BACKEND ASSIGNMENT

1. Insert a new document into a collection named "employees" with fields like "name," "position," "salary," and "hire Date."

6. . Build a function to retrieve and display all documents from the "employees" collection.

11. Write a script to update the salary of an employee in the "employees" collection based on their position.

16. Build a script to delete a specific employee document from the "employees" collection based on their name

Db.js:

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

const client=new MongoClient('mongodb://127.0.0.1:27017')

async function insert(d){

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('employeess');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return('record already exist');

    } else{

        const res=await collection.insertOne(d);

        return(' new record created');

    }

}

async function read(d){

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('employeess');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return(result);

    } else{

        return('document not found');

    }

}

async function update(d){ // {'name':'sushma','salary':'20lpa'}

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('employeess');

    const result=await collection.find({'name':d.name}).toArray();

    console.log(result);

    if(result.length>0){

        const res=await collection.updateOne({'name':d.name},{$set:{'salary':d.salary}});

        console.log(res);

        return('salary updated');

    } else{

        return('no user found');

    }

}

async function remove(d){ // {'email':'madhu'}

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('employeess');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        const res=await collection.deleteOne(d);

        console.log(res);

        return('document deleted');

    } else{

        return('no user found');

    }

}

module.exports={

    insert,

    read,

    update,

    remove

}

App.js:

const express=require('express');

const url=require('url')

const {insert,read,update,remove}=require('./db1-6-11-16');

var api=express();

api.get('/',function(){

    console.log('API Server Started');

});

api.get('/insert',async function(req,res){

    var urldata=url.parse(req.url,true);

    var name=urldata.query.name;

    var position=urldata.query.position;

    var salary=urldata.query.salary;

    var hiredate=urldata.query.hiredate;

    mquery={'name':name,'position':position,'salary':salary,'hiredate':hiredate};

    const result=await insert(mquery);

    res.send(result);

});

api.get('/read',async function(req,res){

    var urldata=url.parse(req.url,true);

    var name=urldata.query.name;

    mquery={'name':name};

    const result=await read(mquery);

    res.send(result);

});

api.get('/update',async function(req,res){

    var urldata=url.parse(req.url,true);

    var name=urldata.query.name;

    var newsalary=urldata.query.newsalarysalary;

    mquery={'name':name,'newsalary':newsalary};

    const result=await update(mquery);

    res.send(result);

});

api.get('/remove',async function(req,res){

    var urldata=url.parse(req.url,true);

    var name=urldata.query.name;

    mquery={'name':name};

    const result=await remove(mquery);

    res.send(result);

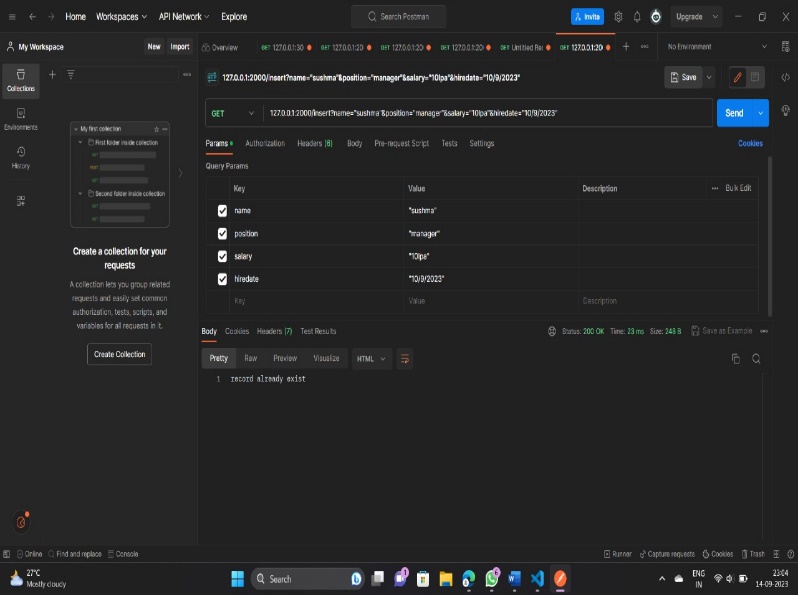
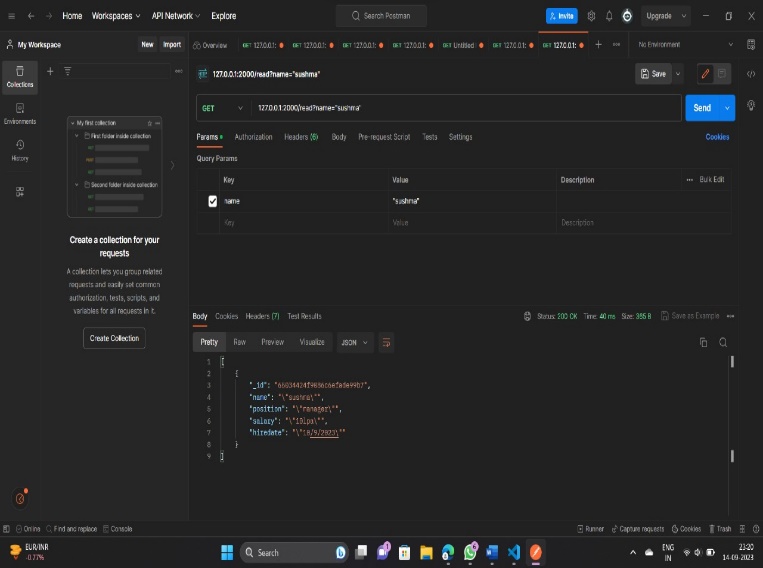
});

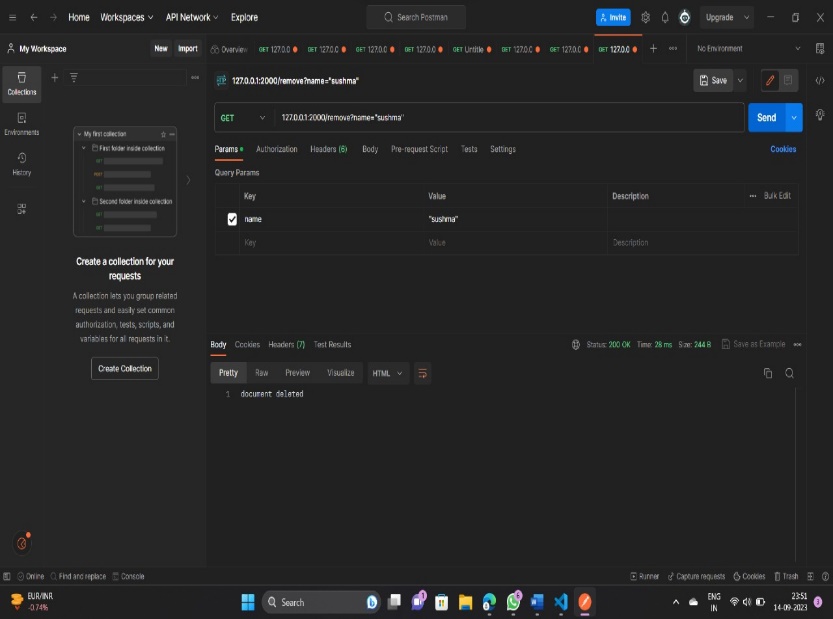
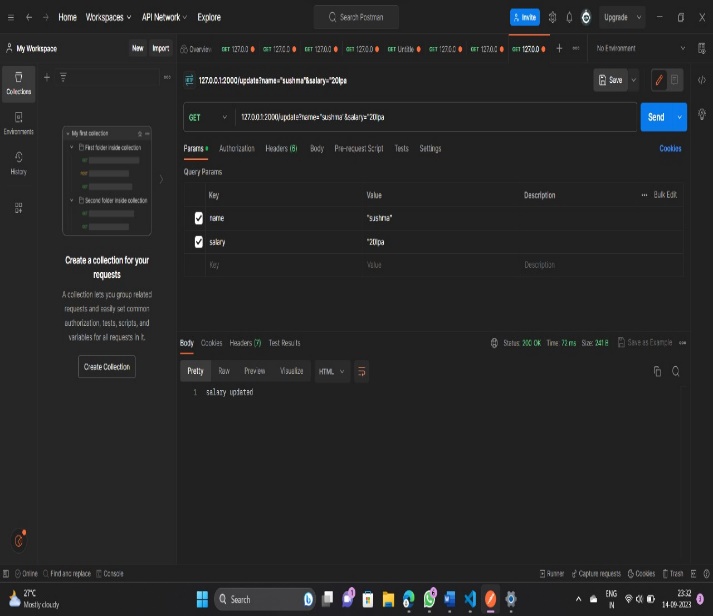
api.listen(2000,function(){

    console.log('API Server Started')

});

OUTPUT:



2. Create a collection named "books" and add at least 5 book documents with fields such as "title," "author," "genre," and "publication Year."

7. Implement a query to find and display all books in the "books" collection that belong to a specific genre.

12. Implement a method to change the "author" field of a book document in the "books" collection.

17. Implement an API route to remove all books from the "books" collection published before a certain year.

App.js

const express=require('express');

const url=require('url')

const {insert,read,update,remove}=require('./db2-5-12-17');

var api=express();

api.get('/',function(){

    console.log('API Server Started');

});

api.get('/insert',async function(req,res){

    var urldata=url.parse(req.url,true);

    var name=urldata.query.name;

    var author=urldata.query.author;

    var genre=urldata.query.genre;

    var year=urldata.query.year;

    mquery={'name':name,'author':author,'genre':genre,'year':year};

    const result=await insert(mquery);

    res.send(result);

});

api.get('/read',async function(req,res){

    var urldata=url.parse(req.url,true);

    var genre=urldata.query.genre;

    mquery={'genre':genre};

    const result=await read(mquery);

    res.send(result);

});

api.get('/update',async function(req,res){

    var urldata=url.parse(req.url,true);

    var name=urldata.query.name;

    var newauthor=urldata.query.newauthor;

    mquery={'name':name,'newauthor':newauthor};

    const result=await update(mquery);

    res.send(result);

});

api.get('/remove',async function(req,res){

    var urldata=url.parse(req.url,true);

    var name=urldata.query.name;

    mquery={'year':year};

    const result=await remove(mquery);

    res.send(result);

});

api.listen(2000,function(){

    console.log('API Server Started')

});

Db.js:

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

const client=new MongoClient('mongodb://127.0.0.1:27017')

async function insert(d){

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('books');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return('record already exist');

    } else{

        const res=await collection.insertOne(d);

        return(' new record created');

    }

}

async function read(d){

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('books');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return(result);

    } else{

        return('document not found');

    }

}

async function update(d){ // {'name':'a3','author':'b6'}

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('books');

    const result=await collection.find({'name':d.name}).toArray();

    console.log(result);

    if(result.length>0){

        const res=await collection.updateOne({'name':d.name},{$set:{'author':d.author}});

        console.log(res);

        return('author updated');

    } else{

        return('no user found');

    }

}

async function remove(d){ // {'email':'madhu'}

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('books');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        if(year<newyear)

        {

        const res=await collection.deleteOne(d);

        console.log(res);

        return('document deleted');

        }

    } else{

        return('no user found');

    }

}

module.exports={

    insert,

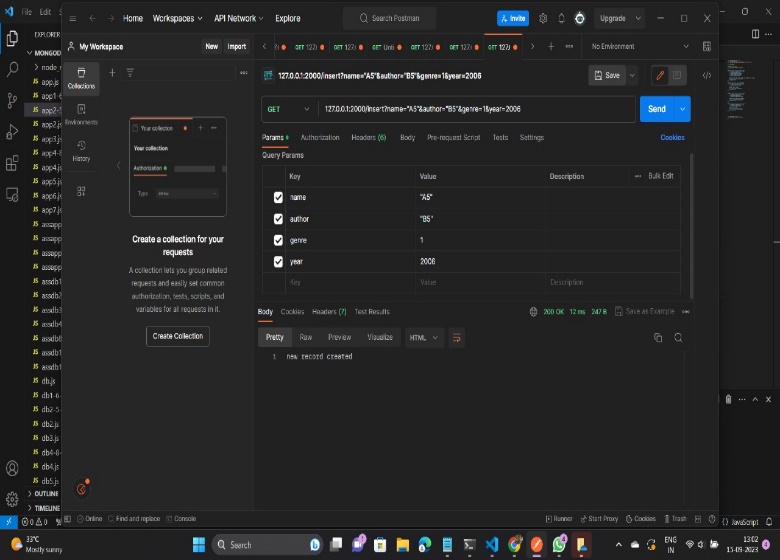
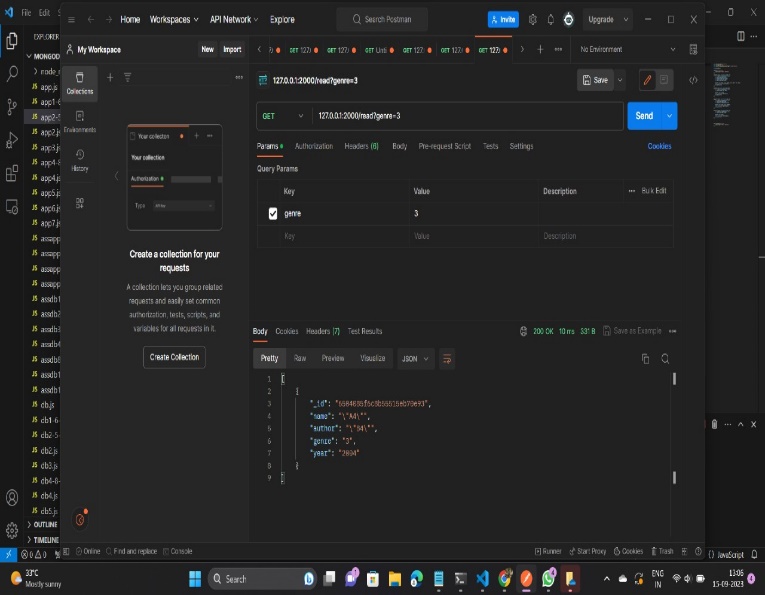
    read,

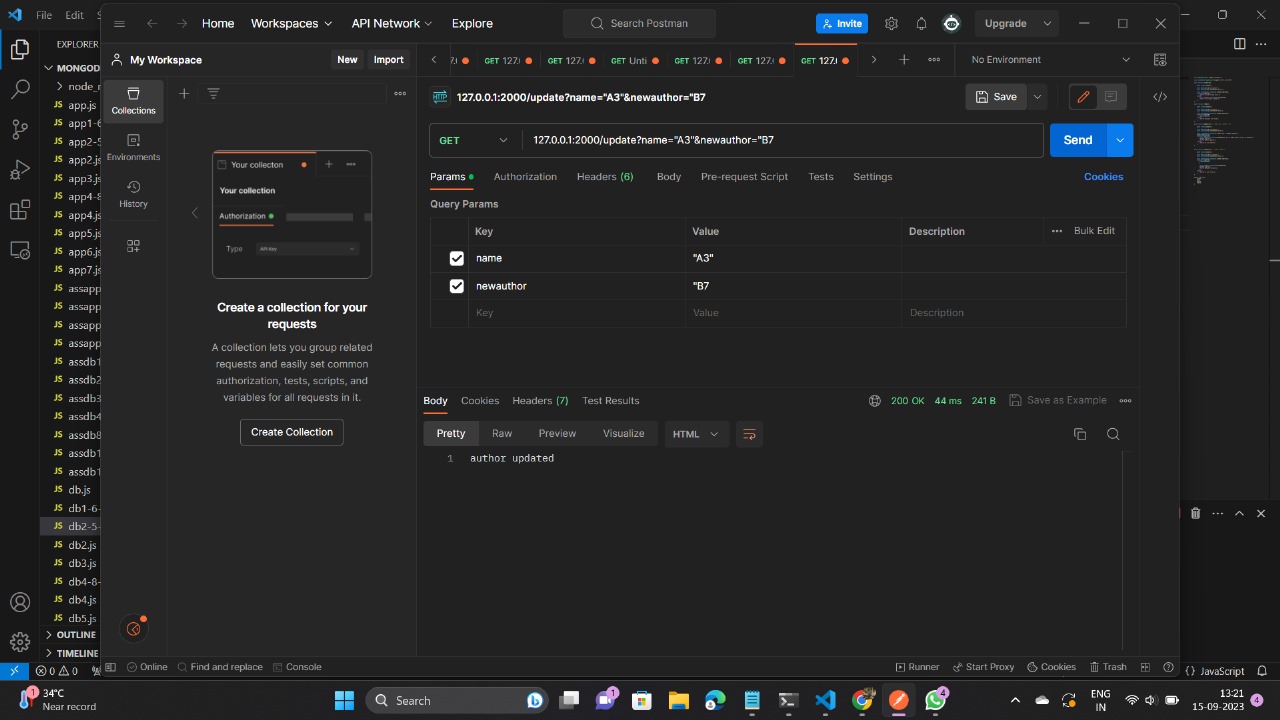
    update,

    remove

}

OUTPUT:



3. Develop a Node.js script to insert multiple student documents at once into a "students" collection.

10. Design a function to find and display the oldest student from the "students" collection

App.js:

const express=require('express');

const url=require('url')

const {insert,read}=require('./dbstudents');

var api=express();

api.get('/',function(){

    console.log('API Server Started');

});

api.get('/insert',async function(req,res){

    var urldata=url.parse(req.url,true);

    var regdno=urldata.query.regdno;

    var branch=urldata.query.branch;

    var age=urldata.query.age;

    mquery={'regdno':regdno,'branch':branch,'age':age};

    const result=await insert(mquery);

    res.send(result);

});

api.get('/read',async function(req,res){

    var urldata=url.parse(req.url,true);

    var age=urldata.query.age;

    mquery={'age':age};

    const result=await read(mquery);

    res.send(result);

});

api.listen(2004,function(){

    console.log('API Server Started')

});

Db.js:

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

const client=new MongoClient('mongodb://127.0.0.1:27017')

async function insert(d){

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('students');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return('student already exist');

    } else{

        const res=await collection.insertOne(d);

        return('new student created');

    }

}

async function read(d){

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('students');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return(result);

    } else{

        return('no student found');

    }

}

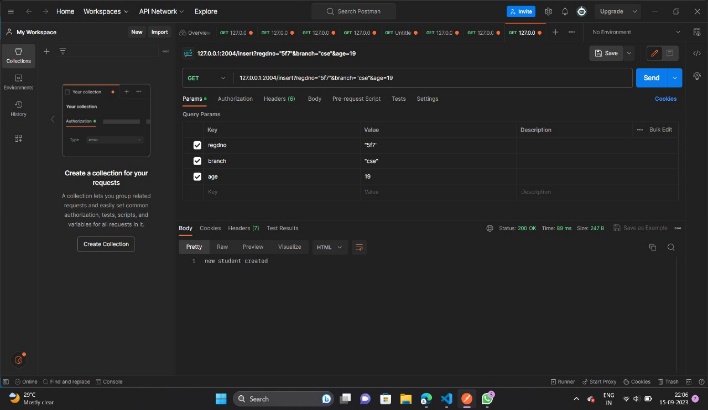
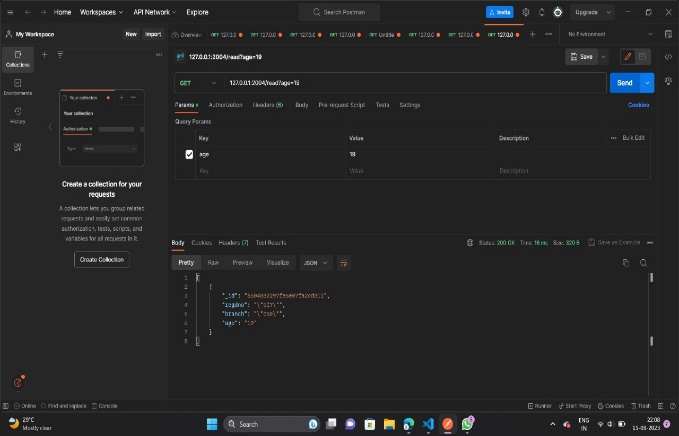
module.exports={

    insert,

    read

}

OUTPUT:

4. Implement a way to store and manage user accounts in a collection named "users" with fields like "username," "email," and "password."

8. Develop a Node.js API to retrieve a user's details from the "users" collection based on their email.

13. Develop an API endpoint to update a user's password in the "users" collection using their username.

18. Develop a function to delete a user from the "users" collection using their email

Db.js:

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

const client=new MongoClient('mongodb://127.0.0.1:27017')

async function insert(d){

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('users');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return('user already exist');

    } else{

        const res=await collection.insertOne(d);

        return('new user created');

    }

}

async function read(d){

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('users');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return(result);

    } else{

        return('no user found');

    }

}

async function update(d){ // {'username':'madhu','newpassword':'1234'}

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('users');

    const result=await collection.find({'username':d.username}).toArray();

    console.log(result);

    if(result.length>0){

        const res=await collection.updateOne({'username':d.username},{$set:{'password':d.newpassword}});

        console.log(res);

        return('password updated');

    } else{

        return('no user found');

    }

}

async function remove(d){ // {'email':'madhu'}

    await client.connect();

    const db=client.db('assignment');

    const collection=db.collection('users');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        const res=await collection.deleteOne(d);

        console.log(res);

        return('user deleted');

    } else{

        return('no user found');

    }

}

module.exports={

    insert,

    read,

    update,

    remove

}

App.js:

const express=require('express');

const url=require('url')

const {insert,read,update,remove}=require('./db4-8-13-18');

var api=express();

api.get('/',function(){

    console.log('API Server Started');

});

api.get('/insert',async function(req,res){

    var urldata=url.parse(req.url,true);

    var username=urldata.query.username;

    var password=urldata.query.password;

    var email=urldata.query.email;

    mquery={'username':username,'password':password,'email':email};

    const result=await insert(mquery);

    res.send(result);

});

api.get('/read',async function(req,res){

    var urldata=url.parse(req.url,true);

    var email=urldata.query.email;

    mquery={'email':email};

    const result=await read(mquery);

    res.send(result);

});

api.get('/update',async function(req,res){

    var urldata=url.parse(req.url,true);

    var username=urldata.query.username;

    var newpassword=urldata.query.newpassword;

    mquery={'username':username,'newpassword':newpassword};

    const result=await update(mquery);

    res.send(result);

});

api.get('/remove',async function(req,res){

    var urldata=url.parse(req.url,true);

    var email=urldata.query.email;

    mquery={'email':email};

    const result=await remove(mquery);

    res.send(result);

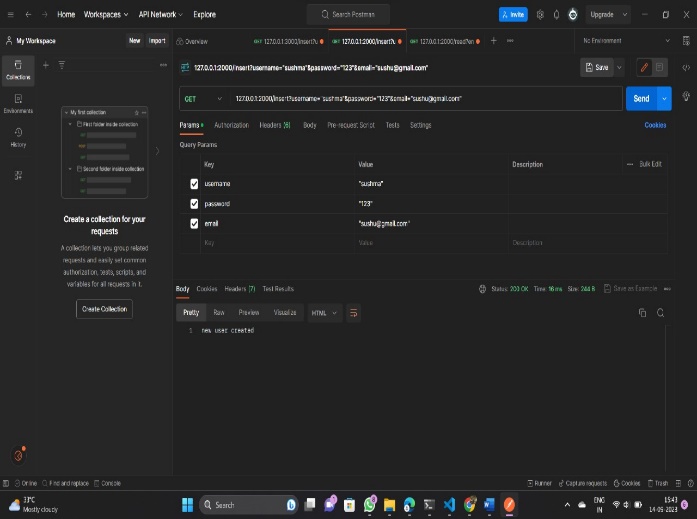
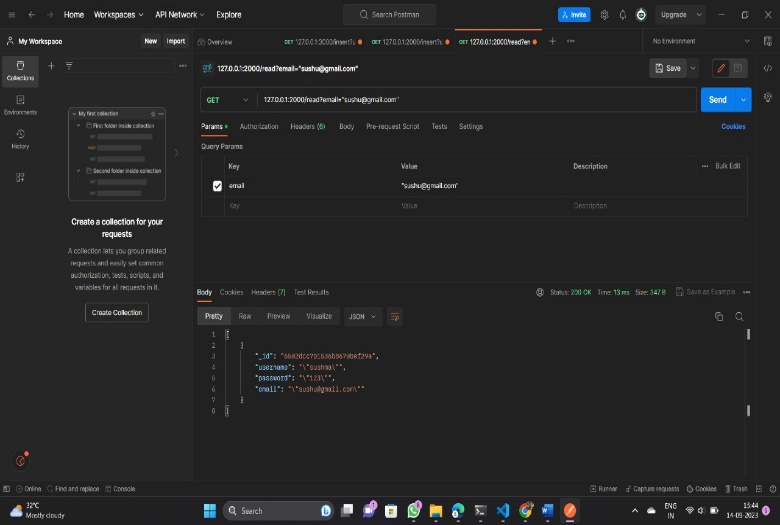
});

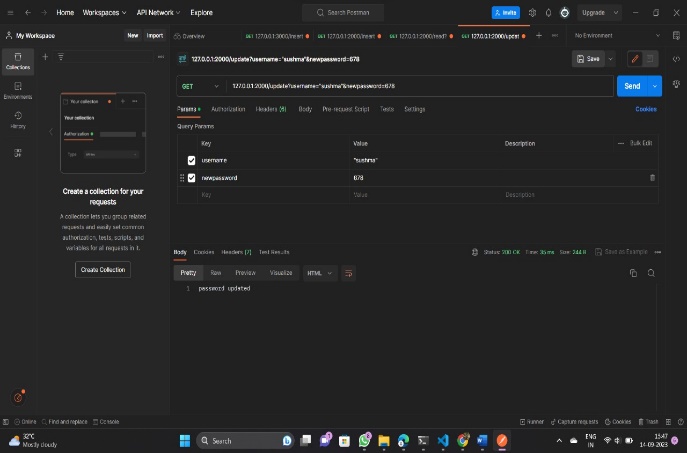
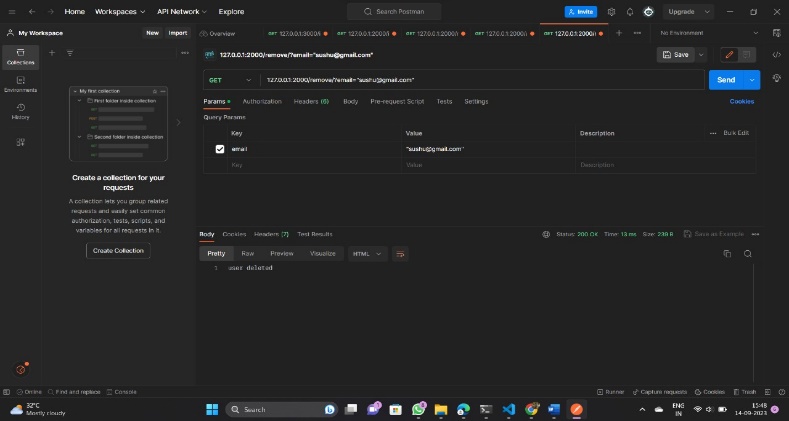
api.listen(2000,function(){

    console.log('API Server Started')

});

OUTPUT:

5. Create a collection "orders" for an e-commerce application, adding documents for orders with fields like "orderNumber," "products," "totalAmount," and "customerName."

9. Create a route to fetch orders from the "orders" collection that have a total amount greater than a specified value.

15. Create a route to update the status of an order in the "orders" collection (e.g., from "pending" to "shipped").

20. Create an API endpoint to cancel an order by removing the corresponding document from the "orders" collection based on the order number.

DB.JS:

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

const client=new MongoClient('mongodb://127.0.0.1:27017')

async function insert(d)

{

    await client.connect();

    const db=client.db('kits');

    const collection=db.collection('orders');

    const result=await collection.find(d).toArray();

    if(result.length>0)

    {

        return('order already exist');

    }

    else{

        const res=await collection.insertOne(d);

        return('new order created');

    }

}

async function read(d){

    await client.connect();

    const db=client.db('kits');

    const collection=db.collection('orders');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        return(result);

    } else{

        return('no order found');

    }

}

async function update(d){

    await client.connect();

    const db=client.db('kits');

    const collection=db.collection('orders');

    const result=await collection.find({'ordernum':d.ordernum}).toArray();

    console.log(result);

    if(result.length>0){

        const res=await collection.updateOne({'ordernum':d.ordernum},{$set:{'status':d.newstatus}});

        console.log(res);

        return('status updated');

    } else{

        return('no record found');

    }

}

async function remove(d){

    await client.connect();

    const db=client.db('kits');

    const collection=db.collection('orders');

    const result=await collection.find(d).toArray();

    if(result.length>0){

        const res=await collection.deleteOne(d);

        console.log(res);

        return('order deleted');

    } else{

        return('no order found');

    }

}

module.exports={

    insert,

    read,

    update,

    remove

}

App.js:

const express=require('express');

const url=require('url')

const {insert,read,update,remove}=require('./dborders');

var api=express();

api.get('/',function(){

    console.log('API Server Started');

});

api.get('/insert',async function(req,res){

    var urldata=url.parse(req.url,true);

    var ordernum=urldata.query.ordernum;

    var products=urldata.query.products;

    var totalamt=urldata.query.totalamt;

    var custname=urldata.query.custname;

    var status=urldata.query.status

    mquery={'ordernum':ordernum,'products':products,'totalamt':totalamt,'custname':custname,'status':status};

    const result=await insert(mquery);

    res.send(result);

});

api.get('/read',async function(req,res){

    var urldata=url.parse(req.url,true);

    var totalamt=urldata.query.totalamt;

    mquery={'totalamt':totalamt};

    const result=await read(mquery);

    res.send(result);

});

api.get('/update',async function(req,res){

    var urldata=url.parse(req.url,true);

    var ordernum=urldata.query.ordernum;

    var newstatus=urldata.query.newstatus;

    mquery={'ordernum':ordernum,'newstatus':newstatus};

    const result=await update(mquery);

    res.send(result);

});

api.get('/remove',async function(req,res){

    var urldata=url.parse(req.url,true);

    var ordernum=urldata.query.ordernum;

    mquery={'ordernum':ordernum};

    const result=await remove(mquery);

    res.send(result);

});

api.listen(2007,function(){

    console.log('API Server Started')

})

OUTPUT:

