**Lab 8**

1-) Create a NoSQL design model for an application to manage a library, taking into

consideration the following requirements:

• Books have an ISBN number and are categorized by author and tagged

by keywords to facilitate search

• Books can be borrowed by students, so the librarian will be able to check all

borrowed books and their return date so he may contact students who are late

returning their books.

a) Develop a RESTful application for the following requirements:

a. Students can login to the system.

const express = require('express');

const router = express.Router();

const jwtManager = require('../jwt/JwtManager')

//LOGIN

router.post('/', function (req, res, next) {

    const fname = req.body.fname;

    const email = req.body.email;

    console.log(`  fname  ${fname}   email  ${email}     `)

    // $or: [ { quantity: { $lt: 20 } }, { price: 10 } ] }

    req.db.collection('students').findOne({ $and: [{ email: { $eq: email } }, { Fname: { $eq: fname } }] })

        .then(data => {

            console.log('get', data)

            // console.log("Email", data.email)

            // console.log("data.fname", data.Fname)

            if (data) {

                //if (fname === data.Fname && email === data.email) {

                console.log(`emaaaaaaaaaaaaaaaaaaaaaaaaa`)

                // const data = {};

                // data.id = 1;

                // data.email = 'uinan@miu.edu';

                // data.comment = 'JWT is awesome'

                // const token = jwtManager.generate(data);

                //  res.json({ data: token, status: 'success' });

                res.json({ Status: "logged in " })

            }

            else {

                res.json({ status: 'invalid\_user' });

            }

        })

        .catch(err => {

            console.log(err)

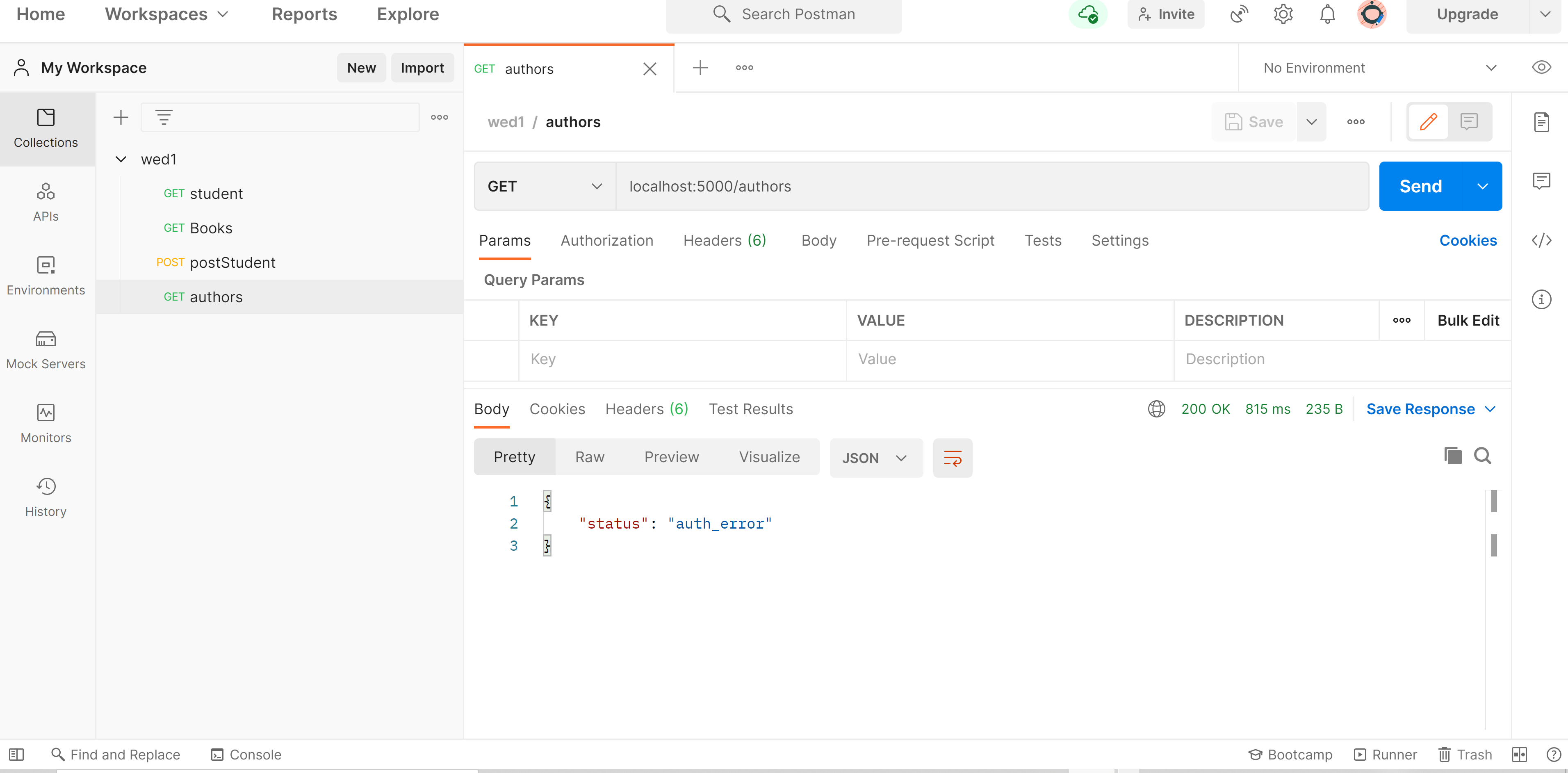
            res.json({ status: "Error" })

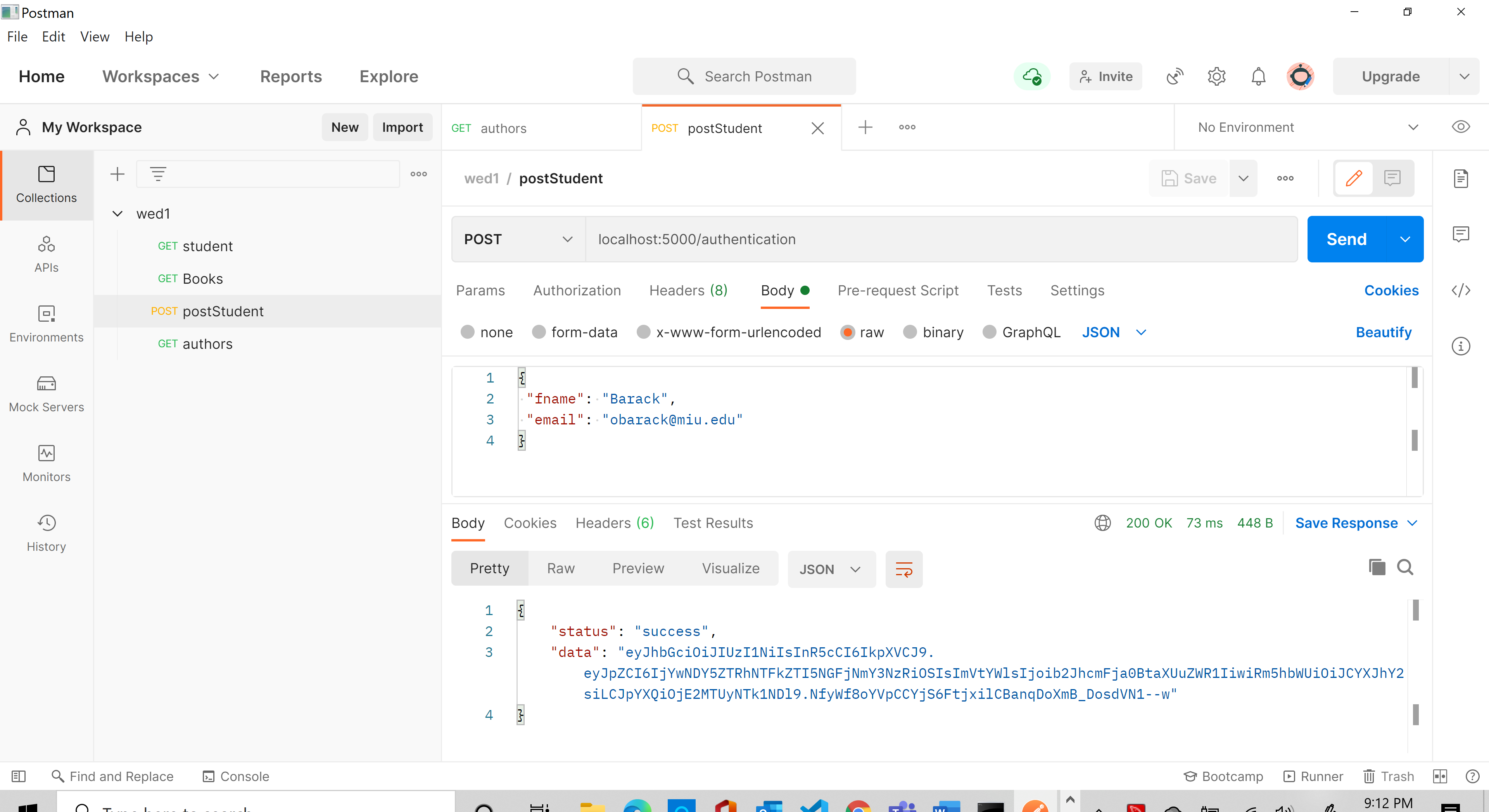
        })

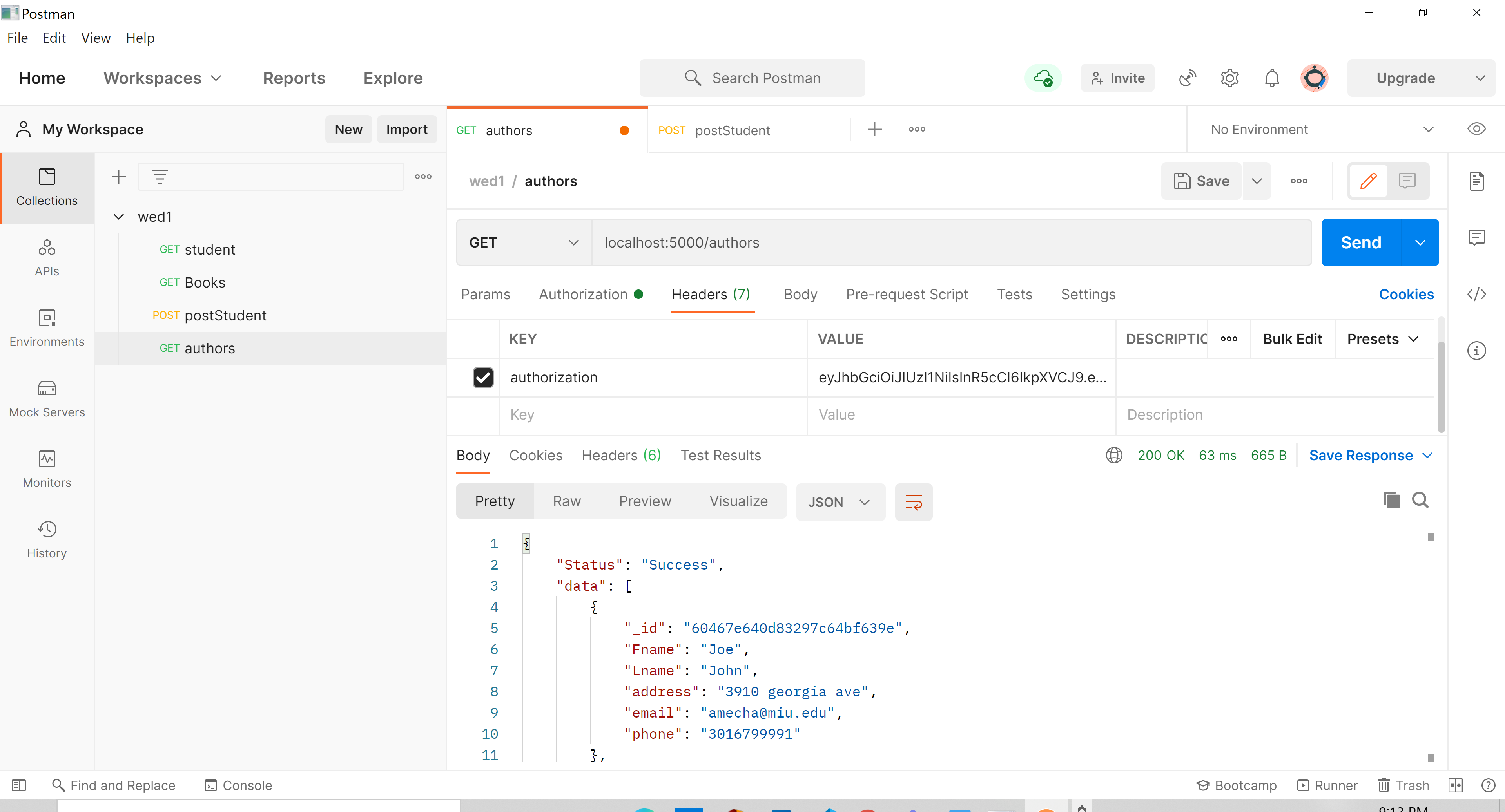
});

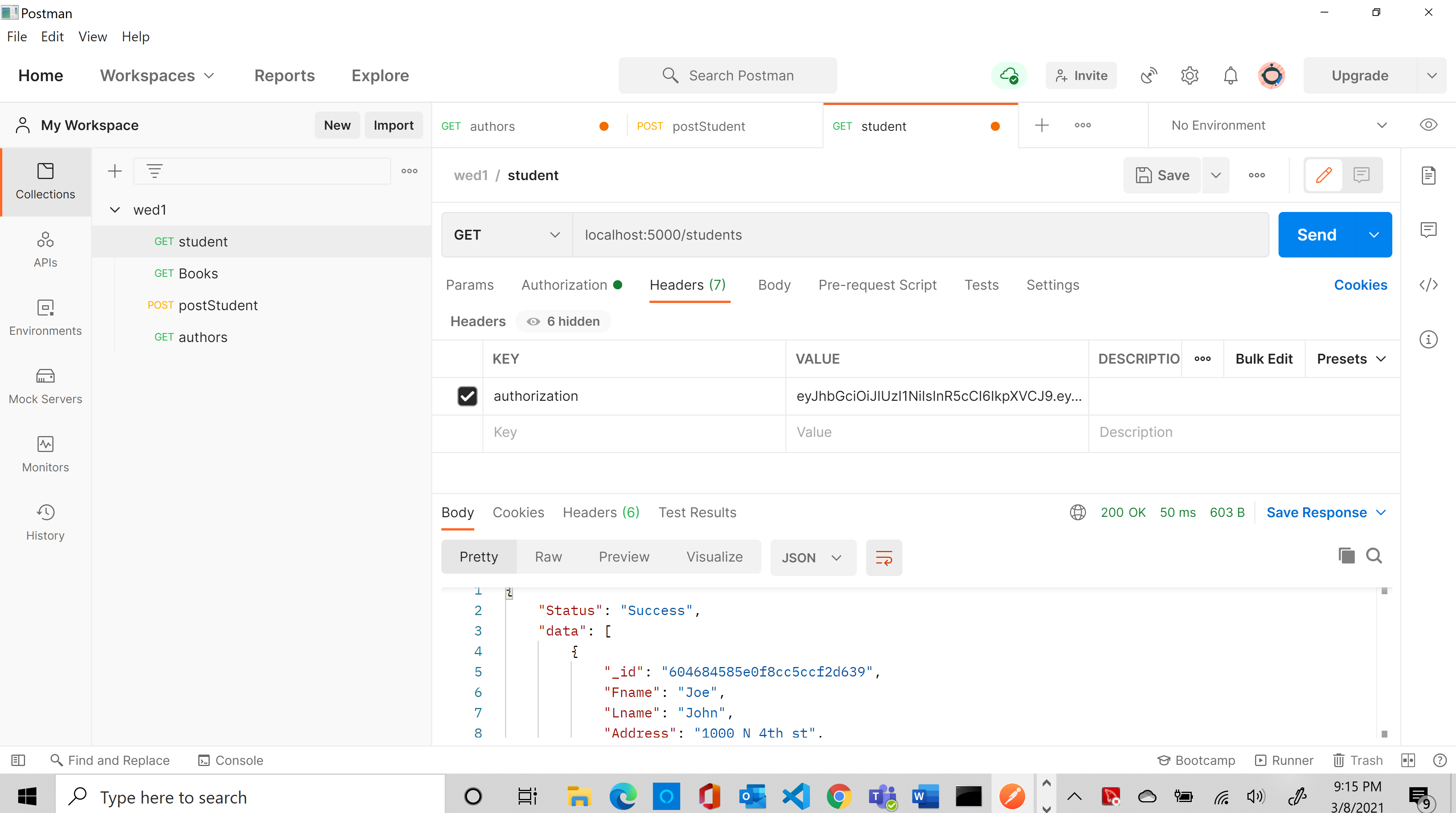
module.exports = router;

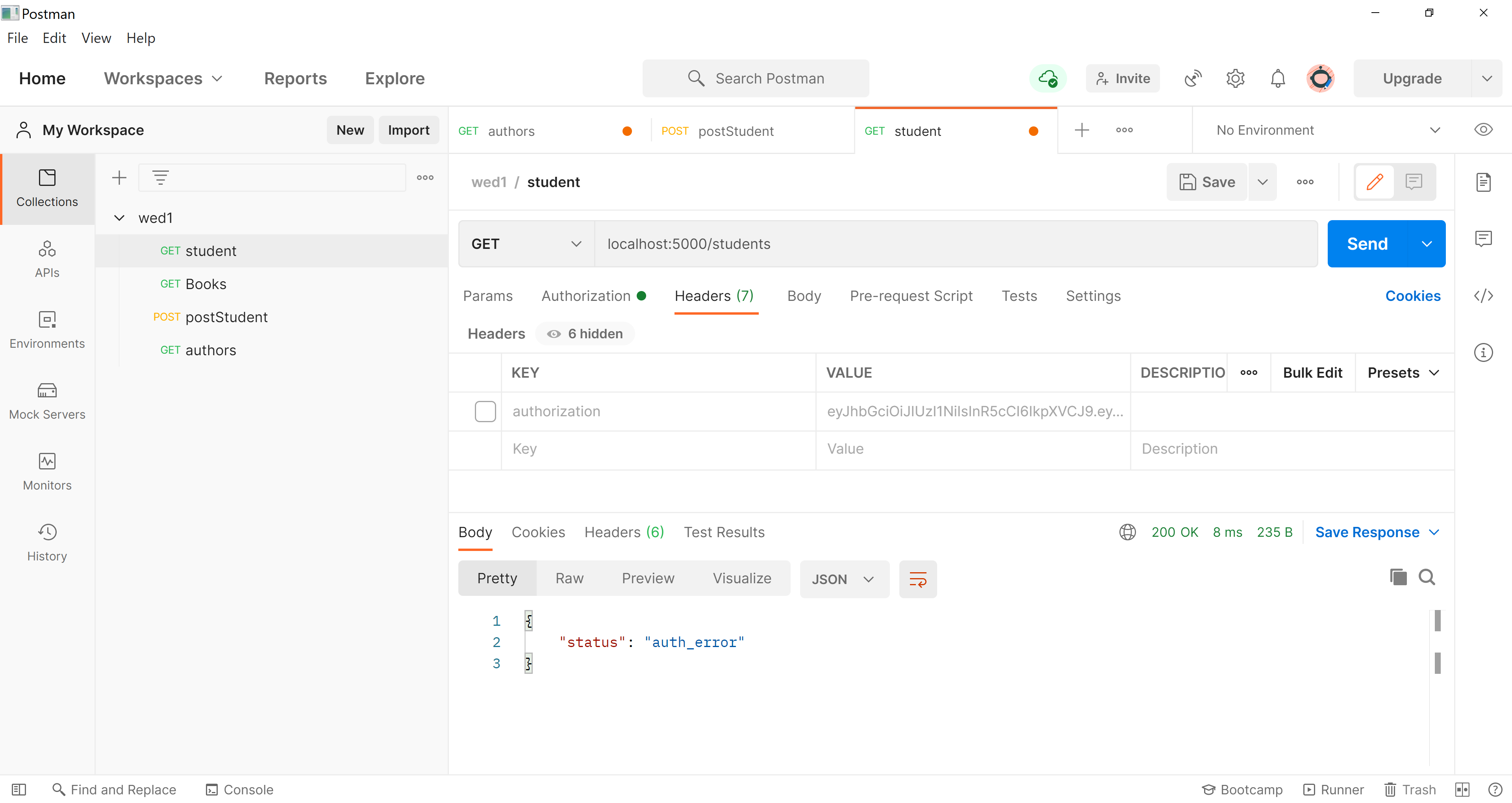
b. Secure your endpoints by using JWT.



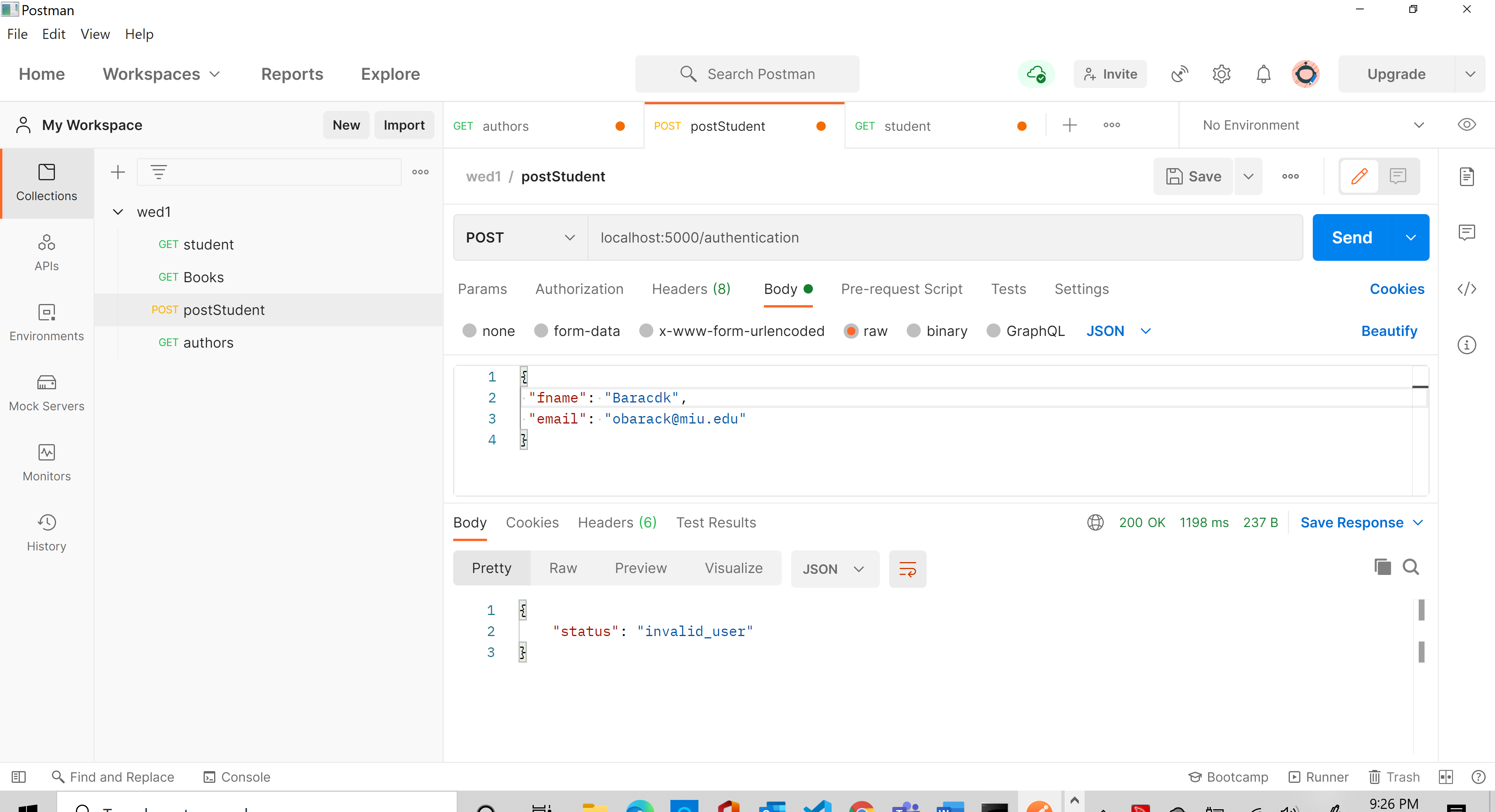








//if we insert invalid user



//if the collection name or db name is wrong

c. Implement CRUD operations for the books.

//display only title having aligorithm

{

    "Status": "Success",

    "data": [

        {

            "\_id": "60467d120d83297c64bf639d",

            "Isbn": "12345",

            "title": "Algorithm",

            "Author": {

                "Fname": "Joe",

                "Lname": "John",

                "\_id": "60467e640d83297c64bf639e"

            },

            "keywork": [

                "sql",

                "nodejs",

                "mongodb"

            ],

            "Borrowers": [

                {

                    "name": "Amanuel",

                    "email": "amecha@miu.edu",

                    "returnDate": "02-09",

                    "\_id": "60469f4551de294ac6f774ba"

                }

            ]

        },

        {

            "\_id": "6046a06c51de294ac6f774bb",

            "Isbn": "78998",

            "title": "Server Side programing",

            "Author": [

                {

                    "Fname": "Umur",

                    "Lname": "Inan",

                    "\_id": "6046a17f51de294ac6f774bc"

                },

                {

                    "Fname": "Asaad",

                    "Lname": "Saad",

                    "\_id": "6046a1eb51de294ac6f774bd"

                }

            ],

            "keywork": [

                "express",

                "nodejs",

                "mongodb"

            ],

            "Borrowers": [

                {

                    "name": "Barack",

                    "email": "obarack@miu.edu",

                    "returnDate": "02-09"

                }

            ]

        }

    ]

}

Only algorithm

{

    "Status": "Success",

    "data": [

        {

            "\_id": "60467d120d83297c64bf639d",

            "Isbn": "12345",

            "title": "Algorithm",

            "Author": {

                "Fname": "Joe",

                "Lname": "John",

                "\_id": "60467e640d83297c64bf639e"

            },

            "keywork": [

                "sql",

                "nodejs",

                "mongodb"

            ],

            "Borrowers": [

                {

                    "name": "Amanuel",

                    "email": "amecha@miu.edu",

                    "returnDate": "02-09",

                    "\_id": "60469f4551de294ac6f774ba"

                }

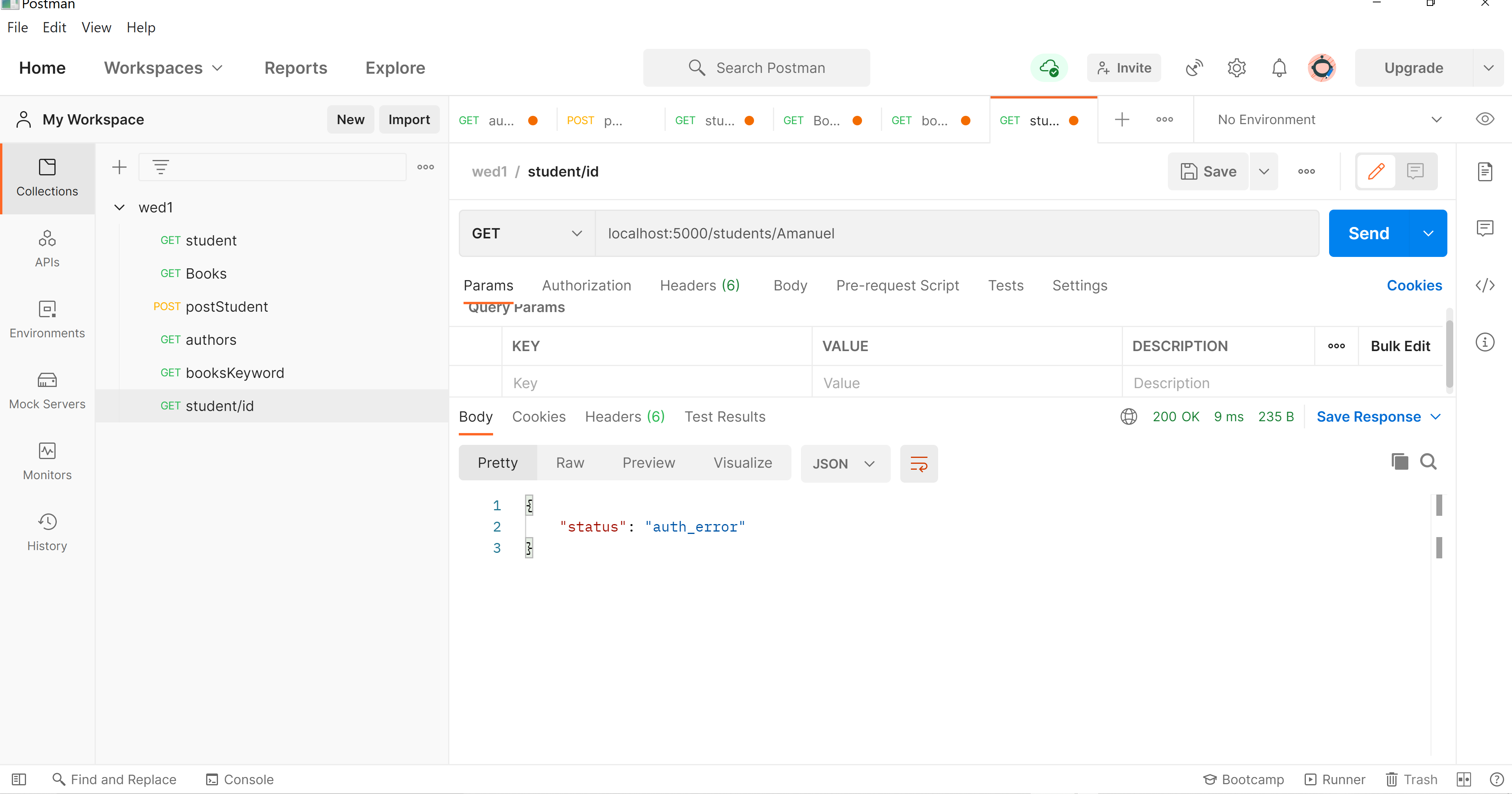
            ]

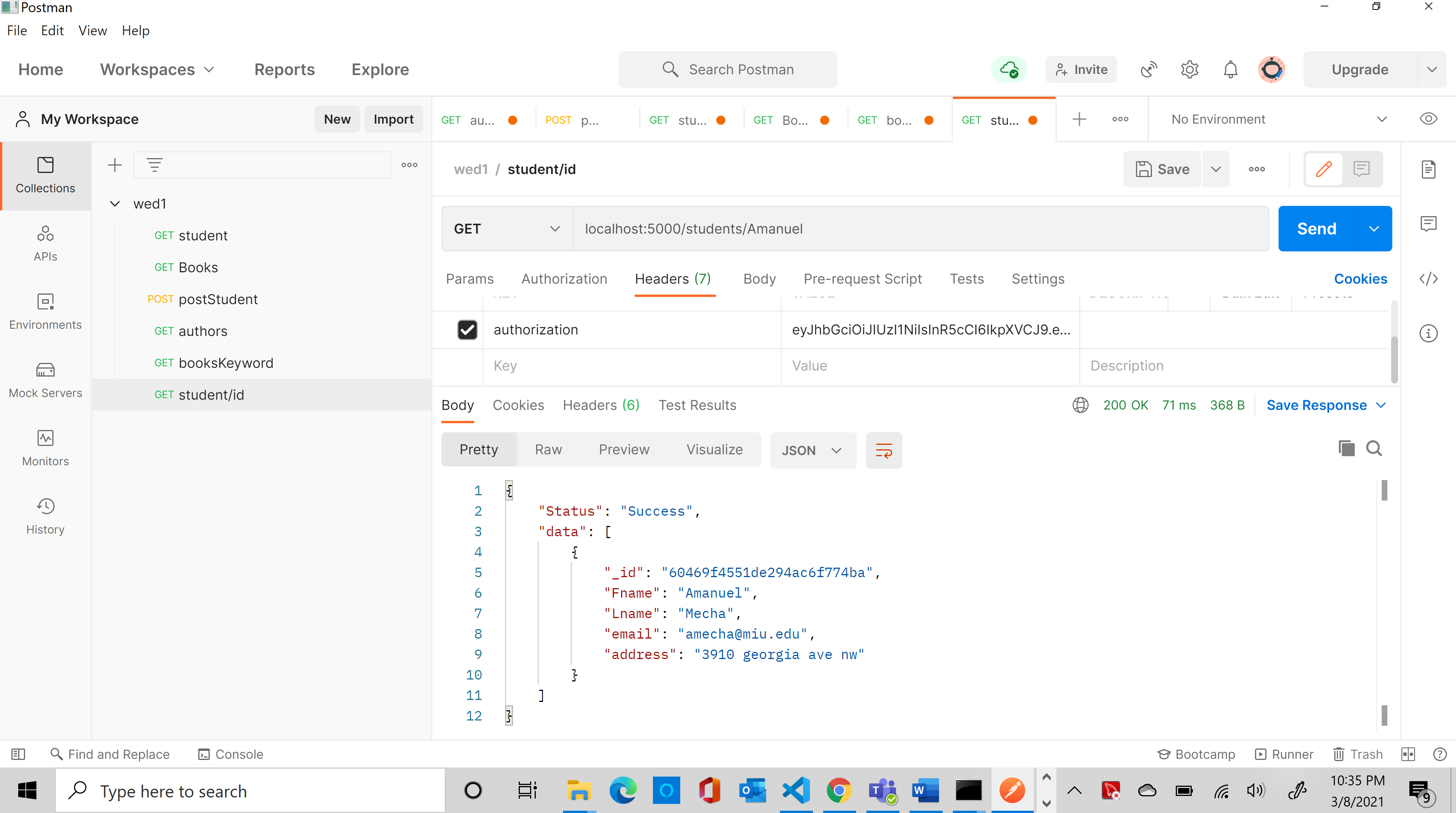
        }

    ]

}

d. Implement CRUD operation for the students.





Add student

router.post('/', function (req, res) {

    //const para = { 'first': req.body.fname, 'last': req.body.lname }

    req.db.collection('students').insertOne(req.body)

        .then(data => {

            res.json({ Status: "Success" })

        })

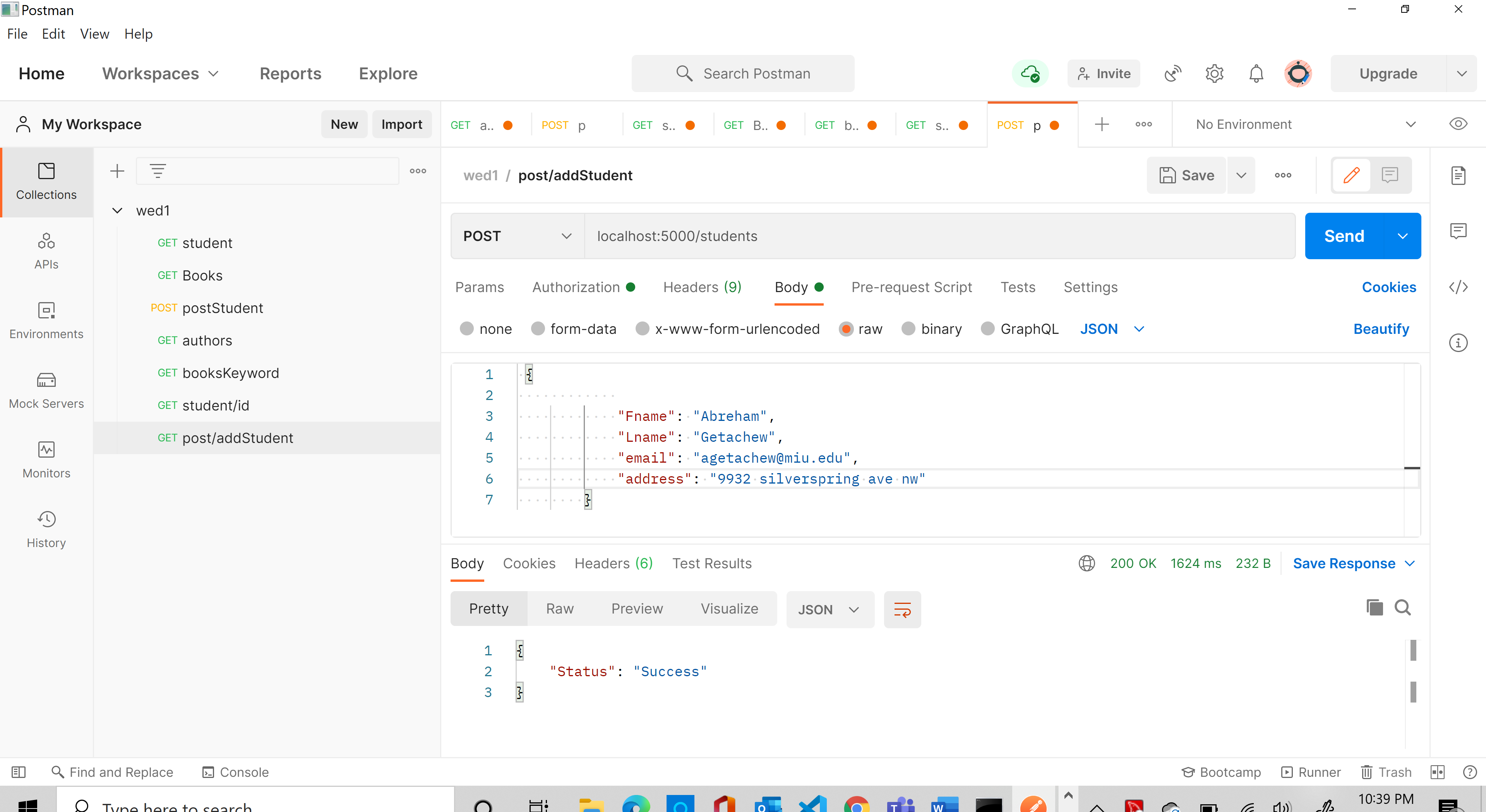
        .catch(err => {

            console.log(err)

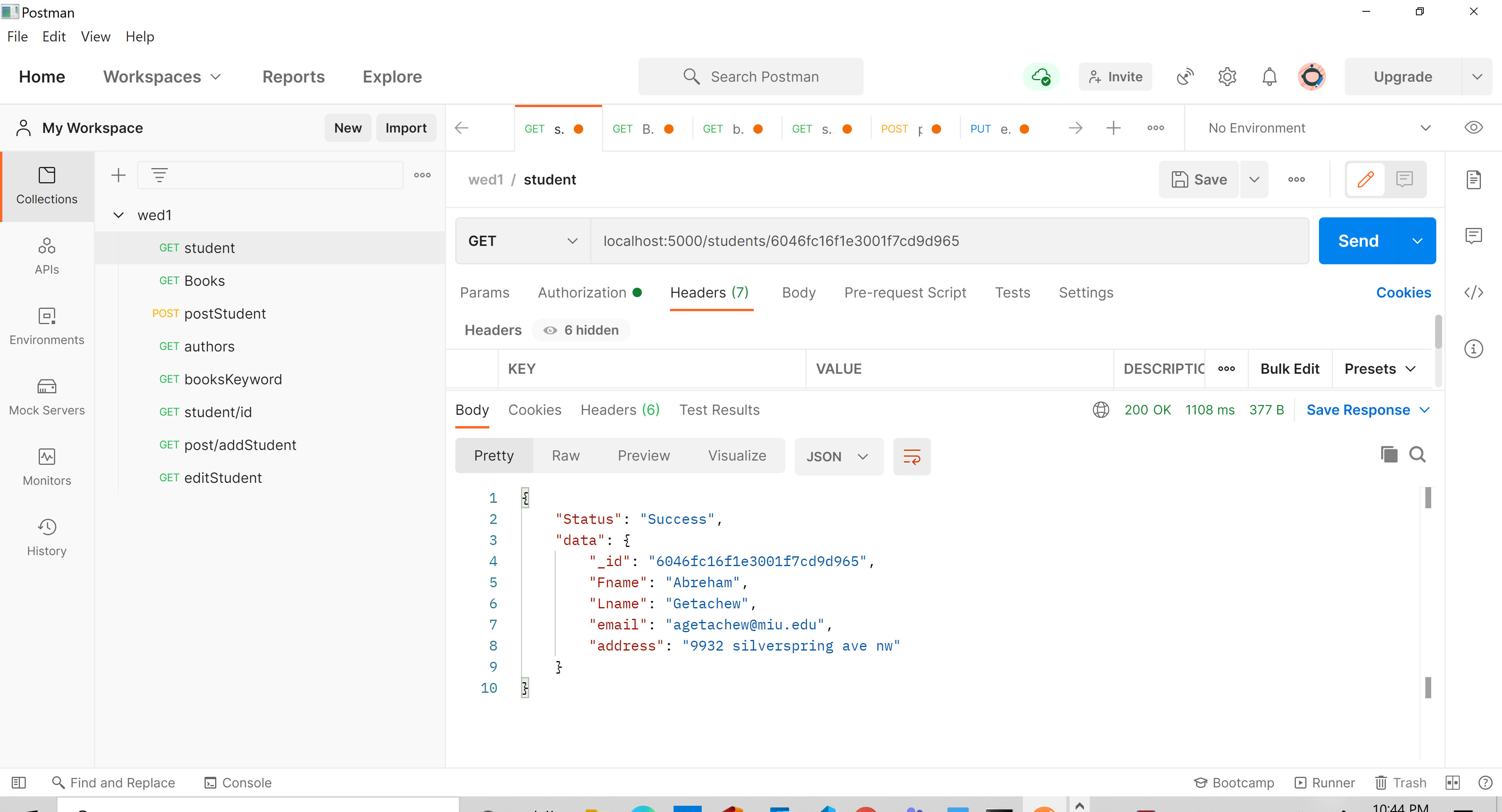
            res.json({ status: "Error" })

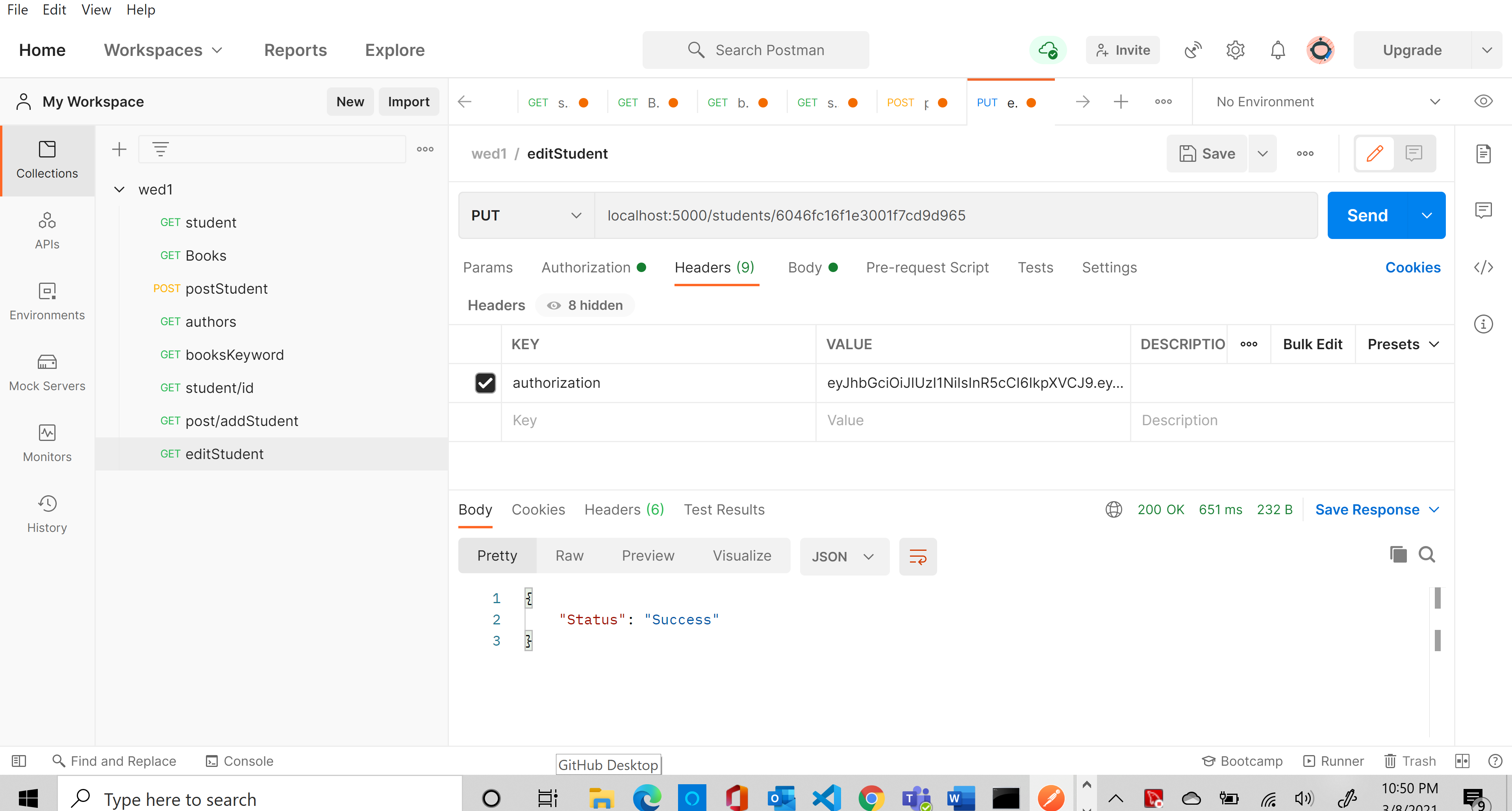
        })

})



Update student





Delete

router.delete('/:id', function (req, res) {

    const id = req.params.id;

    //const para = { 'first': req.body.fname, 'last': req.body.lname }

    //req.db.collection('students').find({ Fname: { $eq: id } }).toArray()

    req.db.collection('students').removeOne({ \_id: { $eq: new ObjectID(id) } })

        .then(data => {

            console.log('get')

            res.json({ Status: "Success" })

        })

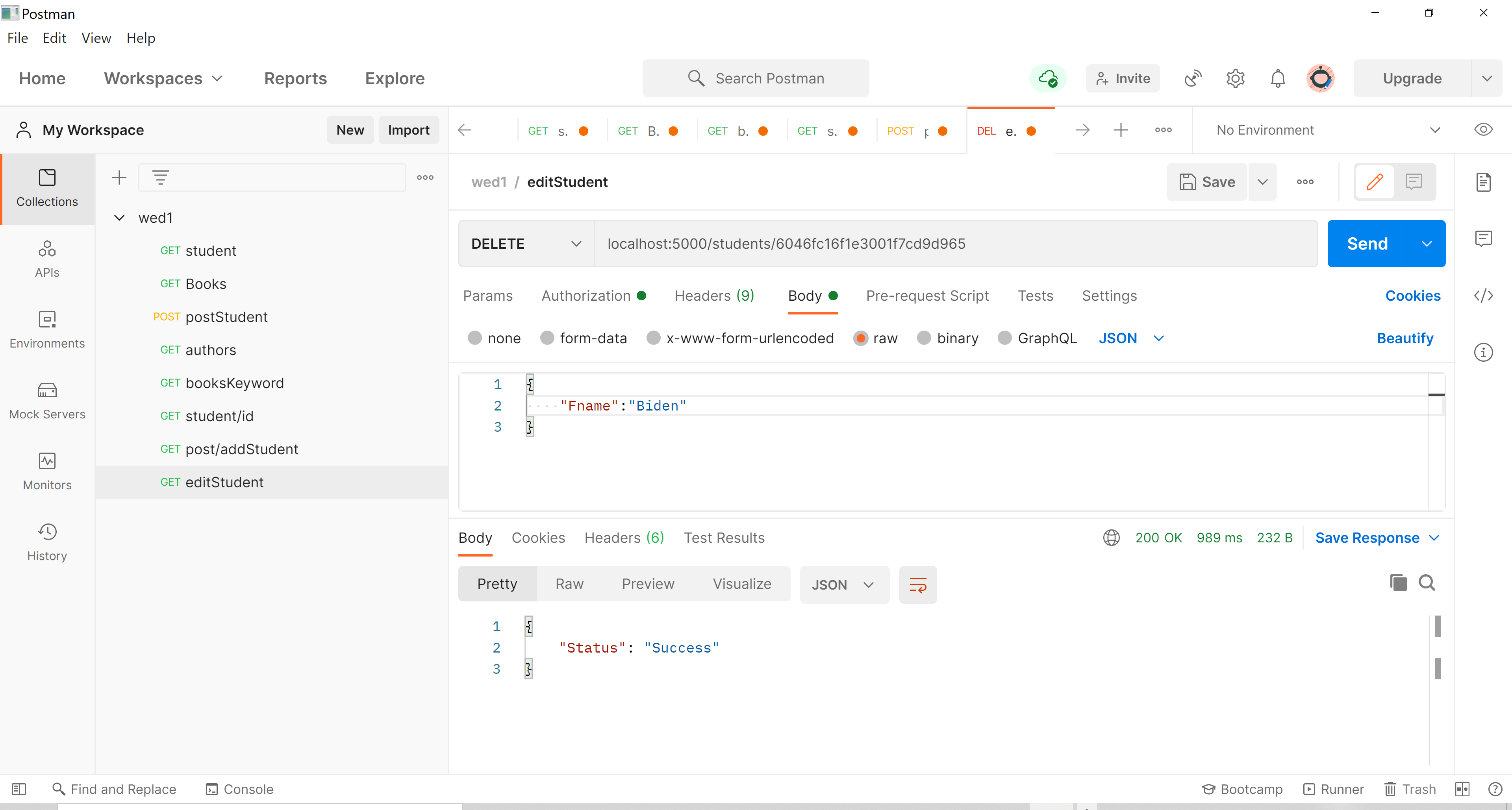
        .catch(err => {

            console.log(err)

            res.json({ status: "Error" })

        })

})



e. Books can be searched by a keyword

display having keyword sql

{

    "Status": "Success",

    "data": [

        {

            "\_id": "6046a06c51de294ac6f774bb",

            "Isbn": "78998",

            "title": "Server Side programing",

            "Author": [

                {

                    "Fname": "Umur",

                    "Lname": "Inan",

                    "\_id": "6046a17f51de294ac6f774bc"

                },

                {

                    "Fname": "Asaad",

                    "Lname": "Saad",

                    "\_id": "6046a1eb51de294ac6f774bd"

                }

            ],

            "keyword": [

                "express",

                "nodejs",

                "mongodb"

            ],

            "Borrowers": [

                {

                    "name": "Barack",

                    "email": "obarack@miu.edu",

                    "returnDate": "02-09"

                }

            ]

        }

    ]

}