**Node JS**

**Introduction of node js**

Node.js is an open source, cross platform runtime environment for executing server-side JavaScript code. Node.js is built on Chrome’s V8 JavaScript engine and is useful for developing applications such as Internet of Things, Real-time chats, Single Page Applications, Microservices architecture, etc.

Node.js is intended to run on a dedicated HTTP server and to employ a single thread with one process at a time. Node.js applications are event-based and run asynchronously that is, non-blocking. Node processes incoming requests in a constant event stack and sends small requests one after the other without waiting for responses. It's best known as a popular means for JavaScript coders to build real-time Web APIs.

Node.js has a unique advantage because frontend developers that write JavaScript for the browser are now able to write the server-side code. In addition to the client-side code without the need to learn a completely different language.

Node.js makes use of packages and modules. These are the libraries that contain various functions and are imported from npm (node package manager).

**Installation of node js**

Installation steps :

1. Download the installer from the official Node.js site link : <https://nodejs.org/en/download/> based on your machine OS.
2. Run the installer. Follow the installer steps, agree the license agreement and follow the instructions by clicking next button.
3. Restart your system/machine to finish the installation process .

Note: For Detail, you can use node js installation manual.

**Node Package Manager (NPM)**

(NPM) provides two main functionalities −

1. Online repositories for node.js packages/modules which are searchable on search.nodejs.org
2. Command line utility to install Node.js packages, do version management and dependency management of Node.js packages.

> npm init -y : will auto create package.json file in your project.

> npm install : will install all modules’ dependencies in package.json. If the modules already have in your project, npm install can only be used. But,if module don’t have declarations in package.json, npm install + module name : Eg : > npm install express.

After installation, the files you installed will be added in node\_modules file.You can check in node\_modules file.

NPM provide many useful Scripts like npm install, npm start, npm stop, npm config etc.

For detail of npm : <https://docs.npmjs.com/cli/npm.html>

**Writing Sample Node JS Program**

How to write simple HTTP server in Node.js? Let's create an file called server.js...

// Import the HTTP module

var http = require('http');

// We define port to listen to

var PORT = 8080;

// Create a server and pass callback is called someone send an request

var server = http.createServer(function(request, response) {

  // Send response to client and print an message

  response.end('Your first HTTP server written in Node.js');

});

// Listen connections on port defined as PORT

server.listen(PORT, function() {

    // Callback called when server is listening

    console.log("Server listening on: http://localhost:%s", PORT);

});

Run server.js file using command shown below:

> node server.js

Now you can open URL in the browser [https://localhost:8080](https://localhost:8080%20%20/en/download/) - it will print “Your first HTTP server written in Node.js message.”

**API**

[API](https://www.mulesoft.com/platform/api) stands for Application Programming Interface. APIs are widely used by developers for implementing various features in their software. They simply use a simple API call within their software to implement complex features instead of having to code it by themselves. An API is the messenger that delivers your request to the provider that you’re requesting it from and then delivers the response back to you. API takes as input in JSON, HTML or simple text format.

Web API is an interface to either a web server or a web browser. These APIs are used extensively for the development of web applications. These APIs work at either the server end or the client end. When used in the context of [web development](https://en.wikipedia.org/wiki/Web_development_tools), an API is typically defined as a set of specifications, such as [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) (HTTP) request messages, along with a definition of the structure of response messages, usually in an Extensible Markup Language ([XML](https://en.wikipedia.org/wiki/XML)) or JavaScript Object Notation ([JSON](https://en.wikipedia.org/wiki/JSON)) format.

**JSON File**

A JSON file is a file that stores simple data structures and objects in JavaScript Object Notation (JSON) format, which is a standard data interchange format. It is primarily used for transmitting data between a web application and a server. JSON files are lightweight, text-based, human-readable, and can be edited using a text editor.

The package.json file in Node.js is the heart of the entire application. It is basically the manifest file that contains the metadata of the project. Thus, understanding and working with this file becomes very important for a successful Node project development. Sample JSON File :

{

"name": "New User",

"email": "newuser@gmail.com",

"phone": "09421729005",

"address": "Mandalay"

}

**Needed Applications**

Before creating project, we need to install applications used in node project.

1. Visual Studio Code : <https://code.visualstudio.com/docs/?dv=win> ( To Write Code)
2. HeidiSQL : <https://download.freedownloadmanager.org/Windows-PC/HeidiSQL/FREE-9.5.0.5196.html> | <https://www.heidisql.com/> (To Create Database)
3. Postman : <https://www.getpostman.com/downloads/> (To Test Developed API)
4. Git : <https://gitforwindows.org/>

Note: Git is needed to install to run project that pushed on Git. If your project is normal that is not on Git, you can run in terminal and also in command line.

For detail to setup, you can use Student OJT manual file.

**Node JS And MySQL**

MySQL is one of the most preferred databases by the developers as it is open sourced as well as efficient. This is why most prominent programming languages like [Java](https://www.edureka.co/blog/what-is-java/), [Python](https://www.edureka.co/blog/python-tutorial/), [Node.js](https://www.edureka.co/blog/nodejs-tutorial/), etc, provides drivers to access and perform transactions. Before using MySQL, we need to install it from MySQL official website:

<https://dev.mysql.com/downloads/installer/>

And, To use in node project , run the following command to install dependency:

> npm install –save mysql

var mysql = require('mysql'); // require to use mysql

var connection = mysql.createConnection({

  host: 'localhost',

  user: 'root',

  password: 'root',

  database: 'sampleData'

}); // create database connection

connection.connect((err) =>{

  if(err) throw err;

  console.log('Mysql Connected...');

}); // connect mysql database

module.exports = connection; // export database.js

This program file is database.js to connect mysql database.

**Node JS Express Framework**

Express is a minimal and flexible Node.js web application framework that facilitates the management of the flow of data between server and routes in the server-side applications. It is a lightweight and flexible framework that provides a wide range of features required for the web as well as mobile application development.

To install Express.js globally you can use the below command:

> npm install -g express

To install it locally into your project folder, you need to execute the below command:

> npm install express –save

// Import the Express Module

var express = require('express');

var app = express();

// Http get method with (/) route

app.get('/', function(req, res) {

  res.send('Hello World!'); // Take request and Response Hello World!

});

// Listen connections on port 3000

app.listen(3000, function() {

  console.log("Example app listening on port 3000!");

});

If you have Node and Express already installed, you can save this code called app.js and run it in a bash command prompt by calling:

> node app.js

In Above Hello World Express example ,we defined a (callback) route handler function for HTTP GET requests to the site root ('/').

Routes allow you to match particular patterns of characters in a URL, and extract some values from the URL and pass them as parameters to the route handler. Often it is useful to group route handlers for a particular part of a site together and access them. In Express this is achieved by using the express.Router object.

For example, we can create our user route in a module named user.js, and then export the Router object, as shown below:

var express = require('express');

var router = express.Router();

// ADD NEW USER POST

router.post('/add',

function(req, res) {

    res.send("User Creation!");

  }

);

// GET ALL USER

router.get('/all',

function(req, res) {

    res.send("Get All User!";);

  }

);

// UPDATE USER PUT

router.put('/update',

function(req, res) {

    res.send("User Edition!");

  }

);

// DELETE USER

router.delete('/delete',

function(req, res) {

    res.send("Delete User!");

  }

);

module.exports = router;

You can call other response methods : res.json() – to send json response , res.sendFile() – to send File response.

To use that router in main app file, we would then require() the route module (user.js), then call use() on the Express application to add the Router to the middleware handling path. The four routes for (CRUD) will then be accessible C => /user/add, R => /user/all, U => /user/update, D => /user/delete .

Following app.js use user.js route :

var express = require('express');

var app = express();

var UserRoute = require('./user'); // require import user.js file

app.use('/user', UserRoute); // user routes from user.js route file

app.listen(3000, () => {

  console.log('Go to http://localhost:3000 to run!');

});

module.exports = app;

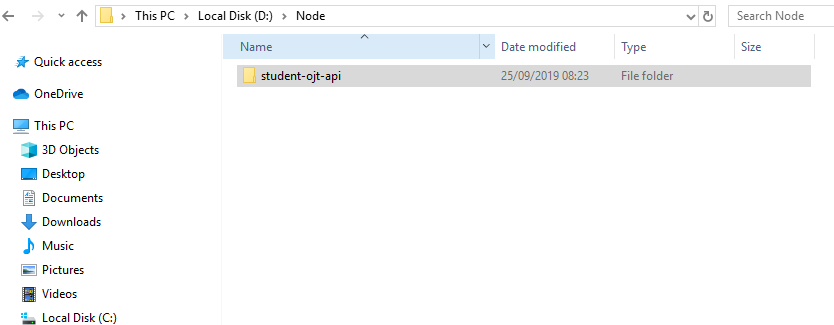
For Detail of Express : [http://expressjs.com](http://expressjs.com/)

**Node JS API Sample Project (CRUD) with Express framework and Mysql Database ( Step-By-Step )**

When needed applications have been finished to download and setup, we can start to build node API project.

**1. Create a project folder**.

Now , I will create “student-ojt-api” in D:\Node\



Before installing dependencies, we need to create package.json file to our project.

**2.To automatically create package.json file**

you can run the following command on the terminal.

> npm init

**3.Install all the dependencies**

that you need by typing the following command in the terminal:

> npm install –save express mysql body-parser nodemon

The above command will install all the dependencies that you need : express, mysql, body-parser and nodemon.

If you open the package.json file, you can see look like this:

{

    "name": "student-ojt-api",

    "version": "1.0.0",

    "main": "app.js",

    "scripts": {

        "dev": "nodemon app.js"

    },

    "devDependencies": {

        "nodemon": "^1.19.2"

    },

    "dependencies": {

        "body-parser": "^1.19.0",

        "cors": "^2.8.5",

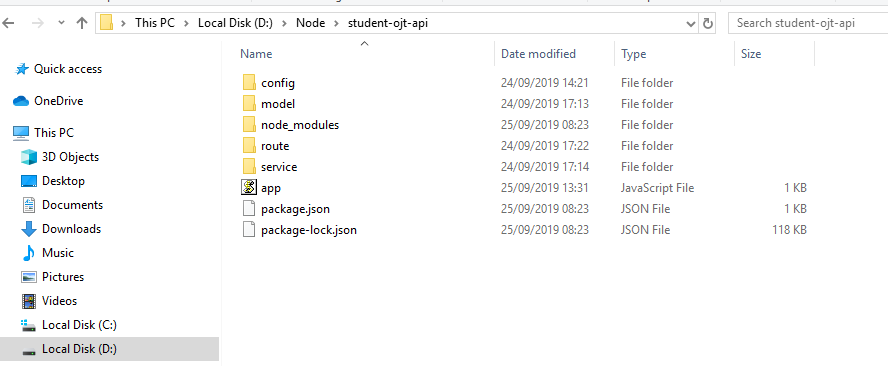
        "express": "^4.17.1",

        "mysql": "^2.17.1",

    }

}

**4.To add new folders**( model , service, route, config ) **and also add app.js.**



**5.To create database.js file and env.json**

For database connection, database.js file in the model folder and env.json file in the config folder to add config data

For **env.json** file : In this file , write config data for Database. In Next, you will also create for another configurations. File Path => D:\Node\student-ojt-api\config\env.json

{

    "Database": {

      "host": "localhost",

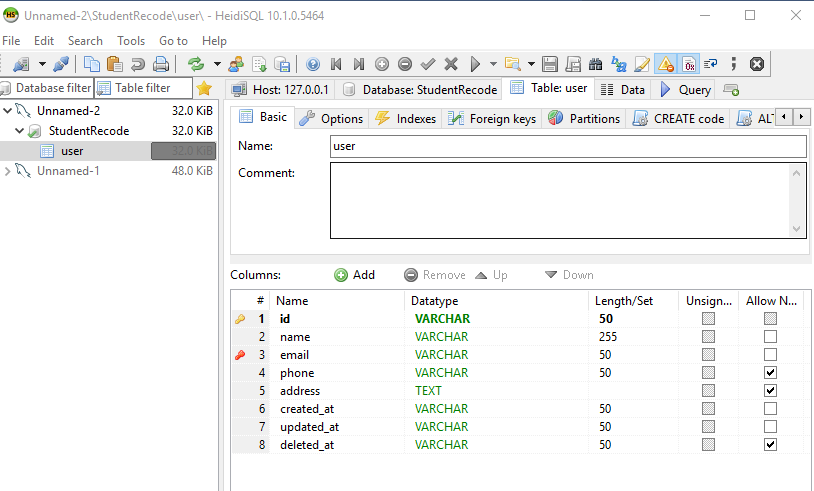
      "user": "root",

      "password": "root",

      "database": "StudentRecord"

    }

}

And then, **database.js** will be created. Before creation, we need to create database ( StudentRecord ). So, Open HeidiSQL , create StudentRecode database, and create user Table. For table fields , See the following :

When you have been created database, you can write **database.js** in your project. Like this,

var util = require('util');

var mysql = require('mysql');

var config = require('../config/env.json'); // use database config file data

var createConnection = mysql.createPool( {

    host: config.Database.host,

    user: config.Database.user,

    password: config.Database.password,

    database: config.Database.database

});

// Ping database to check for common exception errors.

createConnection.getConnection((err, connection) => {

  if (err) {

      if (err.code === 'PROTOCOL\_CONNECTION\_LOST') {

        console.error('Database connection was closed.')

      }

      if (err.code === 'ER\_CON\_COUNT\_ERROR') {

          console.error('Database has too many connections.')

      }

      if (err.code === 'ECONNREFUSED') {

          console.error('Database connection was refused.')

      }

  }

// release connection

  if (connection) connection.release()

    return

})

// Promisify for Node.js async/await.

createConnection.query = util.promisify(createConnection.query)

module.exports = createConnection;

**6.** **Create user\_model.js in model folder** to write insert, update, select, delete query’s methods.

And then, Firstly, We try to create a user. So, in the **user\_model.js** , write **userCreate** method.

var connection = require('./database');

var uuid = require('uuid');//to get auto generate id

var Moment = require("moment"); // to get datetime for created\_at and updated\_at

class UserModel {

    static userCreate(userCreateParam) {

        const userData = {

            id: uuid.v1(),

            name: userCreateParam.name,

            email: userCreateParam.email,

            phone: userCreateParam.phone,

            address: userCreateParam.address,

            created\_at: Moment().format(),

            updated\_at: Moment().format(),

        }

        connection.query('INSERT INTO user SET ?', userData);

        return userData;

    }

}

module.exports = UserModel; // export user model

In this method, For auto generate user id, use uuid. For Times( Created At and Update At), use moment. Therefore, We need to install them. Use following command:

> npm install –save uuid moment

**7.Create index.js in model folder**, to import all models you created.

When you want to use models, you can call index.js and can use models.

See this code:

const UserModel = require("./user\_model"); // call user model (import)

module.exports = {

  User: UserModel,

};

Note: You can create anther models by this flow.

After creating model, need to create service layer.

**8.** **Create user\_service.js in service folder** (to call methods from model and call from routes).

Firstly, **createUser** service method will be write. See:

var Model = require("../model"); // call model

class UserService {

    static createUser(userCreateParams) {

        try {

//call userCreate method from User Model and send user data param

           var createdUser = Model.User.userCreate(userCreateParams);              return createdUser;

        } catch (error) {

           console.log(error);

        }

    }

}

module.exports = UserService;

Note: You can create anther models by this flow.

After creating model and service , need to create route.

**9.** **Create user\_route.js in route folder**, to create routes for user CRUD.

Now, you have been finished user create model and also call from service. So,how to call which route. Need to create user\_route.js and create a route for user creation.

var express = require('express');

var router = express.Router(); // use router of express

var UserService = require('../service/user\_service');

// ADD NEW USER POST

// Use post method of router because of creation.

router.post('/add',

  async function(req, res) {

    var result = await UserService.createUser(req); // call createUser service

    res.json(result); // response data with json format

  }

);

module.exports = router; // export user router

**10. Create index.js in route folder**, to import all routes you created.

When you want to call routes from app.js(main file of project), you can call index.js.

See this code:

var UserRoute = require('./user\_route'); // call user router (import)

module.exports = {

  UserRoute: UserRoute,

};

**11. Modify app.js (main)**

After creating route, need to declare and use routes in app.js. If you forget to use route, that routes will not be know if you call. See this code:

var express = require('express');

var app = express();

var Router = require('./route/index'); // call index file from route folder

app.use(express.json()); // for json input data access, express bodyparser

app.use('/user', Router.UserRoute); // use user routes

app.listen(3000, () => {

  console.log('Go to http://localhost:3000 to run queries!');

});

module.exports = app;

Run Your project with the following command:

> npm run dev

Now, you can test api, with <http://localhost:3000> in Postman.

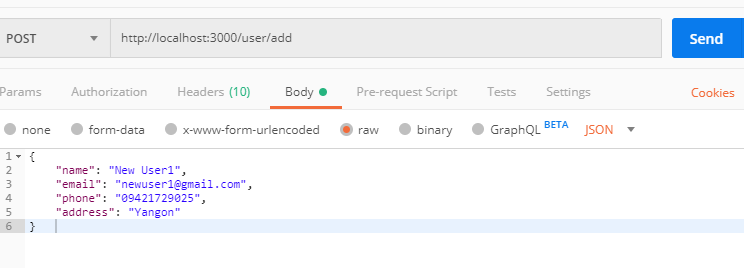
**12. Test API with Postman And Check Data in HeidiSQL**

Method : POST

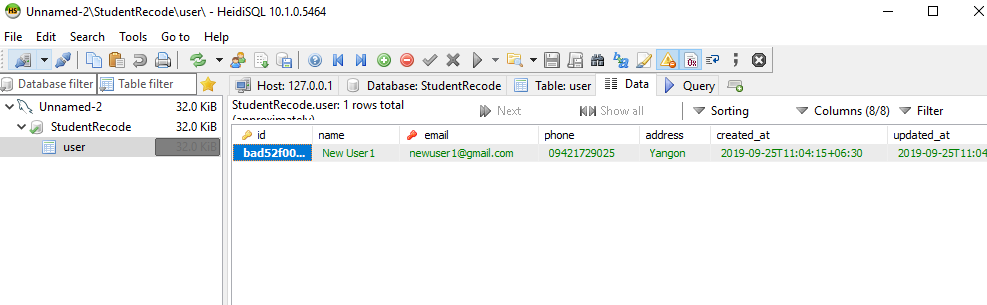
Url : <http://localhost:3000/user/add>

Request data : { "name": "New User1", "email": "newuser1@gmail.com", "phone": "09421729025", "address": "Yangon" }

Click : [ Send ] Button . See the following image:



And then, You can see data, in your database user table. See data by using HeidiSQL.



Now, User Creation have been finished. So, write for user select( get all users).

**13.Write userGetAll() in user\_model.js**

static async userGetAll() {

        var result = await connection.query(`SELECT \* FROM user`);

        return result;

    }

**14.Write getAllUsers() in user\_service.js**

static async getAllUsers() {

        try {

//call userGetAll method from User Model

          const allUser = await Model.User.userGetAll();

          return allUser;

        } catch (error) {

            console.log(error);

        }

    }

**15.Create route to get all user in user\_route.js**

// GET ALL USER

// Use get method to get user

router.get('/all',

  async function(req, res) {

    var result = await UserService.getAllUsers(req); // call getAllUsers service

    res.json(result); // response data with json format

  }

);

**16. Test API with Postman**

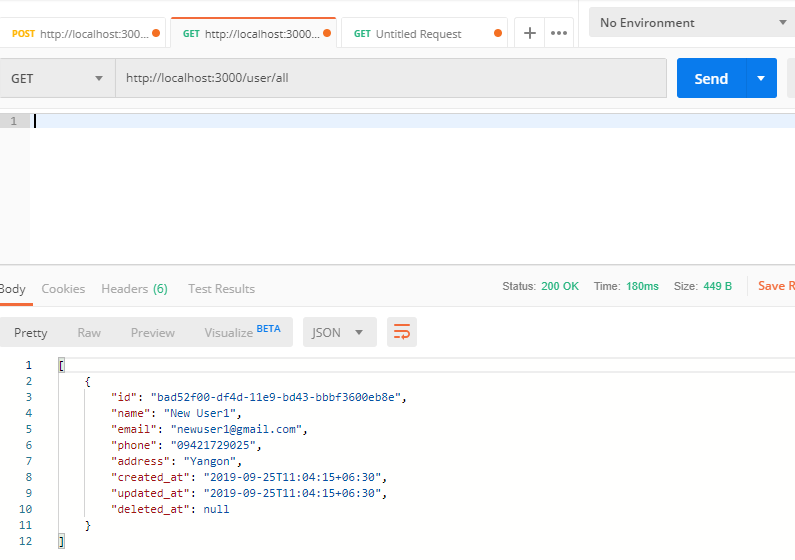
Method : GET

Url : http://localhost:3000/user/all

Request data : -

Click : [ Send ] Button

See the following image:

****

You can see response data in Postman. Finish Get All Users....

And, Start to update User.

**17.Write userUpdate() in user\_model.js**

static async userUpdate(userUpdateParam, userId) {

        const userData = {

            id: userId,

            name: userUpdateParam.name,

            email: userUpdateParam.email,

            phone: userUpdateParam.phone,

            address: userUpdateParam.address,

            created\_at: Moment().format(),

            updated\_at: Moment().format(),

            deleted\_at: null

        };

        const query = "update user SET ? where id ='"+ userId +"'";

        connection.query(query, userData);

        return userData;

    }

**18.Write updateUser() in user\_service.js**

static async updateUser(updateParams) {

        try {

//call userUpdate method from User Model

            const updatedUser = await Model.User.userUpdate(updateParams.body, updateParams.params.id);

            return updatedUser;

        } catch (error) {

            console.log(error);

        }

    }

**19.Create route to update user in user\_route.js**

// UPDATE USER

// Use put method to update

router.put('/update/(:id)',

  async function(req, res) {

    var result = await UserService.updateUser(req); // call updateUser service

    res.json(result); // response data with json format

  }

);

**20. Test API with Postman And Check Data in HeidiSQL**

Method : PUT

Url : [http://localhost:3000/user/update/{user-id}](http://localhost:3000/user/update/%7buser-id%7d) user-id => for data you want to update

Request data :

{

"name": "New User",

"email": "newuser@gmail.com",

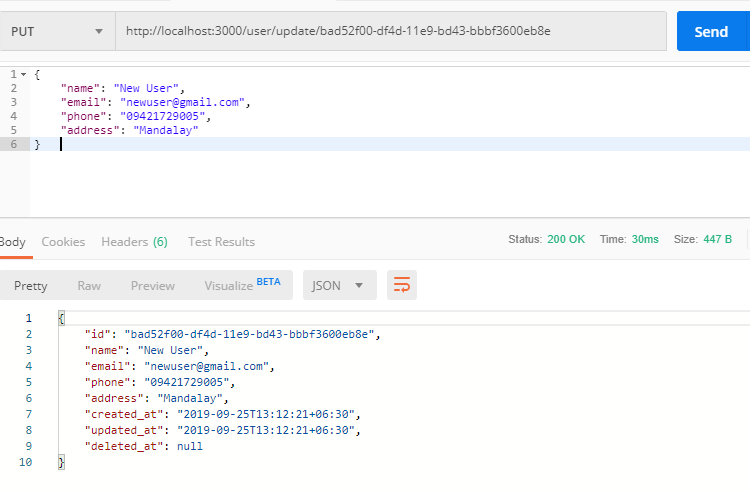
"phone": "09421729005",

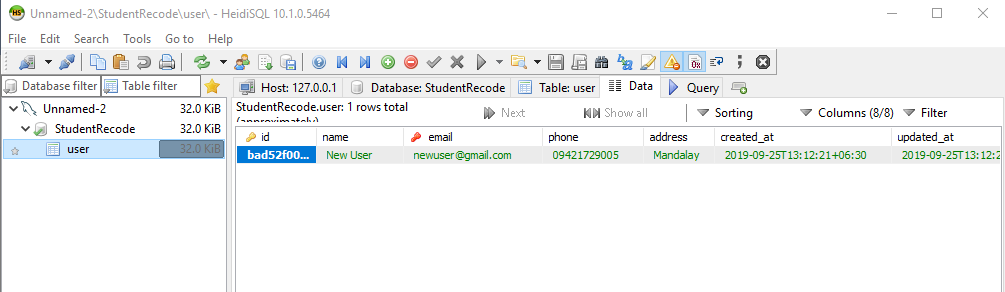
"address": "Mandalay"

}

Click : [ Send ] Button

See the following image:





You can see updated data in Postman and heidiSQL. Finish Update User....

And, Start to Delete User.

**21. Write userDelete() in user\_model.js**

static async userDelete(userId) {

        const result = await connection.query(`DELETE FROM user WHERE id = '${userId}'`);

        return result;

    }

**22.Write updateUser() in user\_service.js**

static async deleteUser(userId) {

        try {

//call userDelete method from User Model

          const deletedUser = await Model.User.userDelete(userId);

          return deletedUser;

        } catch (error) {

            console.log(error);

        }

    }

**23. Create route to delete user in user\_route.js**

// DELETE USER

// Use delete method to delete user

router.delete('/delete/(:id)',

  async function(req, res) {

    var result = await UserService.deleteUser(req.params.id); // call deleteUser service

    res.json(result);

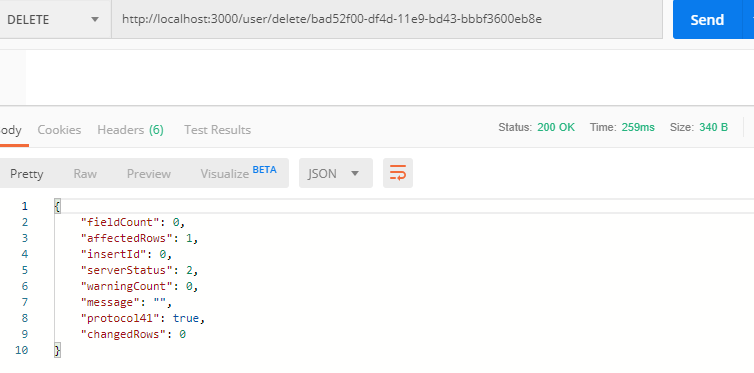
  }

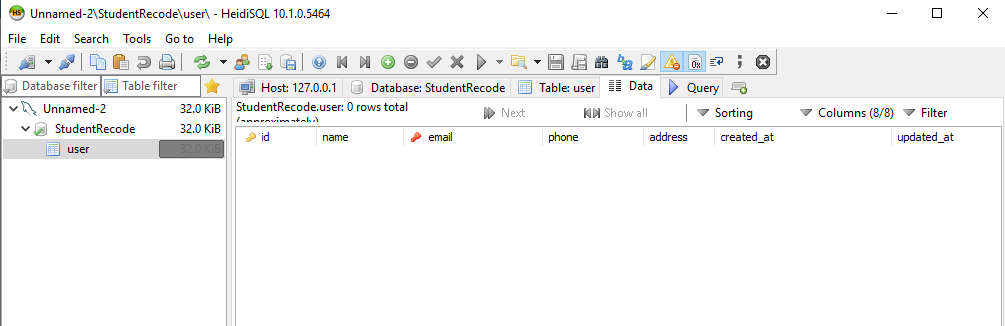
);

**24. Test API with Postman And Check Data in HeidiSQL**

Method : DELETE

Url : [http://localhost:3000/user/delete /{user-id}](http://localhost:3000/user/delete%20/%7buser-id%7d) user-id => for data you delete

Click : [ Send ] Button . See the following image: 



You can see that data have been deleted in heidiSQL. Finish Delete User....

Now, CRUD for User finish. Like above follow, You can add another new modules.

If you want to see code detail and want to download:

URL: <https://github.com/MaySoeThinzarMoe/OJT>

**Conclusion**

**Manual is about Node JS and how to build a CRUD application (Create, Read, Update, Delete) using node.js, express and mysql. You can try to develop new modules and can build new project by using this. I think you also understand the use of Postman and HeidSQL. If you want to choose anthor api testing tool, REST-Assured , SoapUI, Jmeter, KarateDSL, Fiddler, etc....** are testing tool. I can also learn detail of node JS by entering following reference links.

Thank a lots for your time.

**Reference**

* <https://www.w3schools.com/nodejs/>
* <https://nodejs.dev/how-much-javascript-do-you-need-to-know-to-use-nodejs>
* <https://www.javaworld.com/article/2079190/6-things-you-should-know-about-node-js.html>
* <https://www.edureka.co/blog/nodejs-tutorial/>
* <https://www.merixstudio.com/blog/what-you-need-know-about-nodejs-short-technology-guide/>
* <https://www.nodebeginner.org/>
* <https://www.zibtek.com/blog/types-of-applications-you-can-build-with-node-js/>