**Practical No 10**

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

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

**Code:**

const express= require('express');

const fs= require('fs');

const app=express();

const port= 3000;

const users=JSON.parse(fs.readFileSync('users.json','utf-8'));

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

res.send(users);

})

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

const user=parseInt(req.params.id);

const userId=users.find((u)=>u.id===user);

if(userId){

res.json(userId);

}

else{

res.send(`User with ${userId} is not found`);

}

})

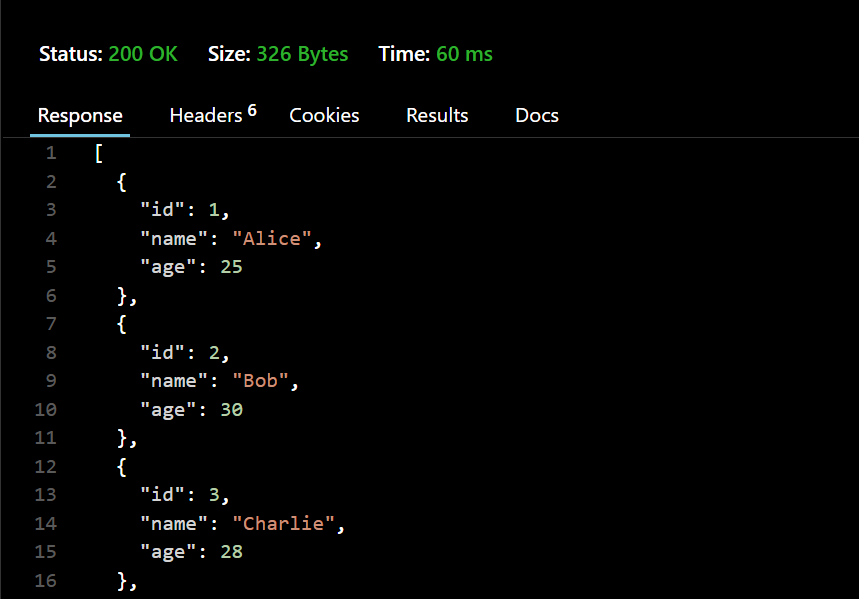
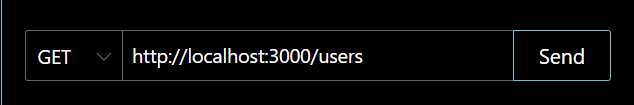
app.listen(port,()=>{

console.log(`App is listening on port ${port}`);

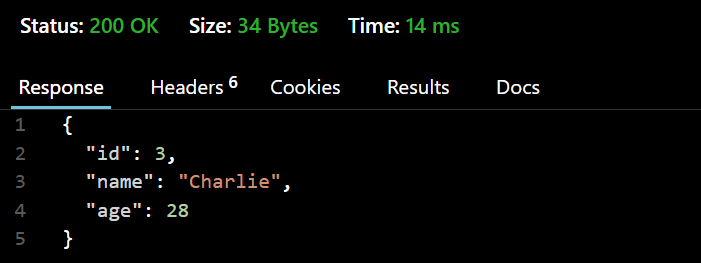
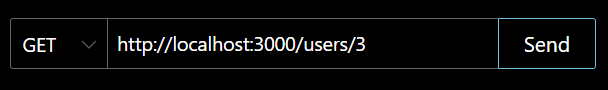
})

**Output:**

Displaying all users



Displaying users based on specific id



Q2] Update the data of the user using patch method

**Code:**

const express = require('express');

const app = express();

const fs = require('fs');

const port = 3000;

app.use(express.json());

let users = JSON.parse(fs.readFileSync('users.json', 'utf-8'));

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

res.json(users);

});

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

const userId = parseInt(req.params.id);

const user = users.find(u => u.id === userId);

if (user) {

res.json(user);

} else {

res.status(404).json({ message: "User not found" });

}

});

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

const userId = parseInt(req.params.id);

const newName = req.query.name || req.body.name;

if (!newName) {

return res.status(400).json({ message: "Name is required" });

}

let user = users.find(u => u.id === userId);

if (!user) {

return res.status(404).json({ message: `User with id ${userId} not found` });

}

user.name = newName;

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

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

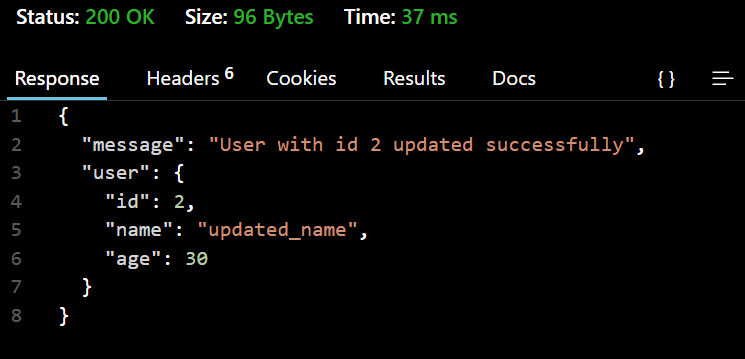
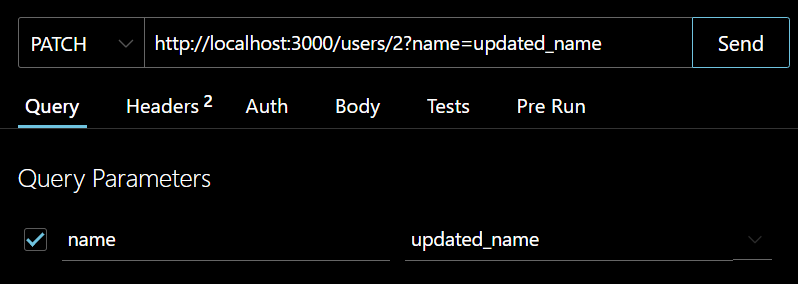
});

app.listen(port, () => {

console.log(`App is listening on port ${port}`);

});

**Output:**



Q3] A. Add a new user to json file

**Code**

const express = require("express");

const fs = require("fs");

const app = express();

const PORT = 3000;

app.use(express.json());

const users = JSON.parse(fs.readFileSync("users.json", "utf8"));

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

res.json(users);

});

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

const userId = parseInt(req.params.id);

const user = users.find(u => u.id === userId);

if (user) {

res.json(user);

} else {

res.status(404).json({ message: "User not found" });

}

});

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

const userId = parseInt(req.params.id);

const { name, age } = req.body;

if (!name || !age) {

return res.status(400).json({ message: "Name and age are required" });

}

const userExists = users.find(u => u.id === userId);

if (userExists) {

return res.status(400).json({ message: `User with ID ${userId} already exists` });

}

const newUser = { id: userId, name, age };

users.push(newUser);

fs.writeFileSync("users.json", JSON.stringify(users, null, 2), "utf8");

res.status(201).json({ message: "User added successfully", user: newUser });

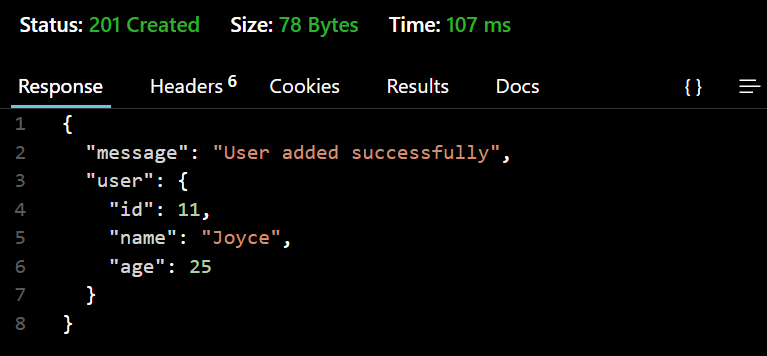
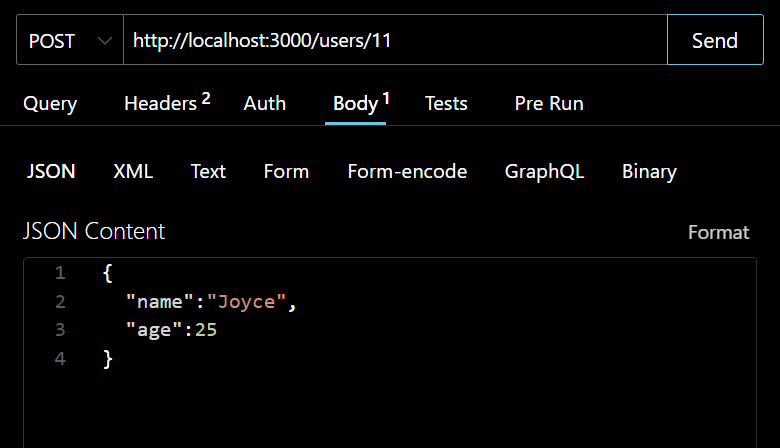
});

app.listen(PORT, () => {

console.log(`Server running on http://localhost:${PORT}`);

});

**Output:**



B. Delete a user with id. Send a response after performing delete operation.

**Code:**

const express = require("express");

const fs = require("fs");

const app = express();

const PORT = 3000;

const users = JSON.parse(fs.readFileSync("users.json", "utf8"));

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

res.json(users);

});

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

const userId = parseInt(req.params.id);

const user = users.find(u => u.id === userId);

if (user) {

res.json(user);

} else {

res.status(404).json({ message: "User not found" });

}

});

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

const id=parseInt(req.params.id);

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

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

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

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

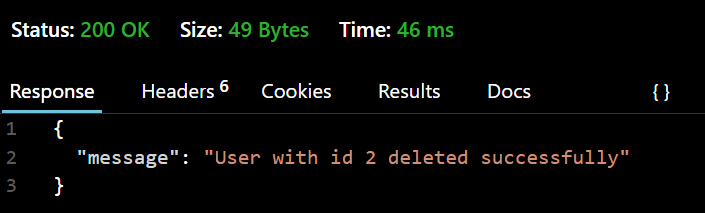
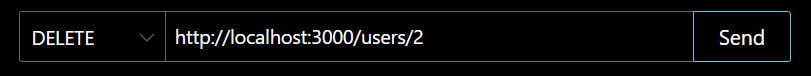
});

app.listen(PORT, () => {

console.log(`Server running on http://localhost:${PORT}`);

});

**Output:**



Q4. Connect Node.js with mongodb

1. Create database

**Code:**

const mongoose = require("mongoose");

async function connectDB() {

try {

await mongoose.connect("mongodb://127.0.0.1:27017/MyDB", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log("Connected to MongoDB Compass successfully!");

} catch (err) {

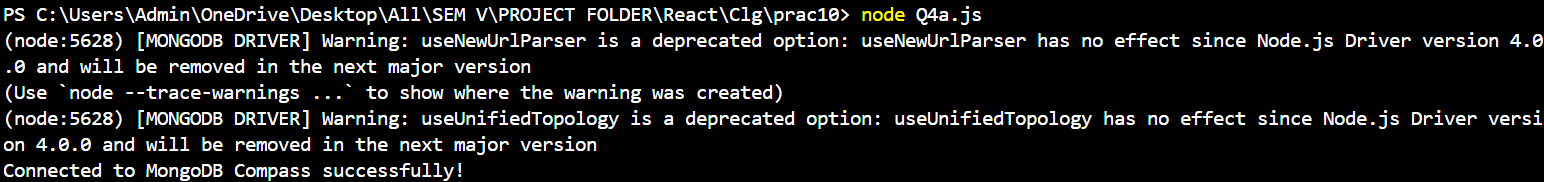
console.error("Error connecting to MongoDB Compass:", err);

}

}

connectDB();

**Output:**



1. Create collection

**Code:**

const mongoose = require("mongoose");

async function main() {

try {

await mongoose.connect("mongodb://127.0.0.1:27017/MyDB", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log("Connected to MongoDB Compass successfully!");

const userSchema = new mongoose.Schema({

id:Number,

name: String,

age: Number,

});

const User = mongoose.model("User", userSchema);

console.log("Collection 'Users' created successfully!");

mongoose.connection.close();

} catch (err) {

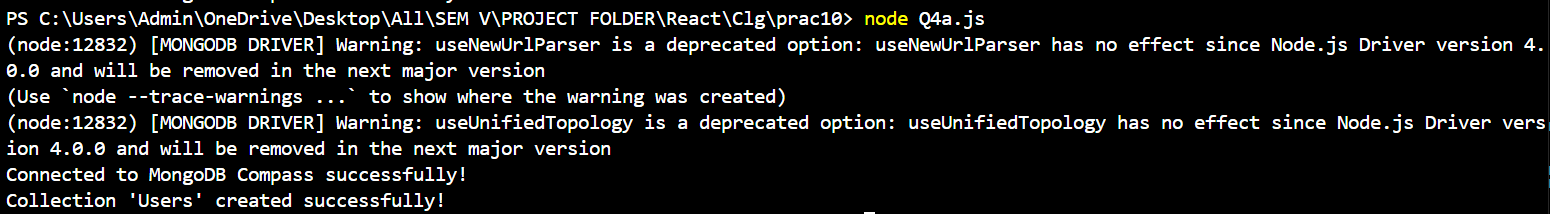
console.error("Error:", err);

}

}

main();

**Output:**



1. Add new users to the collection

**Code:**

MySchema.js

const mongoose = require("mongoose");

const userSchema = new mongoose.Schema({

id: { type: Number, unique: true },

name: String,

age: Number,

});

const User = mongoose.model("User", userSchema);

module.exports = User;

Q4a.js

const mongoose = require("mongoose");

const readline = require("readline");

const User = require("./MySchema");

async function main() {

try {

await mongoose.connect("mongodb://127.0.0.1:27017/MyDB", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log("Connected to MongoDB Compass successfully!");

console.log("Collection 'Users' created successfully!");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

rl.question("Enter User ID: ", async (id) => {

id = parseInt(id);

const existingUser = await User.findOne({ id });

if (existingUser) {

console.error("Error: ID already exists. Choose a different ID.");

rl.close();

mongoose.connection.close();

return;

}

rl.question("Enter Name: ", (name) => {

rl.question("Enter Age: ", async (age) => {

age = parseInt(age);

const newUser = new User({ id, name, age });

await newUser.save();

console.log("User added successfully!");

rl.close();

mongoose.connection.close();

});

});

});

} catch (err) {

console.error("Error:", err);

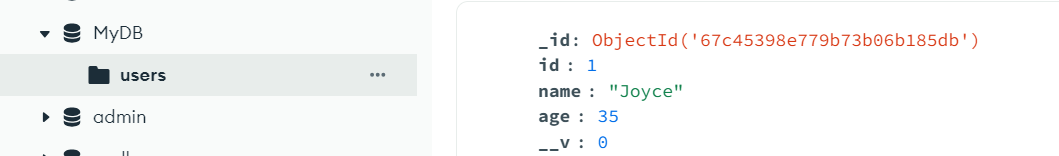
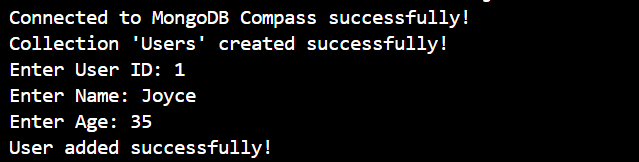
mongoose.connection.close();

}

}

main();

**Output:**



1. Delete user from the collection using specific id

**Code:**

const mongoose = require("mongoose");

const readline = require("readline");

const User = require("./MySchema");

async function main() {

try {

await mongoose.connect("mongodb://127.0.0.1:27017/MyDB", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log("Connected to MongoDB Compass successfully!");

console.log("Collection 'Users' is ready!");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

rl.question("Enter User ID to delete: ", async (id) => {

id = parseInt(id);

const existingUser = await User.findOne({ id });

if (!existingUser) {

console.error("Error: User ID not found.");

} else {

await User.deleteOne({ id });

console.log(`User with ID ${id} deleted successfully!`);

}

rl.close();

mongoose.connection.close();

});

} catch (err) {

console.error("Error:", err);

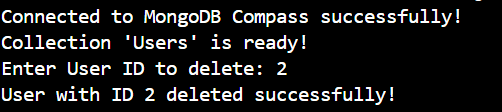
mongoose.connection.close();

}

}

main();

**Output:**



1. Update the data of user for specific id

**Code:**

const mongoose = require("mongoose");

const readline = require("readline");

const User = require("./MySchema");

async function main() {

try {

await mongoose.connect("mongodb://127.0.0.1:27017/MyDB", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log("Connected to MongoDB Compass successfully!");

console.log("Collection 'Users' is ready!");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

rl.question("Enter User ID to update: ", async (id) => {

id = parseInt(id);

const existingUser = await User.findOne({ id });

if (!existingUser) {

console.error("Error: User ID not found.");

rl.close();

mongoose.connection.close();

return;

}

rl.question("Enter new Name: ", (name) => {

rl.question("Enter new Age: ", async (age) => {

age = parseInt(age);

await User.updateOne({ id }, { $set: { name, age } });

console.log(`User with ID ${id} updated successfully!`);

rl.close();

mongoose.connection.close();

});

});

});

} catch (err) {

console.error("Error:", err);

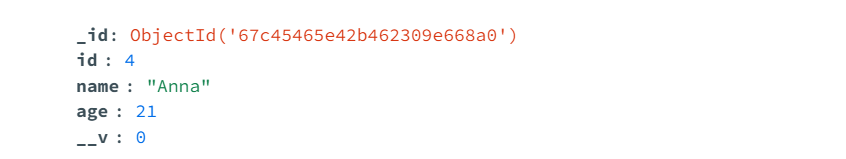
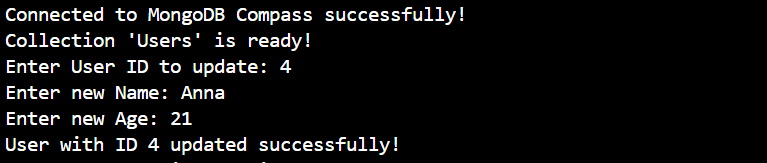
mongoose.connection.close();

}

}

main();

**Output:**



1. Display data of all users

**Code**

const mongoose = require("mongoose");

const User = require("./MySchema");

async function main() {

try {

await mongoose.connect("mongodb://127.0.0.1:27017/MyDB", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log("Connected to MongoDB Compass successfully!");

console.log("Fetching all users...");

const users = await User.find();

if (users.length === 0) {

console.log("No users found in the database.");

} else {

console.log("Users in the database:");

console.table(users.map(user => ({ ID: user.id, Name: user.name, Age: user.age })));

}

mongoose.connection.close();

} catch (err) {

console.error("Error:", err);

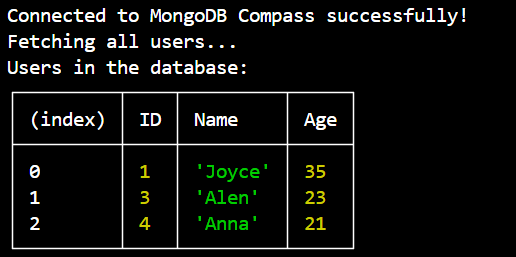
mongoose.connection.close();

}

}

main();

**Output:**



1. Display data of user based on id parameter

**Code:**

const mongoose = require("mongoose");

const readline = require("readline");

const User = require("./MySchema");

async function main() {

try {

await mongoose.connect("mongodb://127.0.0.1:27017/MyDB", {

useNewUrlParser: true,

useUnifiedTopology: true,

});

console.log("Connected to MongoDB Compass successfully!");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

rl.question("Enter User ID to fetch details: ", async (id) => {

id = parseInt(id);

const user = await User.findOne({ id });

if (!user) {

console.error("Error: User ID not found.");

} else {

console.log("User Details:");

console.table({ ID: user.id, Name: user.name, Age: user.age });

}

rl.close();

mongoose.connection.close();

});

} catch (err) {

console.error("Error:", err);

mongoose.connection.close();

}

}

main();

**Output:**

