Lab 9

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 written by authors 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('/:id', function (req, res, next) {

    const path = req.params.id;

    console.log('req/url', path)

    const fname = req.body.fname;

    const email = req.body.email;

    let collectionName;

    if (path === 'student') {

        collectionName = 'students';

        console.log('collectionName', collectionName)

    } else if (path === 'library') {

        collectionName = 'library';

    } else {

        res.json({ Status: 'No matching collection found' })

    }

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

    req.db.collection(collectionName).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`, data.\_id)

                const datas = {};

                datas.id = data.\_id

                datas.email = data.email

                datas.Fname = data.Fname

                datas.role = data.role

                const token = jwtManager.generate(datas);

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

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

            }

            else {

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

            }

        })

        .catch(err => {

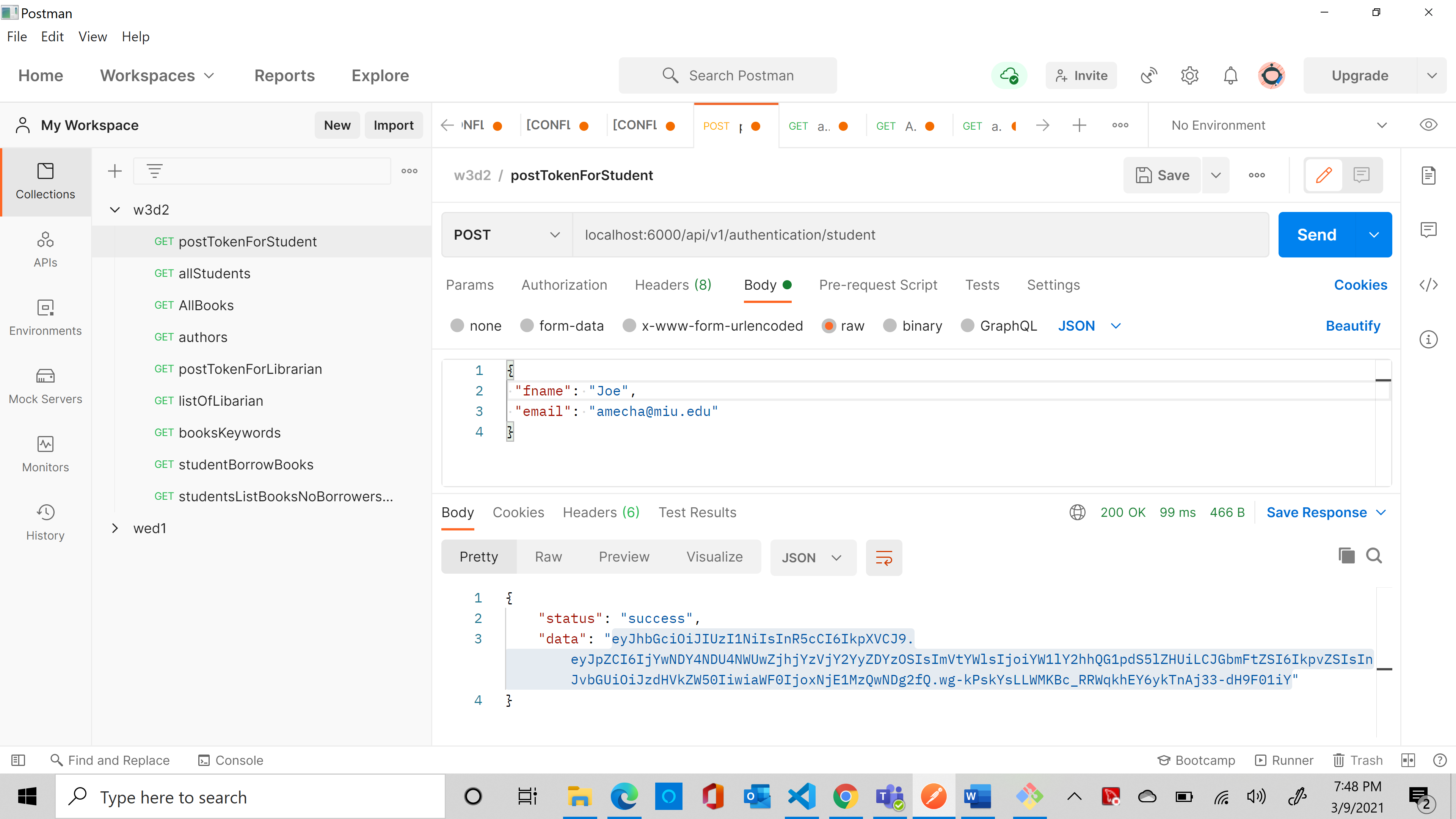
            console.log(err)

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

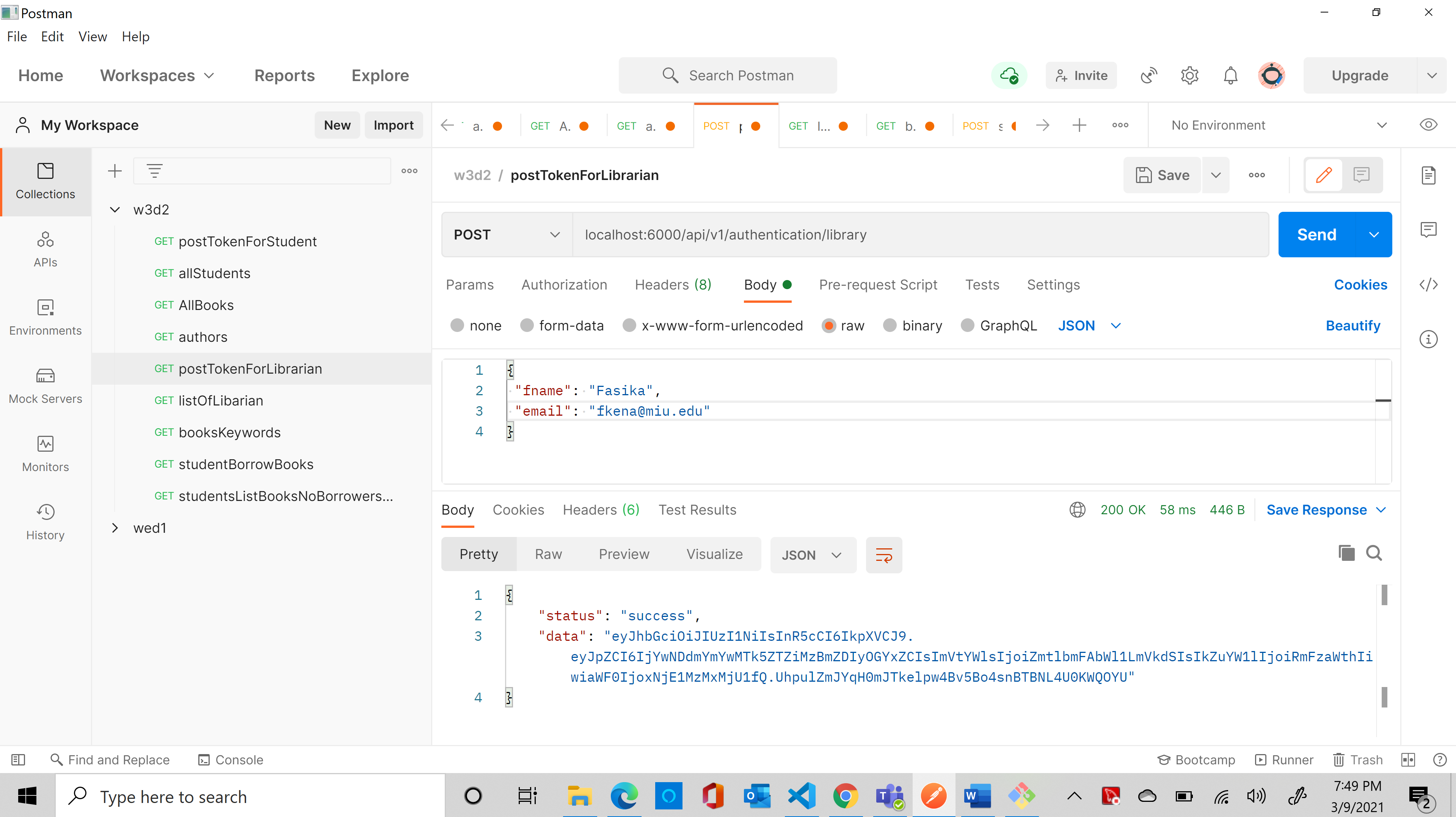
        })

});

module.exports = router;



b. Librarians can login to the system.



c. Secure your endpoints by using JWT.

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

class Authorization {

    authenticate(req, res, next) {

        if (req.url === '/api/v1/authentication/student' || req.url === '/api/v1/authentication/library') {

            next();

            return;

        }

        const header = req.headers.authorization;

        console.log('header', header)

        if (!header) {

            return res.json({ status: 'auth\_error' });

        } else {

            const data = jwtManager.verify(header);

            if (!data) {

                return res.json({ status: 'auth\_error' });

            }

            //authorization

            if (req.url === '/api/v1/students') {

                //console.log(`dkfkasfkajkfajkfja`, req.url)

                if (data.role == 'student') {

                    return next();

                } else {

                    return res.json({ status: 'auth\_error' });

                }

            } else if (req.url === '/api/v1/library') {

                if (data.role == 'librarian') {

                    next();

                } else {

                    return res.json({ status: 'auth\_error' });

                }

            }

            next();

        }

    }

}

module.exports = new Authorization();

d. Implement CRUD operations for the books.

e. Implement CRUD operation for the students.

f. Implement CRUD operation for the librarians.

g. Books can be searched by a keyword.

router.get('/:keyword', function (req, res) {

    console.log('req.params.keyword', req.params.keyword)

    const para = req.params.keyword;

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

    //req.db.collection('books').find({ title: { $eq: para } }).toArray()

    req.db.collection('books').find({ keyword: { $in: [para] } }).toArray()

        .then(data => {

            console.log('get')

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

        })

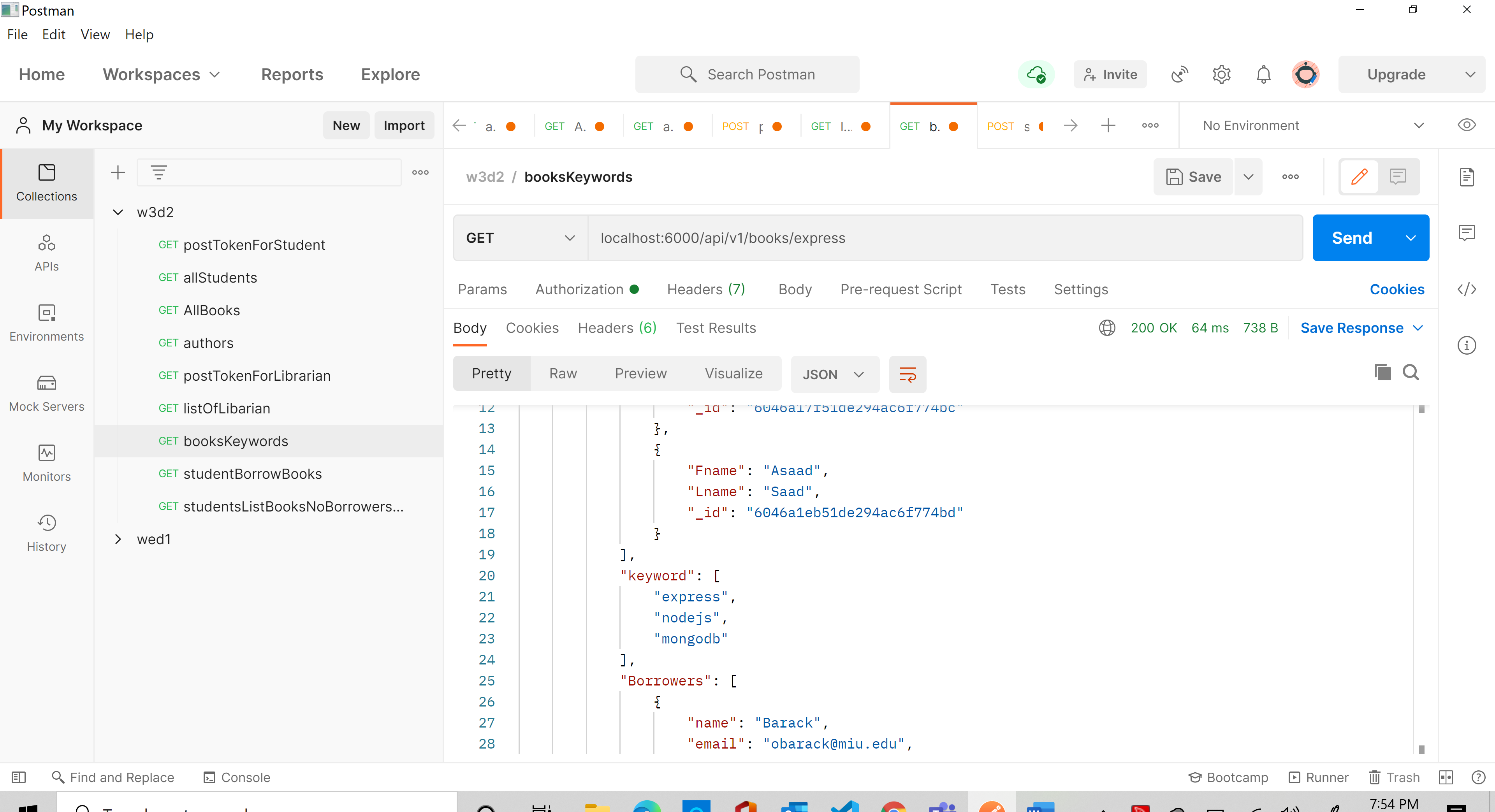
        .catch(err => {

            console.log(err)

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

        })

})



h. Students/ Librarians can sign up to the system. (hash the password using

bcrypt)

\

//my new code

const express = require('express');

const router = express.Router();

const bcrypt = require('bcryptjs');

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

    let docs = req.body;

    const role = req.body.role;

    let collectionName;

    if (role === 'student') {

        collectionName = 'students';

    } else if (role === 'librarian') {

        collectionName = 'library';

    } else {

        res.json({ status: "Sorry you can sign up to the system" })

    }

    req.db.collection(collectionName).findOne({ email: docs.email })

        .then(datas => {

            console.log('data', datas)

            if (!datas) {

                // const hashedPass = bcrypt.hashSync('docs.password', 12);

                // docs.password = hashedPass;

                req.db.collection(collectionName).insertOne(docs)

                    .then(data => {

                        console.log({ status: "inserted successfully" });

                        res.json({ status: "You are added to the system!" })

                    })

                    .catch(err => { console.log(err) })

            } else {

                res.json({ Status: "user already exist" })

            }

        })

        .catch(err => {

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

        })

    //const hashedPass = bcrypt.hashSync(password, 12);

})

module.exports = router

//students signed up successful

app.post('/api/v1/signUp', function (req, res) {

  const docs = req.body;

  bcrypt

    .hash(docs.password, 12)

    .then(hashedPassword => {

      docs.password = hashedPassword

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

        .then(data => {

          console.log({ status: "inserted successfully" });

          res.json({ status: "Your sign-up request was successful!" })

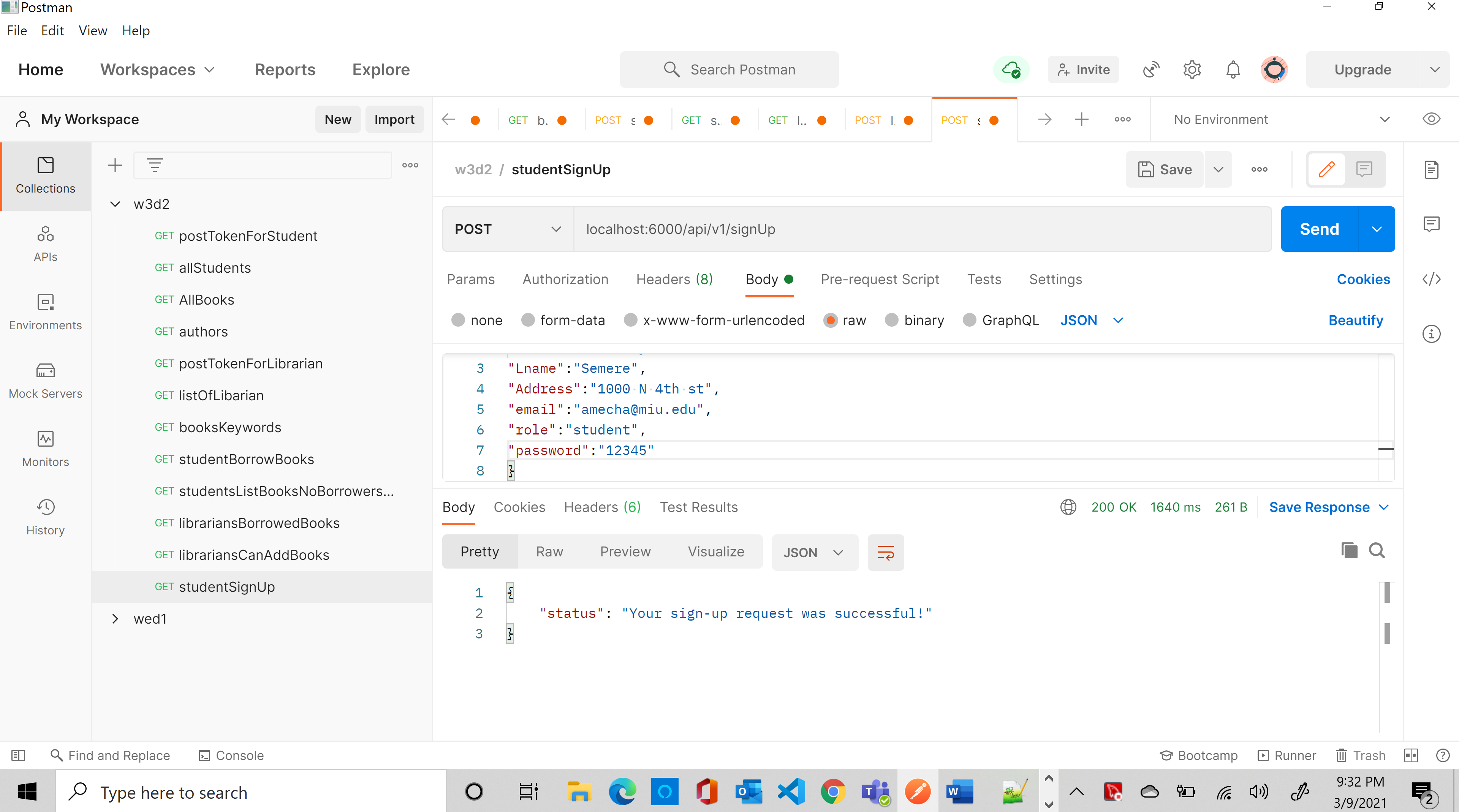
        })

        .catch(err => { console.log(err) })

    })

    .catch(err => { console.log(err) })

})



i. Librarians can

i. List all the borrowed books.

router.get('/borrowed', function (req, res) {

    req.db.collection('books').find({ Borrowers: { $ne: [] } }).toArray()

        .then(data => {

            console.log('get')

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

        })

        .catch(err => {

            console.log(err)

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

        })

})

{

    "Status": "Success",

    "data": [

        {

            "\_id": "60467d120d83297c64bf639d",

            "Isbn": "12345",

            "title": "Algorithm",

            "Author": {

                "Fname": "Joe",

                "Lname": "John",

                "\_id": "60467e640d83297c64bf639e"

            },

            "keyword": [

                "sql",

                "nodejs",

                "mongodb"

            ],

            "Borrowers": [

                {

                    "name": "Amanuel",

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

                    "returnDate": "02-09",

                    "\_id": "60469f4551de294ac6f774ba"

                },

                {

                    "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"

                }

            ],

            "keyword": [

                "express",

                "nodejs",

                "mongodb"

            ],

            "Borrowers": [

                {

                    "name": "Barack",

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

                    "returnDate": "02-09"

                },

                {

                    "\_id": "6046fbdc398e42651c09bb17",

                    "Fname": "Sofia",

                    "Lname": "Brain",

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

                    "address": "1112 georgia ave nw",

                    "returnDate": "03/19/2021"

                }

            ]

        }

    ]

}

router.get('/borrowed', function (req, res) {

    req.db.collection('books').find({ Borrowers: { $ne: [] } }).project({ title: 1, Borrowers: 1 }).toArray()

        .then(data => {

            console.log('get')

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

        })

        .catch(err => {

            console.log(err)

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

        })

})

{

    "Status": "Success",

    "data": [

        {

            "\_id": "60467d120d83297c64bf639d",

            "title": "Algorithm",

            "Borrowers": [

                {

                    "name": "Amanuel",

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

                    "returnDate": "02-09",

                    "\_id": "60469f4551de294ac6f774ba"

                },

                {

                    "name": "Amanuel",

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

                    "returnDate": "02-09",

                    "\_id": "60469f4551de294ac6f774ba"

                },

                {}

            ]

        },

        {

            "\_id": "6046a06c51de294ac6f774bb",

            "title": "Server Side programing",

            "Borrowers": [

                {

                    "name": "Barack",

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

                    "returnDate": "02-09"

                },

                {

                    "\_id": "6046fbdc398e42651c09bb17",

                    "Fname": "Sofia",

                    "Lname": "Brain",

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

                    "address": "1112 georgia ave nw",

                    "returnDate": "03/19/2021"

                }

            ]

        }

    ]

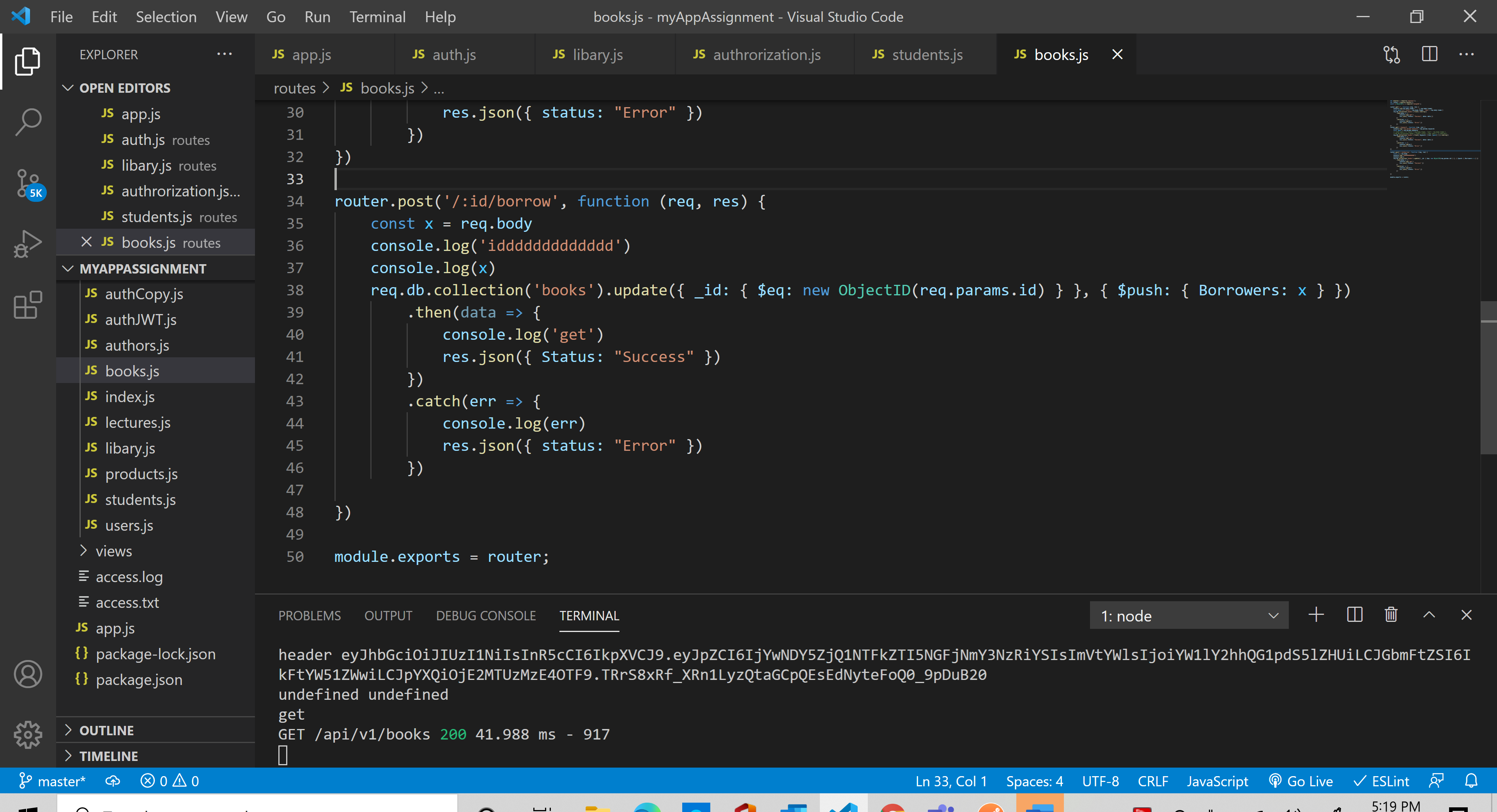
}

//librarian can add books

ii. Execute CRUD operations on the book repository.

j. Students can

i. Borrow books.



{

    "Status": "Success",

    "data": [

        {

            "\_id": "60467d120d83297c64bf639d",

            "Isbn": "12345",

            "title": "Algorithm",

            "Author": {

                "Fname": "Joe",

                "Lname": "John",

                "\_id": "60467e640d83297c64bf639e"

            },

            "keyword": [

                "sql",

                "nodejs",

                "mongodb"

            ],

            "Borrowers": [

                {

                    "name": "Amanuel",

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

                    "returnDate": "02-09",

                    "\_id": "60469f4551de294ac6f774ba"

                },

                {

                    "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"

                }

            ],

            "keyword": [

                "express",

                "nodejs",

                "mongodb"

            ],

            "Borrowers": [

                {

                    "name": "Barack",

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

                    "returnDate": "02-09"

                },

                {

                    "\_id": "6046fbdc398e42651c09bb17",

                    "Fname": "Sofia",

                    "Lname": "Brain",

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

                    "address": "1112 georgia ave nw",

                    "returnDate": "03/19/2021"

                }

            ]

        }

    ]

}

ii. List all the books

{

    "Status": "Success",

    "data": [

        {

            "\_id": "60467d120d83297c64bf639d",

            "Isbn": "12345",

            "title": "Algorithm",

            "Author": {

                "Fname": "Joe",

                "Lname": "John",

                "\_id": "60467e640d83297c64bf639e"

            },

            "keyword": [

                "sql",

                "nodejs",

                "mongodb"

            ]

        },

        {

            "\_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"

            ]

        }

    ]

}