Problem Statement and Solution

**Part I**

1. Start with populating a MySQL DB with some users. Ideally, the DB would be hosted in the cloud. You can use https://www.freemysqlhosting.net/ which gives 5-MB free database and 1-week validity which is sufficient for the demo. You are free to use any other cloud-hosted MySQL DB.
2. You can access a lot of dummy user data at https://reqres.in/ which is Free REST API. This can be done via some sort of automation script (Python or JS).

**Part II**

1. Create a backend using NodeJS and ExpressJS as middleware.
2. Make a very basic CRUD app, which allows operations like GET, ADD, DELETE users and UPDATE user details.

**Solution**

**Overview**

To Build Users RESTful CRUD (Create, Read, Update, Delete) API with NodeJS, ExpressJS as middleware and MySQL

**Required Resources and applications**

* **MySQL:** It is an open-source relational database management system (RDBMS). It organizes data into one or more data tables in which data may be related to each other. The relations help to structure the data. SQL is a Structured Query language, programmers use to create, modify and extract data from the relational database, as well as control user access to the database.

*We can use free MySQL hosting here:* [*https://www.freemysqlhosting.net/*](https://www.freemysqlhosting.net/)

* **NODE JS:** Node.js is a cross-platform, open-source server environment that is a back-end JavaScript runtime environment, runs on the V8 JavaScript Engine, and executes JavaScript code outside a web browser.

*We can download node.js from* [*https://nodejs.org/en/download/*](https://nodejs.org/en/download/)

* **Express JS:** Express.js, or simply Express, is a back end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs.

**Middleware:** It's a function that holds the request object, the response object, and the middleware function. Middleware can also send the response to the server before the request. The next middleware function is commonly represented as a variable named next. Simply middleware is a function that can only be applied using routes.

* **XAMPP:** XAMPP is a free and open-source cross-platform web server solution stack package developed by Apache Friends, consisting mainly of the Apache HTTP Server, MySQL/MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages.

*We can download XAMPP* [*https://www.apachefriends.org/*](https://www.apachefriends.org/)

* **Postman:** The Postman API endpoints enable you to integrate Postman within your development toolchain. You can add new collections, update existing collections, update environments, and add and run monitors directly through the API. This enables you to programmatically access data stored in your Postman account.

*We can download Postman* [*https://www.postman.com/*](https://www.postman.com/)

* **IDE (integrated development environment)** is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of at least a source code editor, build automation tools, and a debugger.

**Visual Studio Code** is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux. It comes with built-in support for JavaScript, Typescript and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, PHP, Go, .NET).

*We can download Visual Studio Code* <https://code.visualstudio.com/>

**Environment check and setup**

We have check that required applications are installed or not. We’ll check Node, NPM and MySQL to installed on machine.

**Creating Our Project**

Now we’ll create our project. First of all, choose file system / drive on where we’ll create our project. After that we create a directory name userAPI. Then navigate to userAPI directory. Command are as below

// Create directory for new project named userAPI

mkdir userAPI

// then Navigate to userAPI  
cd userAPI

Now initialize the node js using npm init

npm init

That will ask a few thing like project name, version, description, author name etc. follow the setup, if you do not to add info manually then use npm init -y which will generate package.json file automatically.

**Install express and other dependencies**

Now we have to install express, body-parser, and MySQL modules in our application. Let’s install them by following in single command:

npm install express body-parser mysql

**Setting up the web server**

Node. js provides capabilities to create your own web server which will handle HTTP requests asynchronously. As we earlier we have created enter point of application is index.js, we will create index.js file using vscode.

**Connect to the database**

We’ll keep all the configurations for MySQL connection in the index.js file.

**Create the Express JS middleware functions for CRUD**

// Create directory for new project named userAPI

const express = require('express');

const mysql = require('mysql');

const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.json());

const connection = mysql.createConnection({

host: 'localhost',

user: 'root',

password: '',

database: 'ocsDB'

});

connection.connect((err) => {

if (err) throw err;

console.log('Connected to MySQL database');

});

// GET all users

app.get('/users', (req, res) => {

const sql = 'SELECT \* FROM users';

connection.query(sql, (err, results) => {

if (err) throw err;

res.json(results);

});

});

// GET user by id

app.get('/users/:id', (req, res) => {

const id = req.params.id;

const sql = `SELECT \* FROM users WHERE id = ${id}`;

connection.query(sql, (err, result) => {

if (err) throw err;

res.json(result);

});

});

// POST new user

app.post('/users', (req, res) => {

const user = req.body;

const sql = 'INSERT INTO users(first\_name,last\_name,email,avatar) values(?,?,?,?)';

connection.query(sql, [user.first\_name,user.last\_name,user.email,user.avatar], (err, result) => {

if (err) throw err;

res.json({ message: 'User created successfully!', id: result.insertId });

});

});

// PUT update user by id

app.put('/users/:id', (req, res) => {

const id = req.params.id;

const user = req.body;

const sql = `UPDATE users SET first\_name = ?,last\_name =?, email = ?,avatar =? WHERE id = ${id}`;

connection.query(sql, [user.first\_name,user.last\_name, user.email,user.avatar], (err, result) => {

if (err) throw err;

res.json({ message: `User with id ${id} updated successfully!` });

});

});

// DELETE user by id

app.delete('/users/:id', (req, res) => {

const id = req.params.id;

const sql = `DELETE FROM users WHERE id = ${id}`;

connection.query(sql, (err, result) => {

if (err) throw err;

res.json({ message: `User with id ${id} deleted successfully!` });

});

});

// Start server

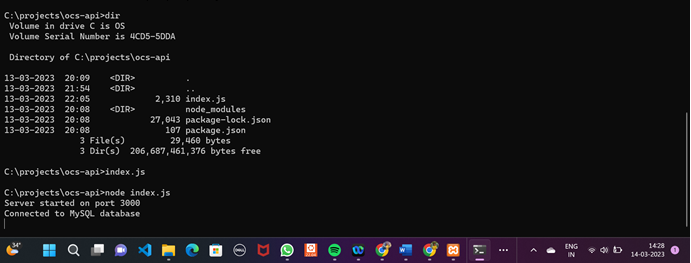
app.listen(3000, () => {

console.log('Server started on port 3000');

});

Now we’ll run the web server using node command. For that we can run

node index.js

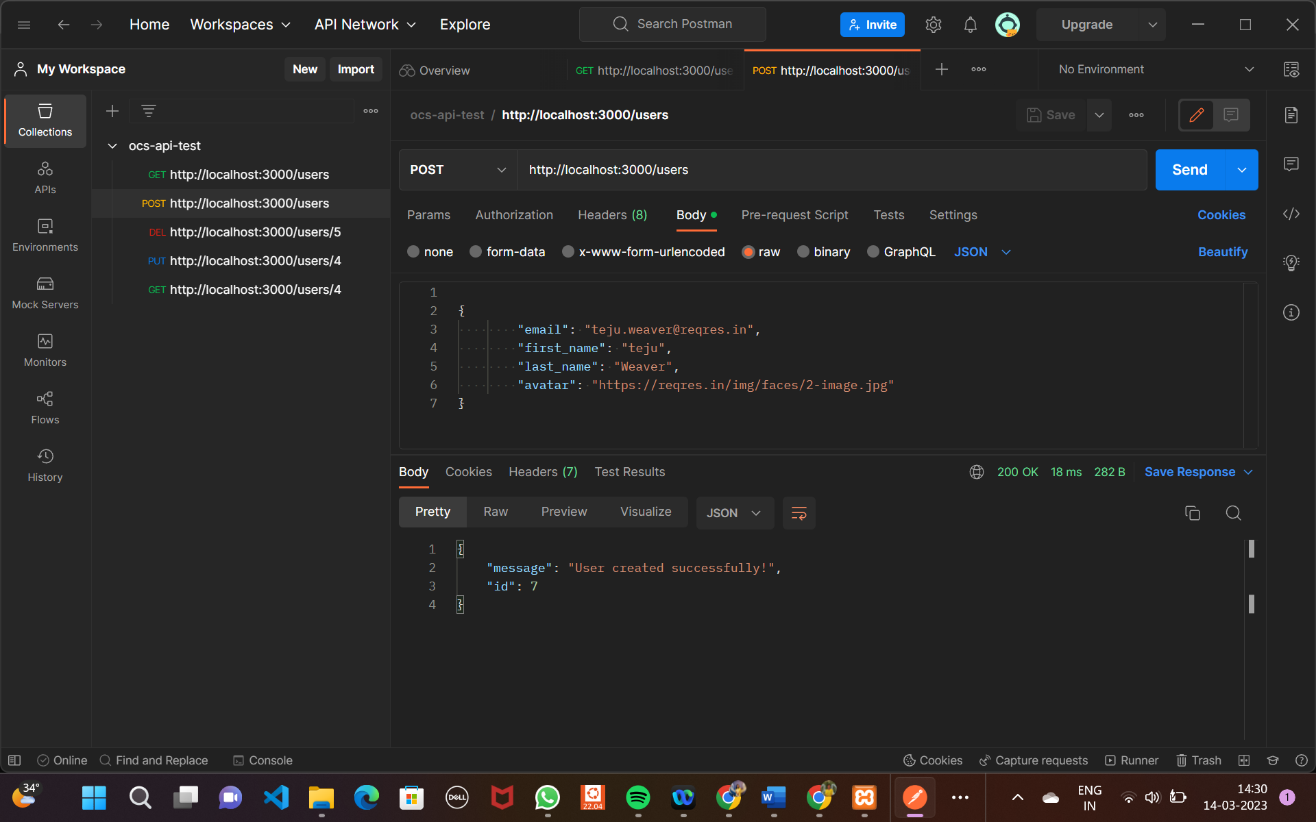


**API end points**

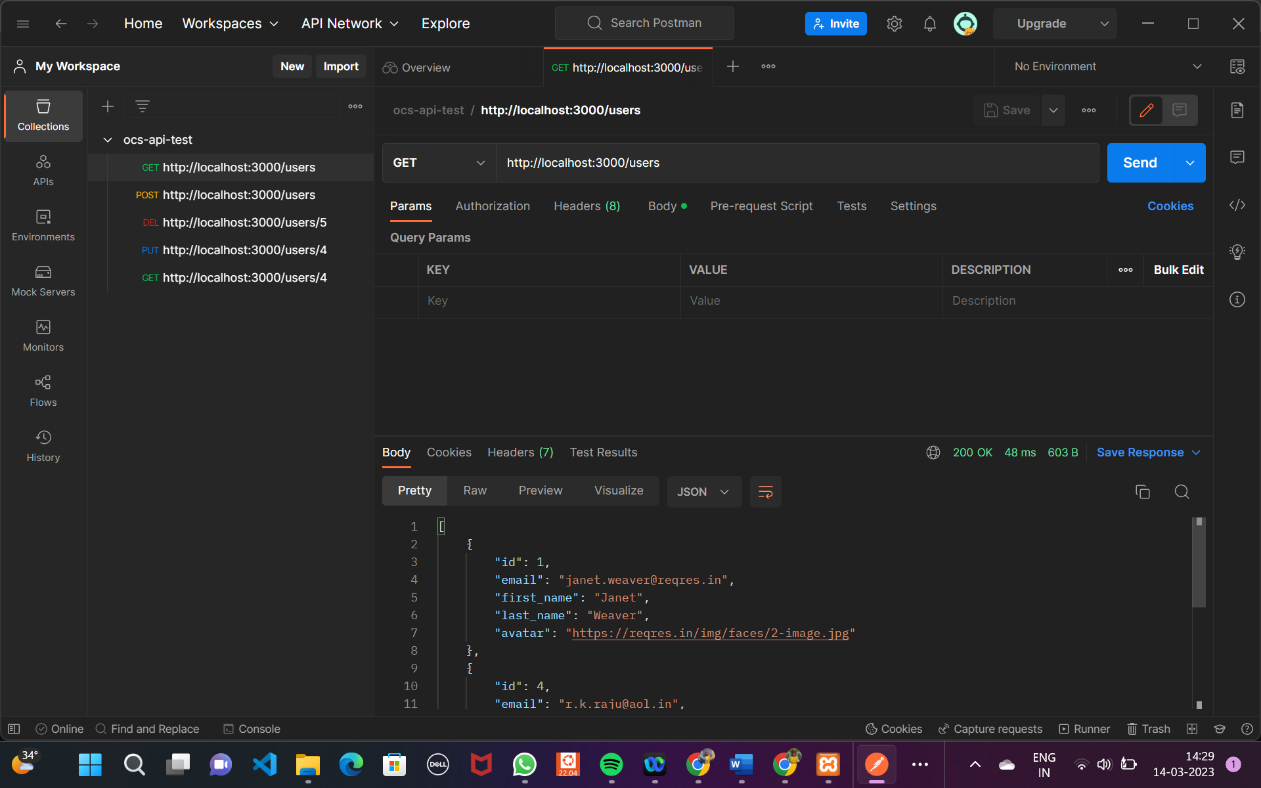
* *GET /users: will give all users stored in database*
* *GET /users/id: will give a specific user with id.*
* *POST /users: user can create a new user*
* *PATCH /users/id: update a user partially*
* *DELETE /users/id: delete a user*
* *PUT /users/id: update a user completely*

**Now Time to set APIs in post**

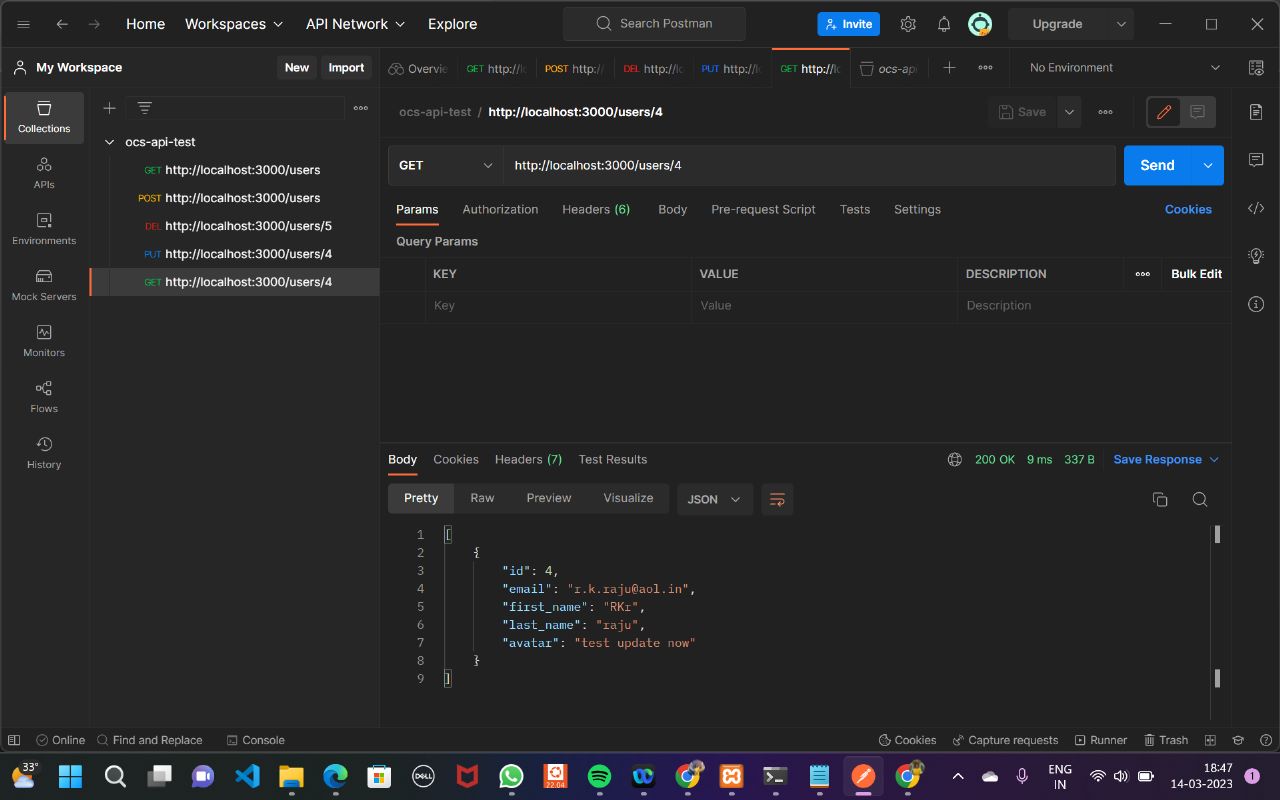
**Creating a new user http://localhost:3000/users using POST method**



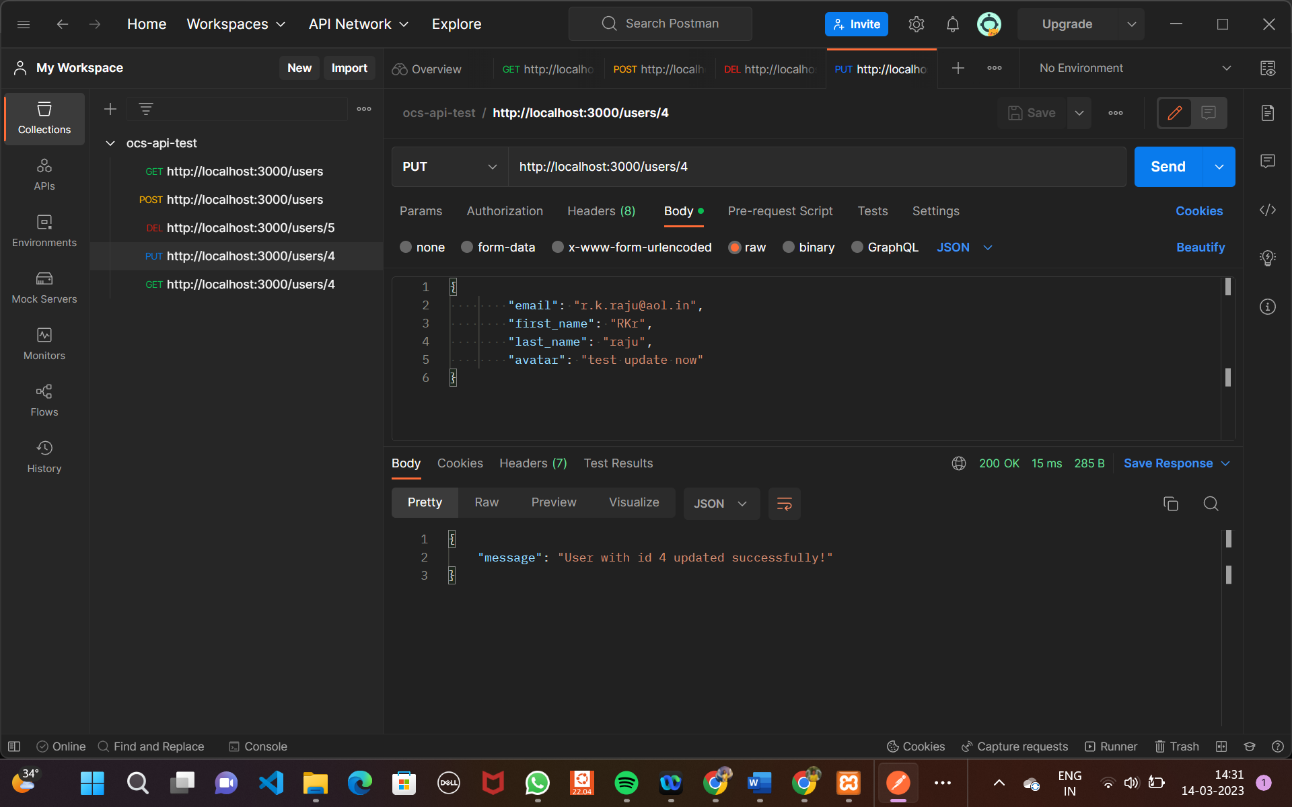
**Get all users list http://localhost:3000/users using GET method**



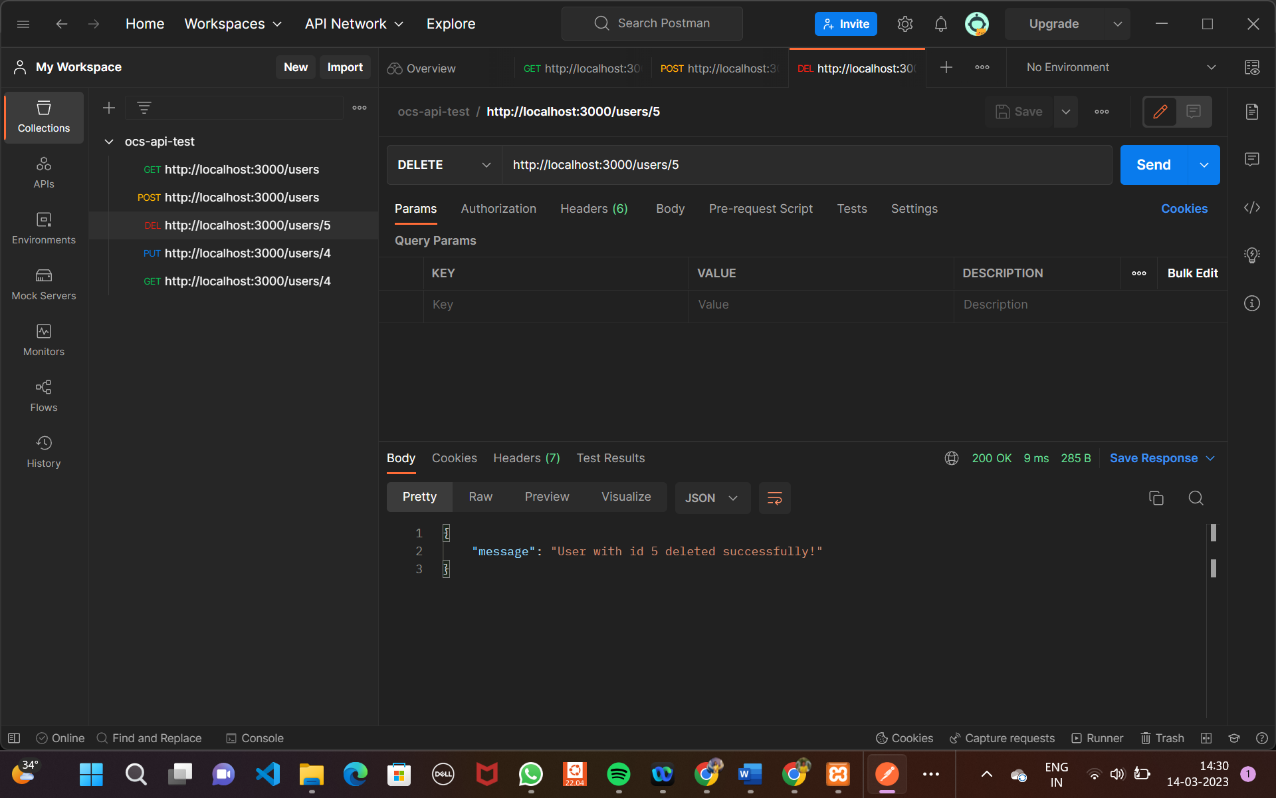
**Get user by specific http://localhost:3000/users/id using GET method**



**Update user by specific id http://localhost:3000/users/id using PUT method**



**Delete user by specific id http://localhost:3000/users/id using DELETE method**

****

**Conclusion**

In this document, we had learnt how to make APIs using Node, Express and MySQL.

You can find here git repository. <https://github.com/rtejasri/users-api>

Thanks for reading.