**Practical No 10**

**Aim: Creating servers using express and handling http methods**

Initial Commands to run on the console:

npm init

init express http nodemon fs readline mongoose

To run a file, node filename.js or nodemon filename.js

Q1 Create a node.js program using express.js that serves a list of users from a JSON file. The program should define an API an endpoint to retrieve the details of users in JSON format.

1. Display details of all users b. Display details based on its parameters such as id

Q2 Update the data of the user with user id 2 using patch method

Q3. Delete a user whose id is 5. Send a response after performing delete operation.

🡪

Users.json

[

{

"id": 1,

"name": "Alice",

"age": 30

}, …..]

Index.js

const express = require('express');

const app = express();

const fs = require('fs');

const usersData = JSON.parse(fs.readFileSync('users.json'));

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

res.json(usersData);

});

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

const userId = req.params.id;

const user = usersData.find(user => user.id === parseInt(userId));

if (!user) {

res.status(404).send('User not found');

} else {

res.json(user);

}

});

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

const users = JSON.parse(fs.readFileSync('users.json'));

const updatedUsers = users.map(user => {

if (user.id === 2) {

user.name = 'Updated Name';

}

return user;

});

fs.writeFileSync('users.json', JSON.stringify(updatedUsers, null, 2));

res.json({ message: 'User with id 2 updated successfully' });

});

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

const users = JSON.parse(fs.readFileSync('users.json'));

const updatedUsers = users.filter(user => user.id !== 5);

fs.writeFileSync('users.json', JSON.stringify(updatedUsers, null, 2));

res.json({ message: 'User with id 5 deleted successfully' });

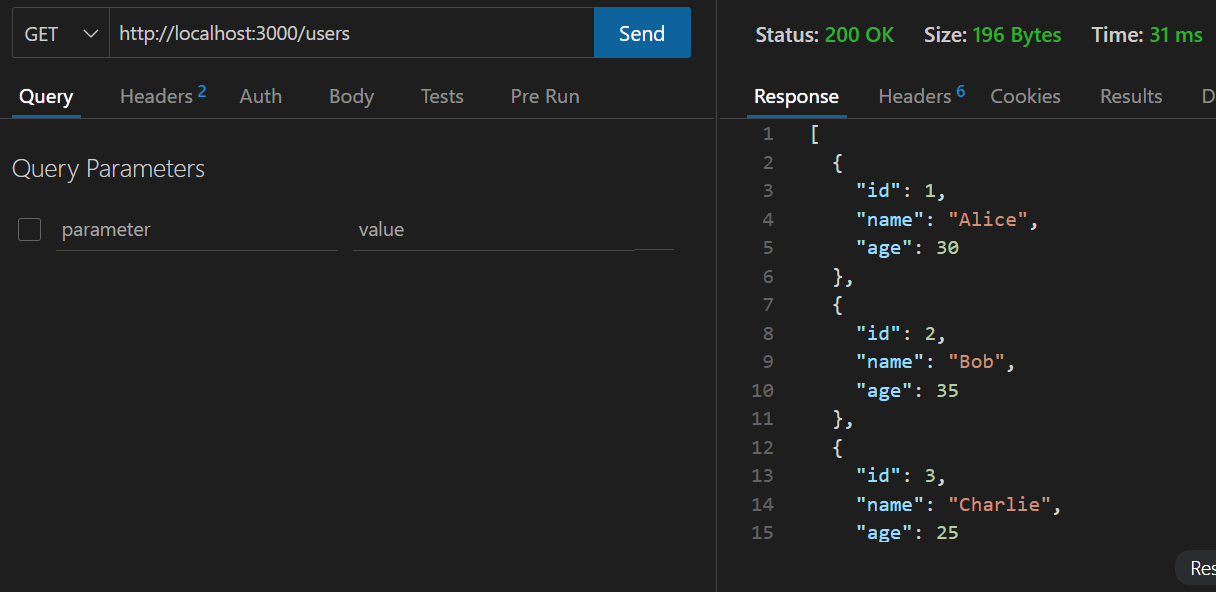
});

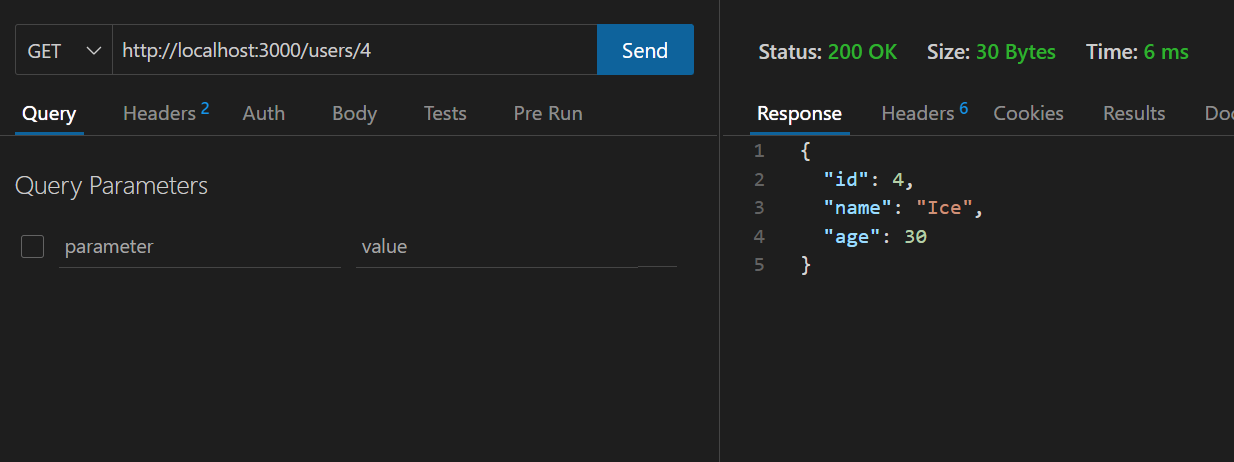
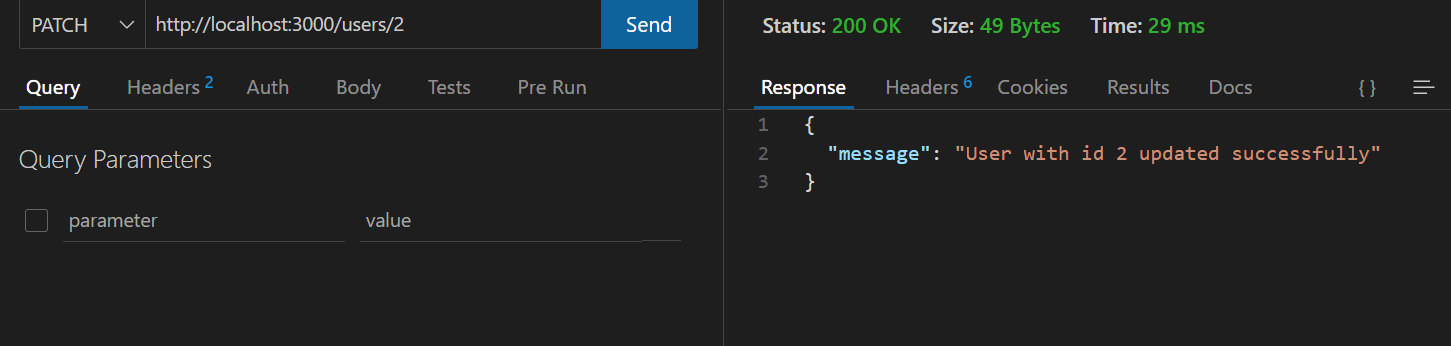
const PORT = 3000;

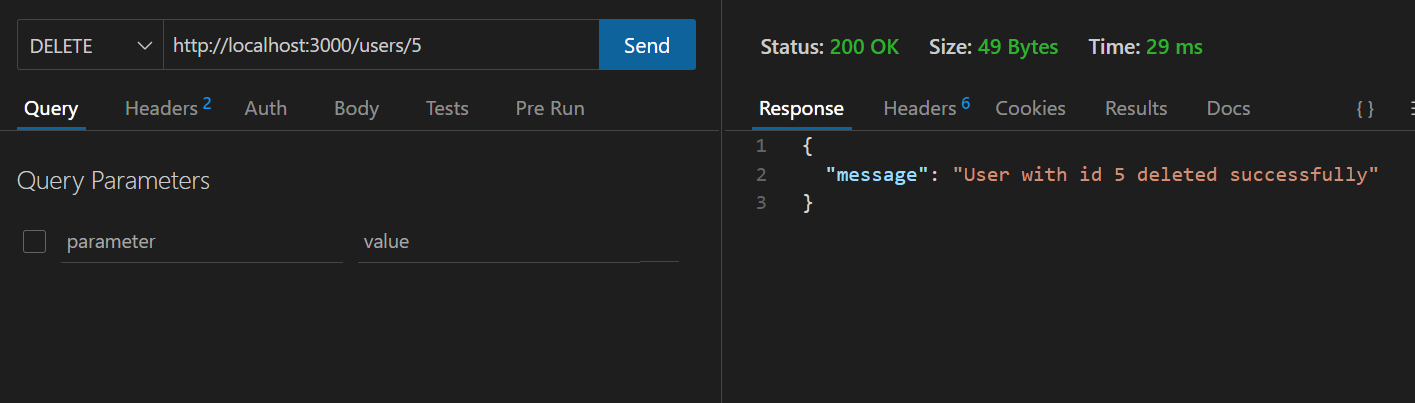
app.listen(PORT, () => {

console.log(`Server is running on port ${PORT}`);

});







Q4 Connect node.js server to Mongodb compass. Create a database, create a collection and insert some sure input data

const { MongoClient } = require('mongodb');

const uri = 'mongodb://localhost:27017';

const dbName = 'myDatabase';

async function main() {c

const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

try {

await client.connect();

console.log('Connected to the MongoDB server');

const db = client.db(dbName);

console.log(`Database "${dbName}" created`);

const collectionName = 'myCollection';

const collection = db.collection(collectionName);

console.log(`Collection "${collectionName}" created`);

const data = [

{ name: 'John', age: 30 },

{ name: 'Jane', age: 25 },

{ name: 'Doe', age: 40 }

];

const result = await collection.insertMany(data);

console.log(`${result.insertedCount} documents inserted`);

} finally {

await client.close();

console.log('Connection closed');

}

}

main().catch(console.error);

