("select * from tbl_user_master", function (err, rows) {
33          if (err) {
34              res.send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));
35          } else {
36              console.log(rows);
37              res.send(JSON.stringify(rows));
38          }
39      })
40  });
41
```

Get All Record in Array (Postman)



Chrome Output


```
// 20200418202959
// http://127.0.0.1:3000/get-all-user-api

[
  {
    "user_id": 1,
    "user_name": "Akash",
    "user_gender": "Male",
    "user_email": "akash.com@gmail.com",
    "user_mobile": 9978621654,
    "user_password": "123",
    "user_address": "Paldi",
    "area_id": 1,
    "user_photo": "11.PNG"
  },
  {
    "user_id": 2,
    "user_name": "Technolabs",
    "user_gender": "Male",
    "user_email": "akashtechnolabs@gmail.com",
    "user_mobile": 9978621654,
    "user_password": "123",
    "user_address": "Khadia",
    "area_id": 1,
    "user_photo": "noimage.png"
  }
]
```

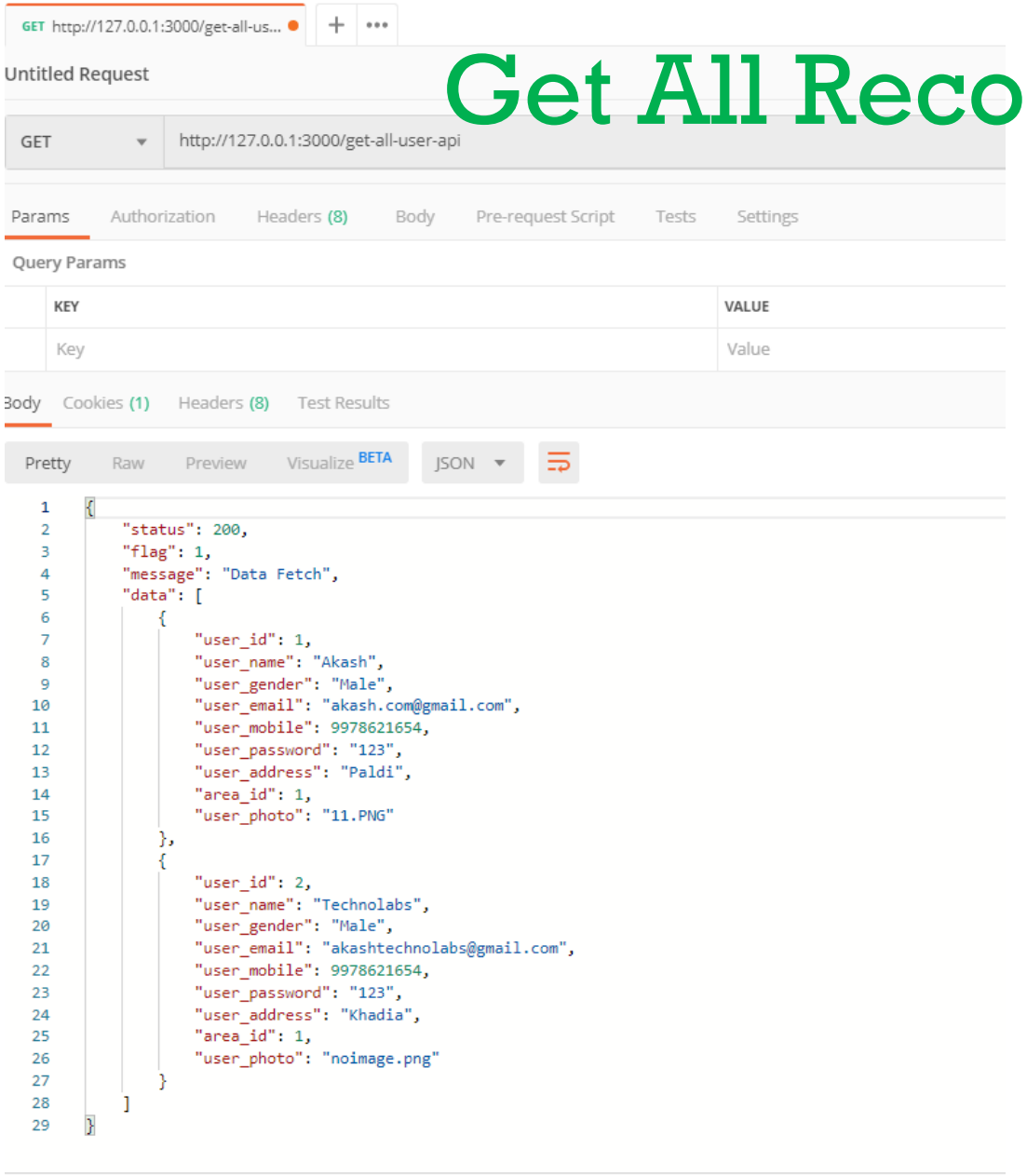
Get All Record with Object

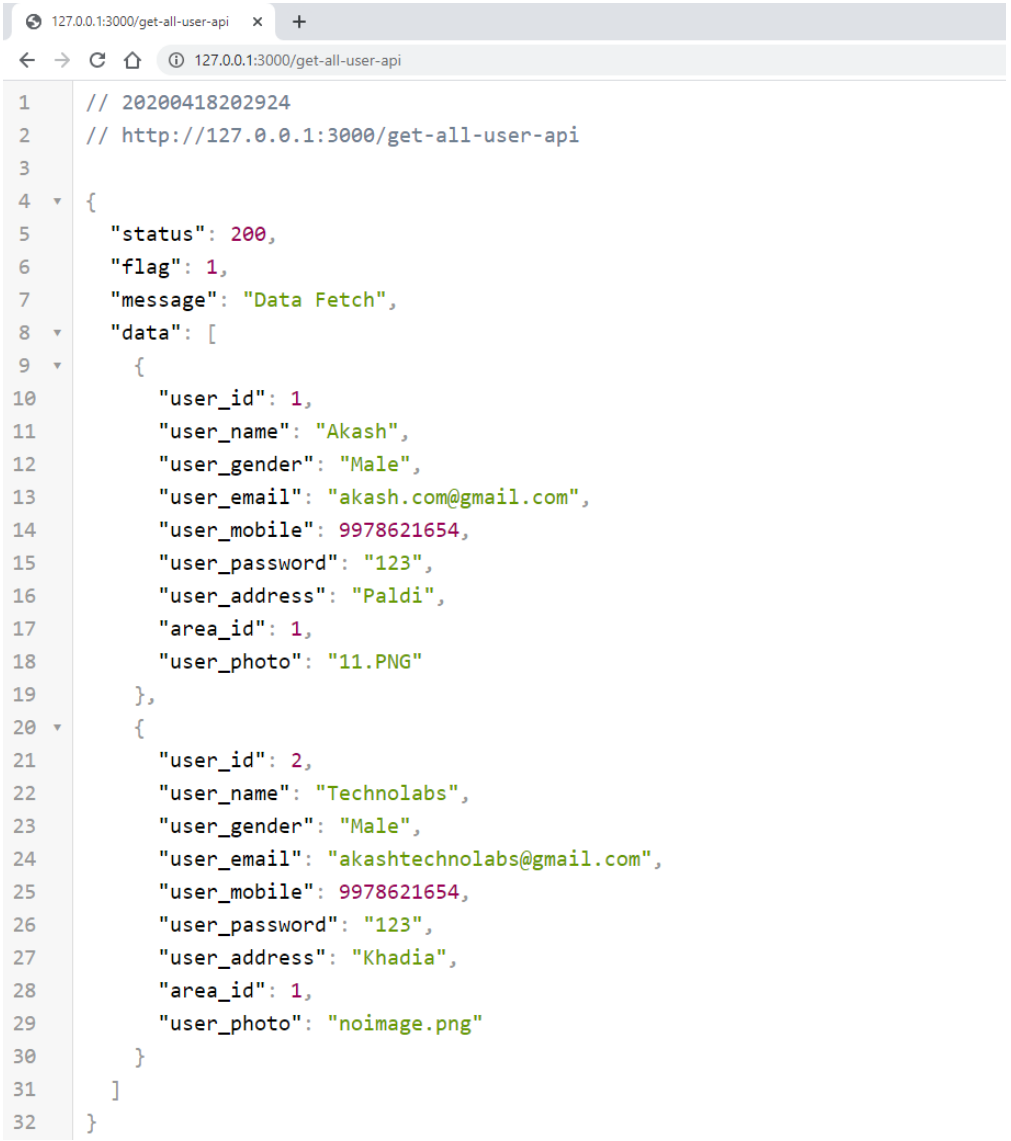
index.js

routes > index.js > router.get('/get-all-user-api') callback > connection.query("select * from tbl_user_master") callback

```
22
23
24  //Display Home Page (Index Page)
25  router.get('/', function (req, res, next) {
26      res.render('index');
27  });
28
29  //Display all records API
30  router.get('/get-all-user-api', function (req, res, next) {
31
32      connection.query("select * from tbl_user_master", function (err, rows) {
33          if (err) {
34              res.send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));
35          } else {
36              console.log(rows);
37              res.send(JSON.stringify({"status": 200,"flag": 1, "message": "Data Fetch", "data": rows}));
38          }
39      })
40  });
41
```

Get All Record with Object






```
// 20200418202924
// http://127.0.0.1:3000/get-all-user-api

{
  "status": 200,
  "flag": 1,
  "message": "Data Fetch",
  "data": [
    {
      "user_id": 1,
      "user_name": "Akash",
      "user_gender": "Male",
      "user_email": "akash.com@gmail.com",
      "user_mobile": 9978621654,
      "user_password": "123",
      "user_address": "Paldi",
      "area_id": 1,
      "user_photo": "11.PNG"
    },
    {
      "user_id": 2,
      "user_name": "Technolabs",
      "user_gender": "Male",
      "user_email": "akashtechnolabs@gmail.com",
      "user_mobile": 9978621654,
      "user_password": "123",
      "user_address": "Khadia",
      "area_id": 1,
      "user_photo": "noimage.png"
    }
  ]
}
```

JS index.js

routes > JS index.js > router.get('/get-single-user-api/:id') callback

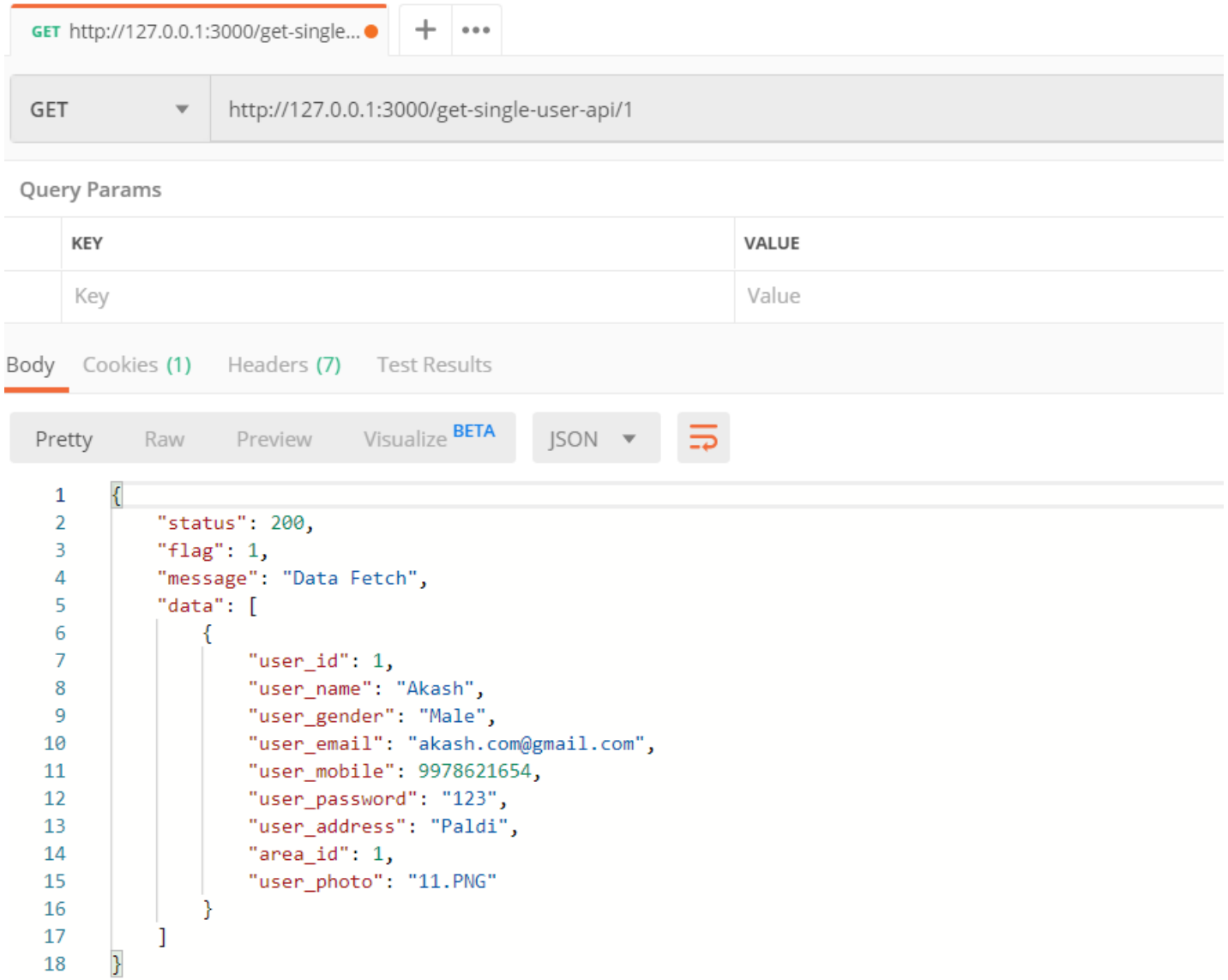
```
42  //Get Single User Data API
43  router.get('/get-single-user-api/:id', function (req, res, next) {
44      var id = req.params.id;
45      console.log("Parameter Value" + id);
46
47      connection.query("select * from tbl_user_master where user_id = ?", [id], function (err, rows) {
48          if (err) {
49              res.send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));
50          } else {
51              console.log(rows);
52              res.send(JSON.stringify({"status": 200,"flag": 1, "message": "Data Fetch", "data": rows}));
53          }
54      })
55
56  });
57
```

Check Data Present or not and Print Counter

index.js

routes > index.js > router.get('/get-all-user-api') callback > connection.query("select * from tbl_user_master") callback

```
28
29  //Display all records API
30  router.get('/get-all-user-api', function (req, res, next) {
31
32      connection.query("select * from tbl_user_master", function (err, rows) {
33          if (err) {
34              res.send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));
35          } else {
36              if(rows && rows.length>0)
37              {
38                  count  = rows.length;
39                  console.log(rows);
40                  res.send(JSON.stringify({"status": 200,"flag": 1, "message": count + " Records Found", "data": rows}));
41              }else{
42                  res.send(JSON.stringify({"status": 200,"flag": 0, "message": "NO Records Found"}));
43              }
44          }
45      })
46  });
```


GET http://127.0.0.1:3000/get-single...
GET
http://127.0.0.1:3000/get-single-user-api/1
Query Params
KEY
VALUE
Key
Value
Body
Cookies (1)
Headers (7)
Test Results
Pretty
Raw
Preview
Visualize BETA
JSON
{
"status": 200,
"flag": 1,
"message": "Data Fetch",
"data": [
{
"user_id": 1,
"user_name": "Akash",
"user_gender": "Male",
"user_email": "akash.com@gmail.com",
"user_mobile": 9978621654,
"user_password": "123",
"user_address": "Paldi",
"area_id": 1,
"user_photo": "11.PNG"
}
]
}

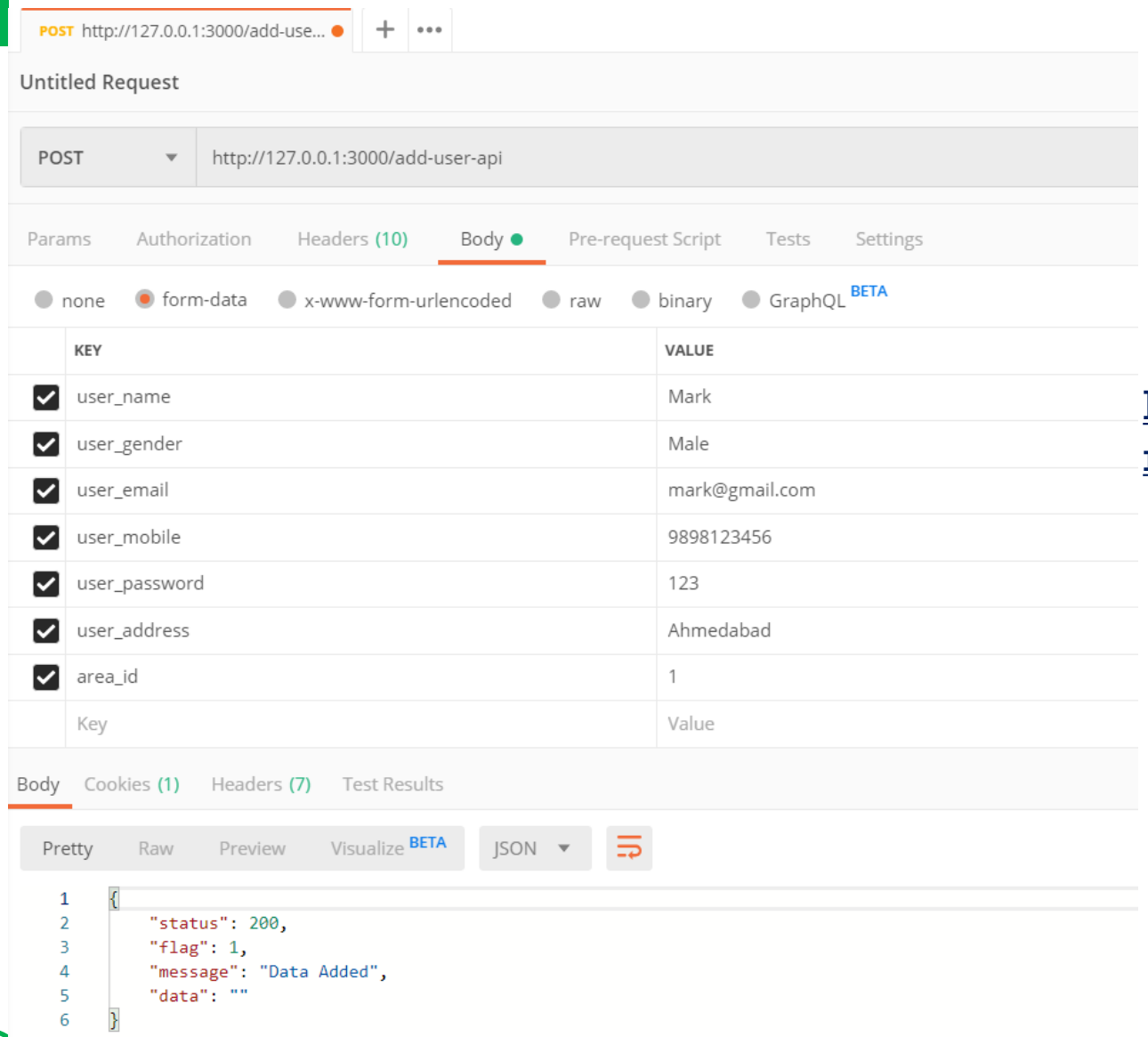

127.0.0.1:3000/get-single-user-a
127.0.0.1:3000/get-single-user-api/1
// 20200418203120
// http://127.0.0.1:3000/get-single-user-api/1
{
"status": 200,
"flag": 1,
"message": "Data Fetch",
"data": [
{
"user_id": 1,
"user_name": "Akash",
"user_gender": "Male",
"user_email": "akash.com@gmail.com",
"user_mobile": 9978621654,
"user_password": "123",
"user_address": "Paldi",
"area_id": 1,
"user_photo": "11.PNG"
}
]
}

Add Record

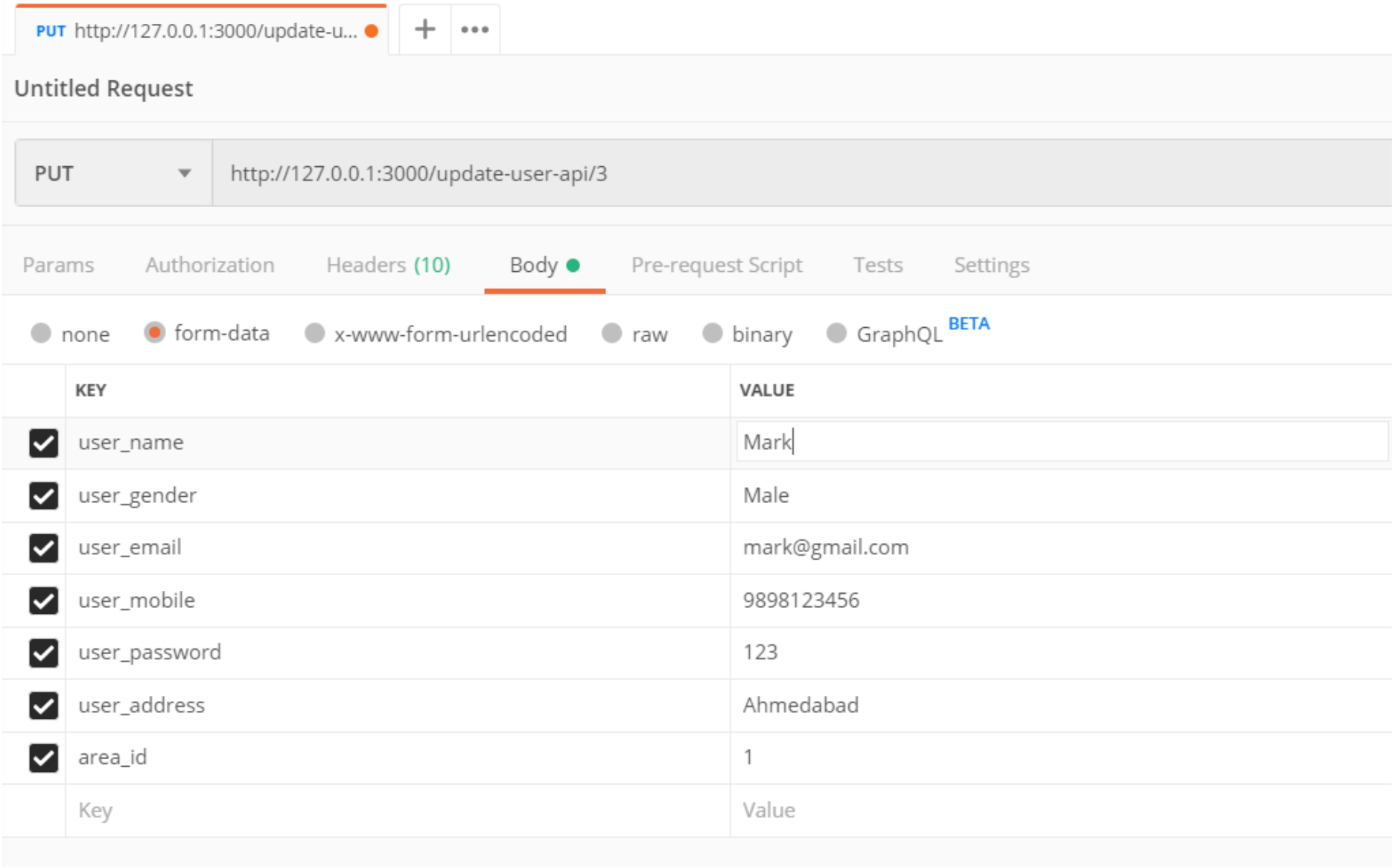
index.js

routes > index.js > router.post('/add-user-api') callback > mybodydata

```
58
59  //Add User API Using Post Method
60  router.post('/add-user-api', function (req, res, next) {
61      console.log("Body Data "+req.body);
62      const mybodydata = {
63          user_name: req.body.user_name,
64          user_gender: req.body.user_gender,
65          user_email: req.body.user_email,
66          user_mobile: req.body.user_mobile,
67          user_password: req.body.user_password,
68          user_address: req.body.user_address,
69          area_id: req.body.area_id
70      }
71      connection.query("insert into tbl_user_master set ?", mybodydata, function (err, result) {
72          if (err) {
73              res.send(JSON.stringify({"status": 500,"flag": 0, "message": "Data Fetch", "data": err}));
74          } else {
75              res.send(JSON.stringify({"status": 200,"flag": 1, "message": "Data Added", "data": ''}));
76          }
77
78      });
79  });
80
```



https://learning.postman.com/docs/sending-requests/requests/


PUT http://127.0.0.1:3000/update-u...
Untitled Request
PUT
http://127.0.0.1:3000/update-user-api/3
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
BETA
KEY
VALUE
user_name
Mark
user_gender
Male
user_email
mark@gmail.com
user_mobile
9898123456
user_password
123
user_address
Ahmedabad
area_id
1
Key
Value

What is the difference between form-data, x-www-form-urlencoded and raw

- These are different Form content types defined by W3C. If you want to send simple text/ ASCII data, then **x-www-form-urlencoded** will work. This is the default.
- But if you have to send non-ASCII text or large binary data, the **form-data** is for that.
- You can use **Raw** if you want to send plain text or JSON or any other kind of string. Like the name suggests, Postman sends your raw string data as it is without modifications. The type of data that you are sending can be set by using the content-type header from the drop down.
- **Binary** can be used when you want to attach non-textual data to the request, e.g. a video/audio file, images, or any other binary data file.

Update

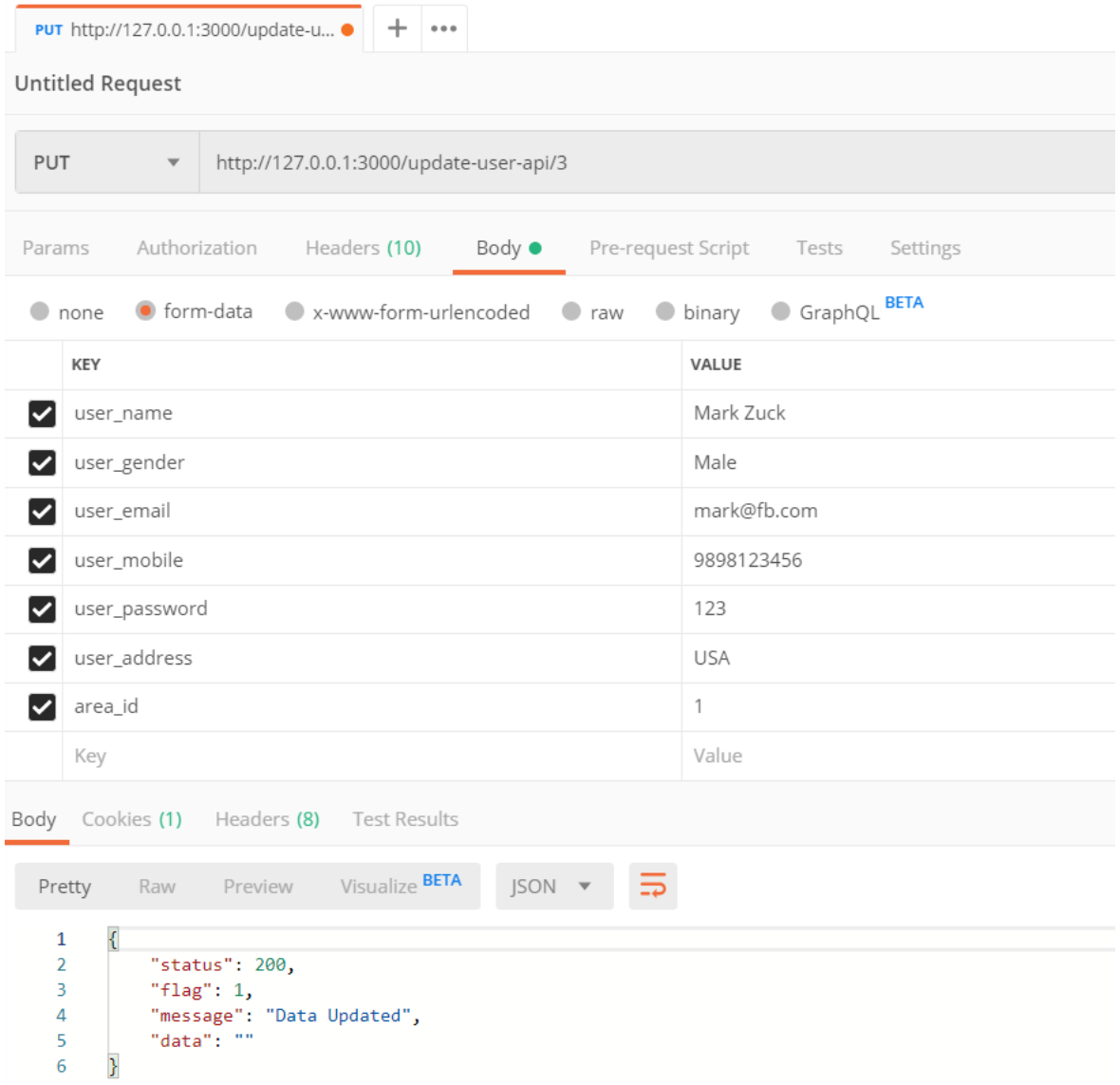
index.js

routes > index.js > router.put('/update-user-api/:id') callback

```
80
81  //Update Record API Using Put method
82  router.put('/update-user-api/:id', function (req, res) {
83      console.log("Parameter ID"+ req.params.id);
84
85      var user_id = req.params.id;
86      var user_name = req.body.user_name;
87      var user_gender = req.body.user_gender;
88      var user_email = req.body.user_email;
89      var user_mobile = req.body.user_mobile;
90      var user_address = req.body.user_address;
91      var area_id = req.body.area_id;
92      // ` is used for multi line string
93      connection.query(`update  tbl_user_master set user_name = ?,user_gender = ?,user_email = ?,user_mobile = ?,
94                        user_address = ?, area_id = ? where user_id = ?  `,
95              [user_name, user_gender, user_email, user_address, area_id, user_mobile, user_id], function (err, respond) {
96              if (err) {
97                  res.send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));
98              } else {
99                  res.send(JSON.stringify({"status": 200,"flag": 1, "message": "Data Updated", "data": ''}));
100             }
101         });
102 });
103
```

Put and Patch

- PUT is a method of modifying resource where the client sends data that updates the entire resource .
- PATCH is a method of modifying resources where the client sends partial data that is to be **updated** without modifying the entire data.


PUT http://127.0.0.1:3000/update-u...
Untitled Request
PUT
http://127.0.0.1:3000/update-user-api/3
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL BETA
KEY
VALUE
user_name
Mark Zuck
user_gender
Male
user_email
mark@fb.com
user_mobile
9898123456
user_password
123
user_address
USA
area_id
1
Key
Value
Body
Cookies (1)
Headers (8)
Test Results
Pretty
Raw
Preview
Visualize BETA
JSON
{
"status": 200,
"flag": 1,
"message": "Data Updated",
"data": ""
}

127.0.0.1:3000/get-all-user-api

127.0.0.1:3000/get-all-user-api


```
        "user_email": "akashtechnolabs@gmail.com",
        "user_mobile": 9978621654,
        "user_password": "123",
        "user_address": "Khadia",
        "area_id": 1,
        "user_photo": "noimage.png"
    },
    {
        "user_id": 3,
        "user_name": "Mark Zuck",
        "user_gender": "Male",
        "user_email": "mark@fb.com",
        "user_mobile": 0,
        "user_password": "123",
        "user_address": "1",
        "area_id": 2147483647,
        "user_photo": "noimage.png"
    }
  ]
}
```


Delete Record

index.js

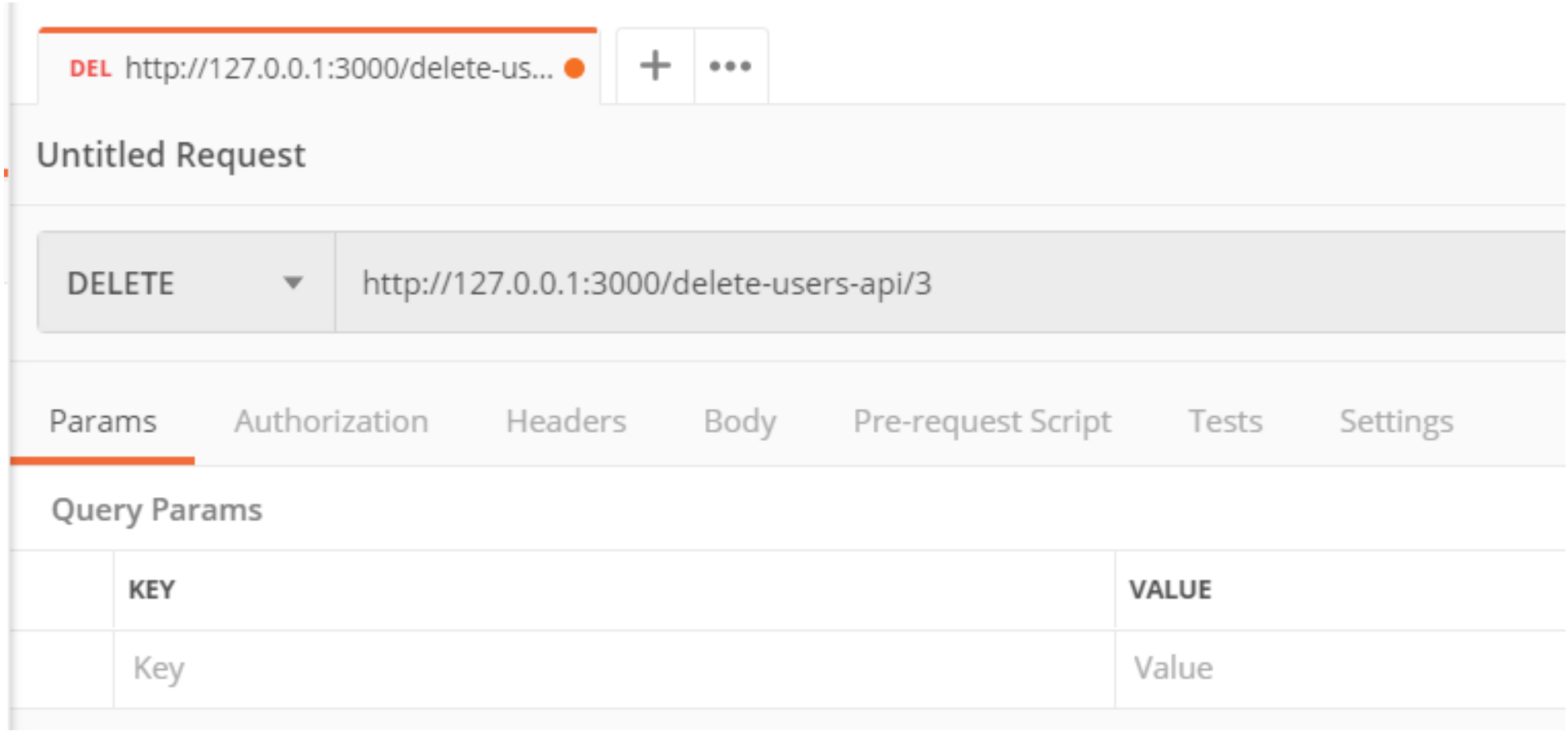
routes > index.js > ...

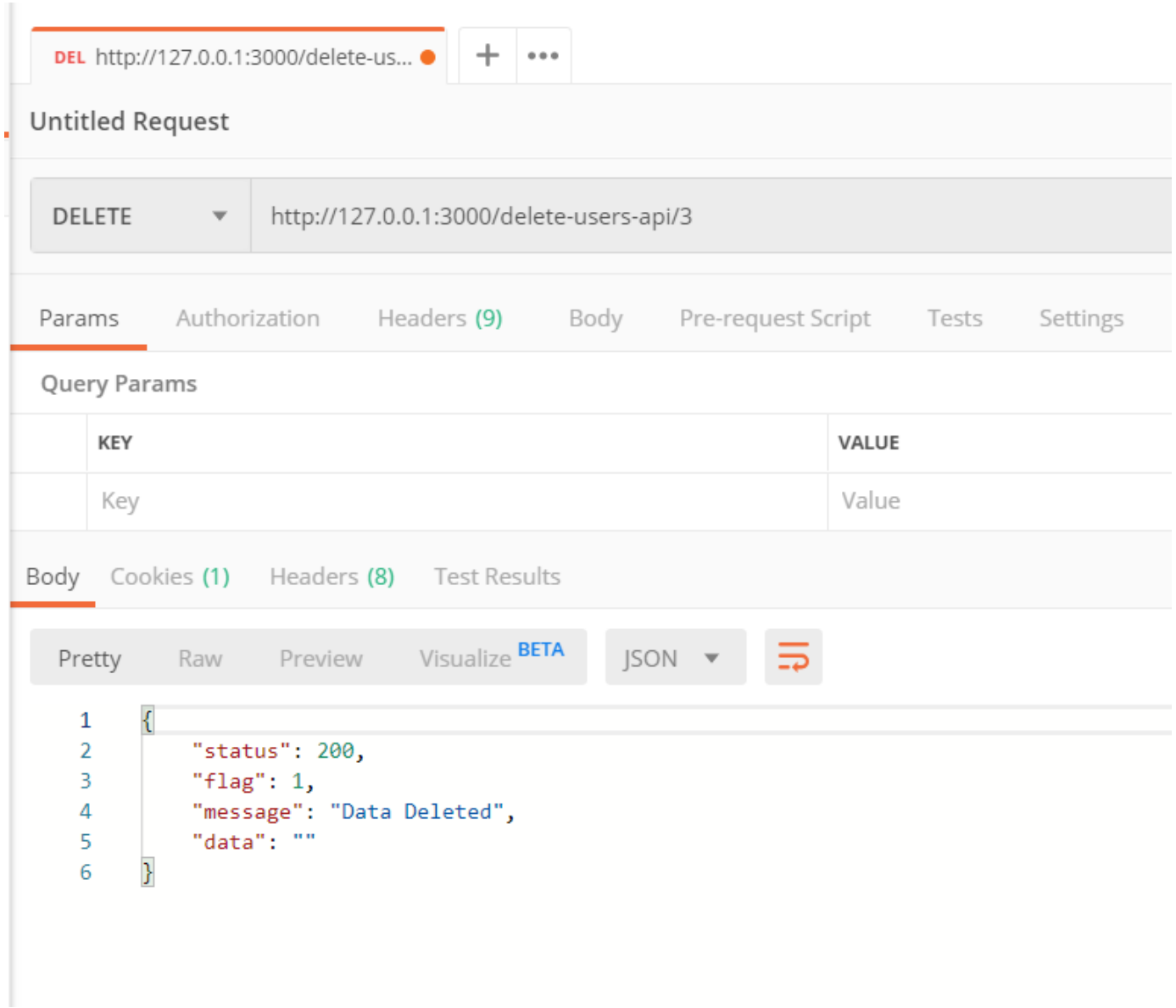
```
106
107  router.delete('/delete-users-api/:id', function (req, res, next) {
108
109      var deleteid = req.params.id;
110      console.log(`Parameter value is ${deleteid}`);
111
112      connection.query("delete from tbl_user_master where user_id = ? ", [deleteid], function (err, rows) {
113          if (err) {
114              res.send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));
115          } else {
116              console.log(rows);
117              res.send(JSON.stringify({"status": 200,"flag": 1, "message": "Data Deleted", "data": ''}));
118          }
119      })
120  });
121
122  module.exports = router;
123
```

index.js

routes > index.js > router.delete('/delete-users-api/:id') callback > connection.query("delete from tbl_user_master where user_id = ? ") callback

```
114  router.delete('/delete-users-api/:id', function (req, res, next) {
115
116      var deleteid = req.params.id;
117      console.log(`Parameter value is ${deleteid}`);
118
119      connection.query("delete from tbl_user_master where user_id = ? ", [deleteid], function (err, rows) {
120          if (err) {
121              res.status(500).send(JSON.stringify({"status": 500,"flag": 0, "message": "Error", "data": err}));
122
123          } else {
124              if(rows && rows.affectedRows >0)
125              {
126                  console.log(rows);
127                  res.status(200).send(JSON.stringify({"status": 200,"flag": 1, "message": "Data Deleted",
128                   "data": rows.affectedRows}));
129              }else{
130                  res.status(200).send(JSON.stringify({"status": 200,"flag": 0, "message": "Data Not Found"}));
131              }
132
133          }
134      })
135  });
```

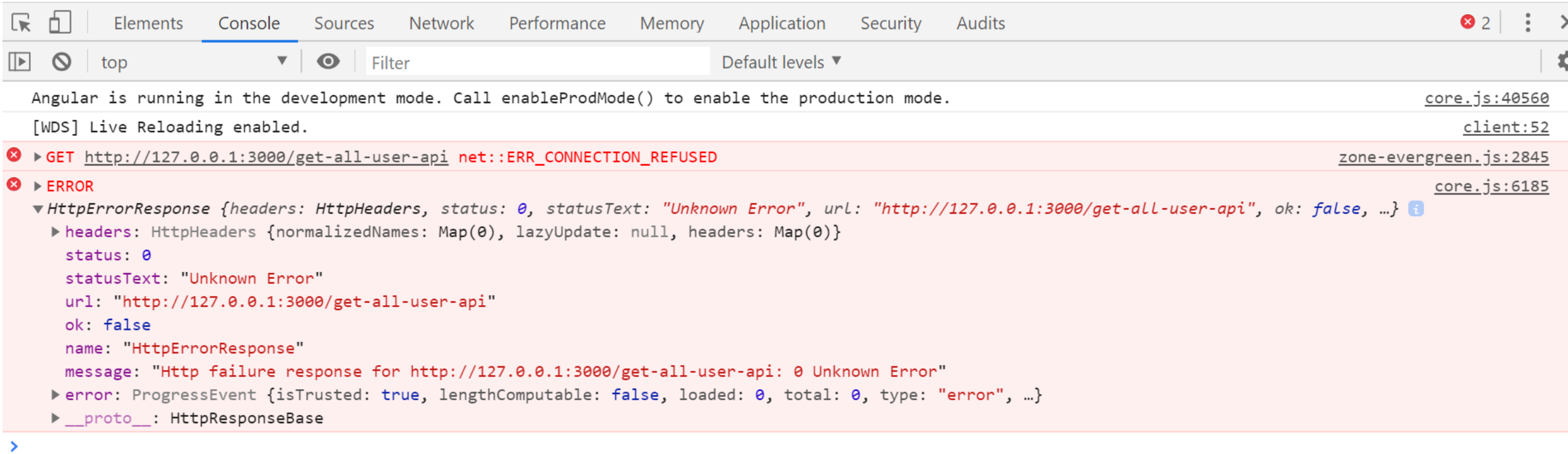

DEL http://127.0.0.1:3000/delete-us...
Untitled Request
DELETE
http://127.0.0.1:3000/delete-users-api/3
Params
Authorization
Headers
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
Key
Value


DEL http://127.0.0.1:3000/delete-us...
Untitled Request
DELETE
http://127.0.0.1:3000/delete-users-api/3
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
Key
Value
Body
Cookies (1)
Headers (8)
Test Results
Pretty
Raw
Preview
Visualize BETA
JSON
{
"status": 200,
"flag": 1,
"message": "Data Deleted",
"data": ""
}

Cors

What is Cors ?

- Cross-Origin Resource Sharing (**CORS**) is a mechanism that uses additional HTTPheaders to tell a browser to let a web application running at one origin (domain) have permission to access selected resources from a server at a different origin



cors

2.8.5 • Public • Published a year ago

Readme | Explore BETA | 2 Dependencies | 6,104 Dependents | 34 Versions

cors

npm v2.8.5 | downloads 15M/month | build passing | coverage 100%

CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.

Follow me (@troygoode) on Twitter!

- Installation
- Usage
  - Simple Usage
  - Enable CORS for a Single Route
  - Configuring CORS
  - Configuring CORS Asynchronously
  - Enabling CORS Pre-Flight
- Configuration Options
- Demo
- License
- Author

Installation

Install

```
> npm i cors
```

Weekly Downloads

3,341,894

Version	License
2.8.5	MIT

Unpacked Size	Total Files
20 kB	6

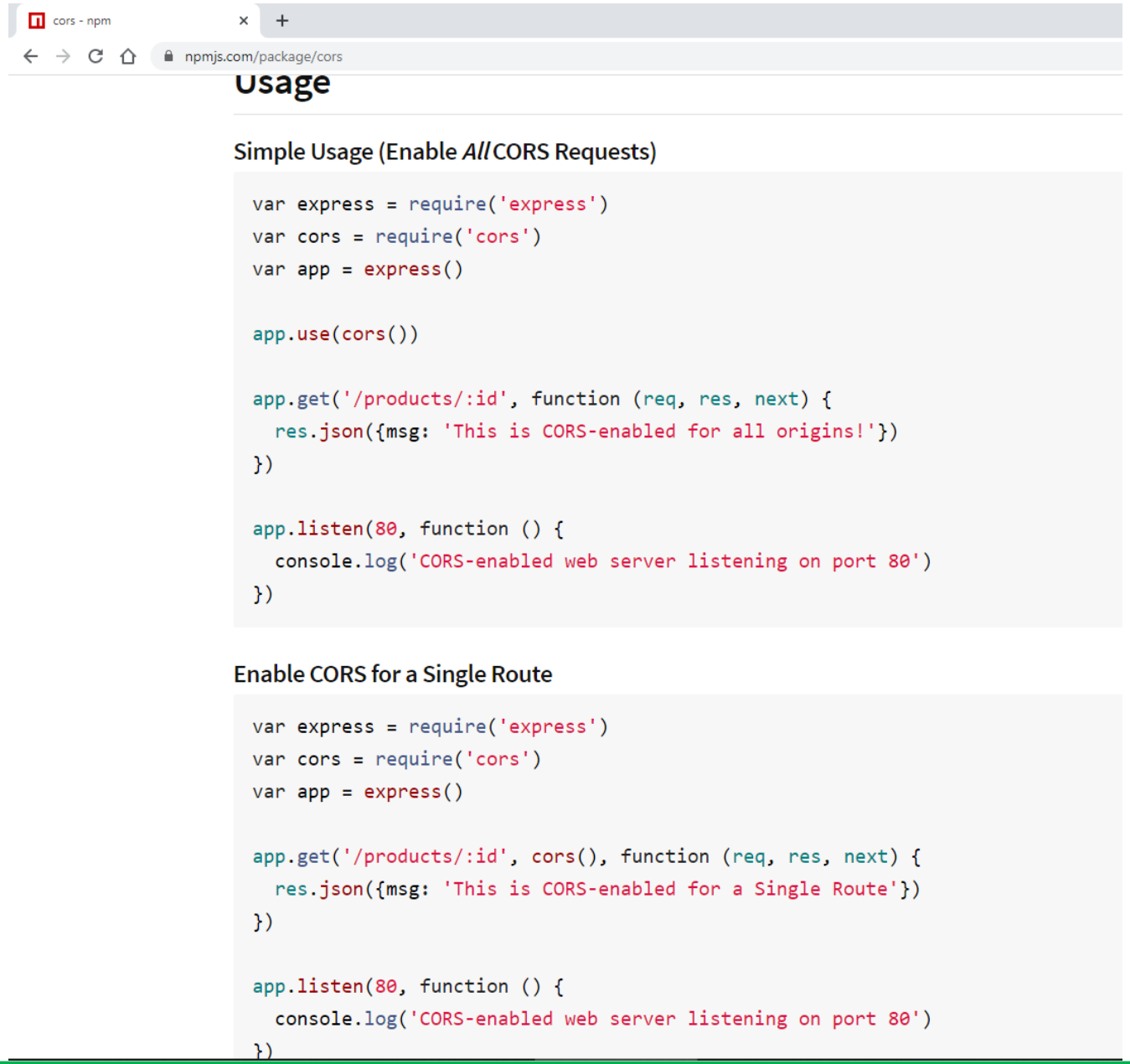
Issues	Pull Requests
7	2

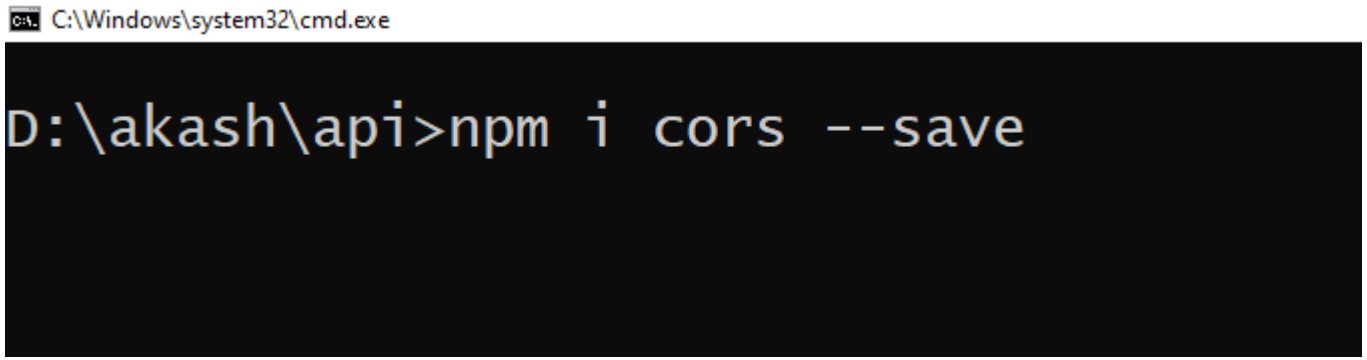
Homepage

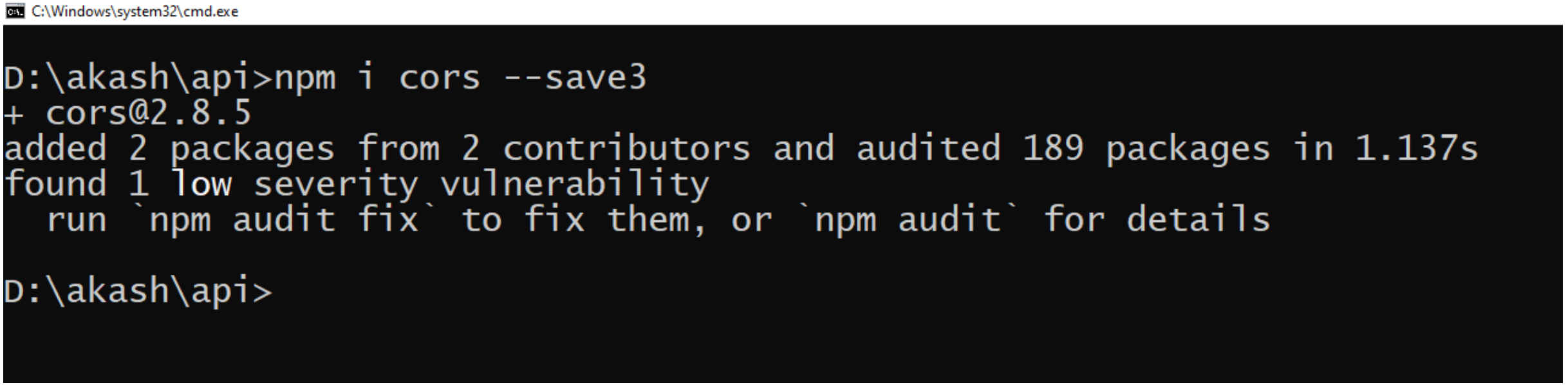
github.com/expressjs/cors#readme

Repository

github.com/expressjs/cors


cors - npm
npmjs.com/package/cors
Usage
Simple Usage (Enable All CORS Requests)
var express = require('express')
var cors = require('cors')
var app = express()
app.use(cors())
app.get('/products/:id', function (req, res, next) {
res.json({msg: 'This is CORS-enabled for all origins!'})
})
app.listen(80, function () {
console.log('CORS-enabled web server listening on port 80')
})
Enable CORS for a Single Route
var express = require('express')
var cors = require('cors')
var app = express()
app.get('/products/:id', cors(), function (req, res, next) {
res.json({msg: 'This is CORS-enabled for a Single Route'})
})
app.listen(80, function () {
console.log('CORS-enabled web server listening on port 80')
})


C:\Windows\system32\cmd.exe
D:\akash\api>npm i cors --save

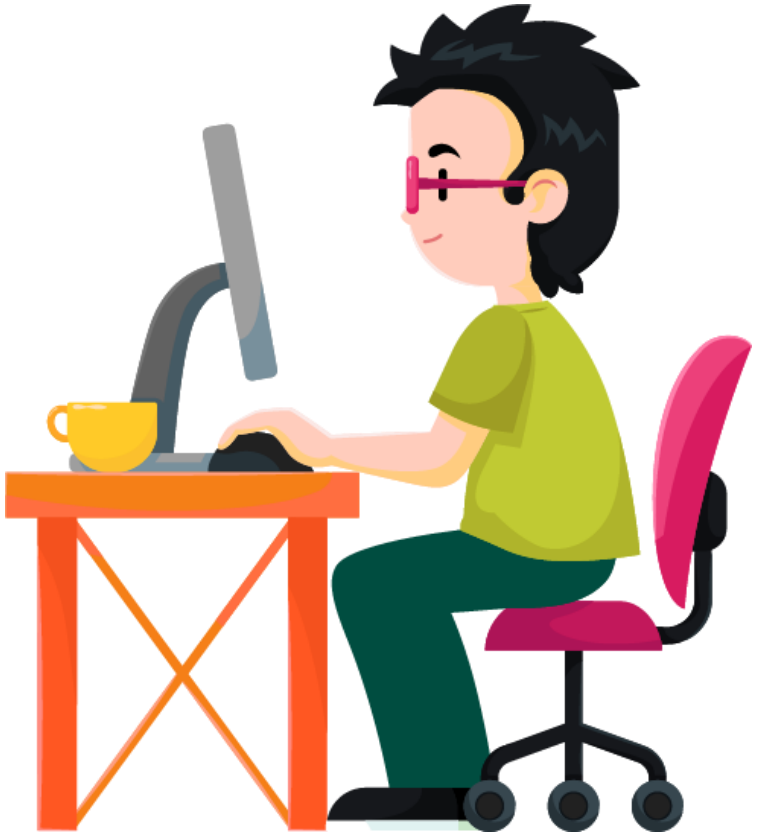

C:\Windows\system32\cmd.exe
D:\akash\api>npm i cors --save3
+ cors@2.8.5
added 2 packages from 2 contributors and audited 189 packages in 1.137s
found 1 low severity vulnerability
run `npm audit fix` to fix them, or `npm audit` for details
D:\akash\api>

app.js

```
var createError = require('http-errors');
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
const fileUpload = require('express-fileupload');

var session = require('express-session');
var flash = require('req-flash');
var cors = require('cors')
var mysql = require('mysql');

var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');

var app = express();

app.use(cors())

// view engine setup
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');


app.use(logger('dev'));
app.use(express.json());
app.use(fileUpload());
app.use(express.urlencoded({ extended: false }));
```

Enjoy ☺

Get Exclusive Video Tutorials



www.aptutorials.com

https://www.youtube.com/user/Akashtips

Get More Details

www.akashsir.com

If You Liked It !

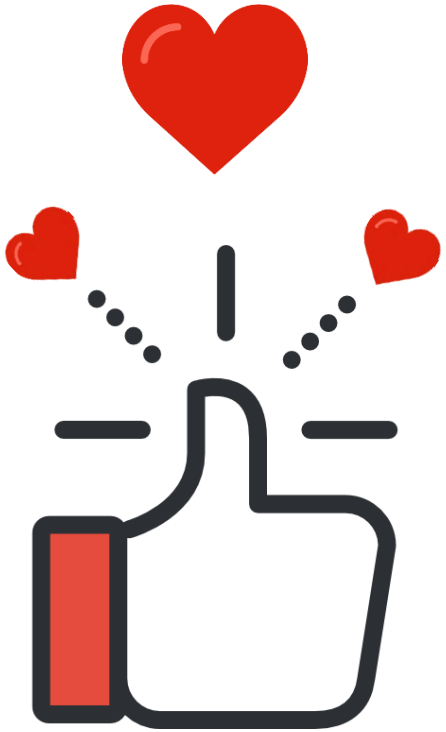
Rating Us Now

Just Dial

https://www.justdial.com/Ahmedabad/Akash-Technolabs-Navrangpura-Bus-Stop-Navrangpura/079PXX79-XX79-170615221520-S5C4_BZDET

Sulekha

https://www.sulekha.com/akash-technolabs-navrangpura-ahmedabad-contact-address/ahmedabad



Connect With Me

Akash Padhiyar
#AkashSir

www.akashsir.com
www.akashtechnolabs.com
www.akashpadhiyar.com
www.aptutorials.com

Social Info

Akash.padhiyar

Akashpadhiyar

Akash_padhiyar

+91 99786-21654

#Akashpadhiyar
#aptutorials