Lab 10

a-)Considering the following restaurants collection that has information about all

restaurants in the USA. Import the data into a local/cloud DB server.

{

"address": { "building": "1007",

"coord": [ -73.856077, 40.848447 ],

"street": "Morris Park Ave",

"zipcode": "10462" },

"district": "Bronx",

"cuisine": "Bakery",

"grades": [ {"date": {"$date": 1393804800000}, "grade": "A", "score": 2},

{"date": {"$date": 1378857600000}, "grade": "A", "score": 6},

{"date": {"$date": 1358985600000}, "grade": "A", "score": 10},

{"date": {"$date": 1322006400000}, "grade": "A", "score": 9},

{"date": {"$date": 1299715200000}, "grade": "B", "score": 14}],

"name": "Morris Park Bake Shop",

"restaurant\_id": "30075445"

}

1. Add at least 5 restaurants to your collection to test your queries.
2. router.post('/', (req, res) => {
3. req.db.collection('restaurants').insertOne(req.body)
4. .then(data => {
5. res.json({ status: "restaurants added successfully" })
6. })
7. .catch(err => {
8. res.json(err)
9. })
10. })

2- Write a MongoDB query to find the restaurants that does not prepare

any cuisine of "American" and their grade score more than 70

router.get('/2', (req, res) => {

    req.db.collection('restaurants').find({ cuisine: { $ne: 'American' }, 'grades.score': { $gt: 10 } }).toArray()

        .then(data => {

            res.json(data)

        })

        .catch(err => {

            res.json(err)

        })

})

3- Write a MongoDB query to find the restaurant\_id, name, district and cuisine for

those restaurants which contains 'Wil' as first three letters for its name.

router.get('/3', (req, res) => {

    req.db.collection('restaurants').find({ 'name': /^wil/ }).project({ 'restaurant\_id': 1, 'name': 1, 'district': 1, 'cuisine': 1 }).toArray()

        .then(data => {

            res.json(data)

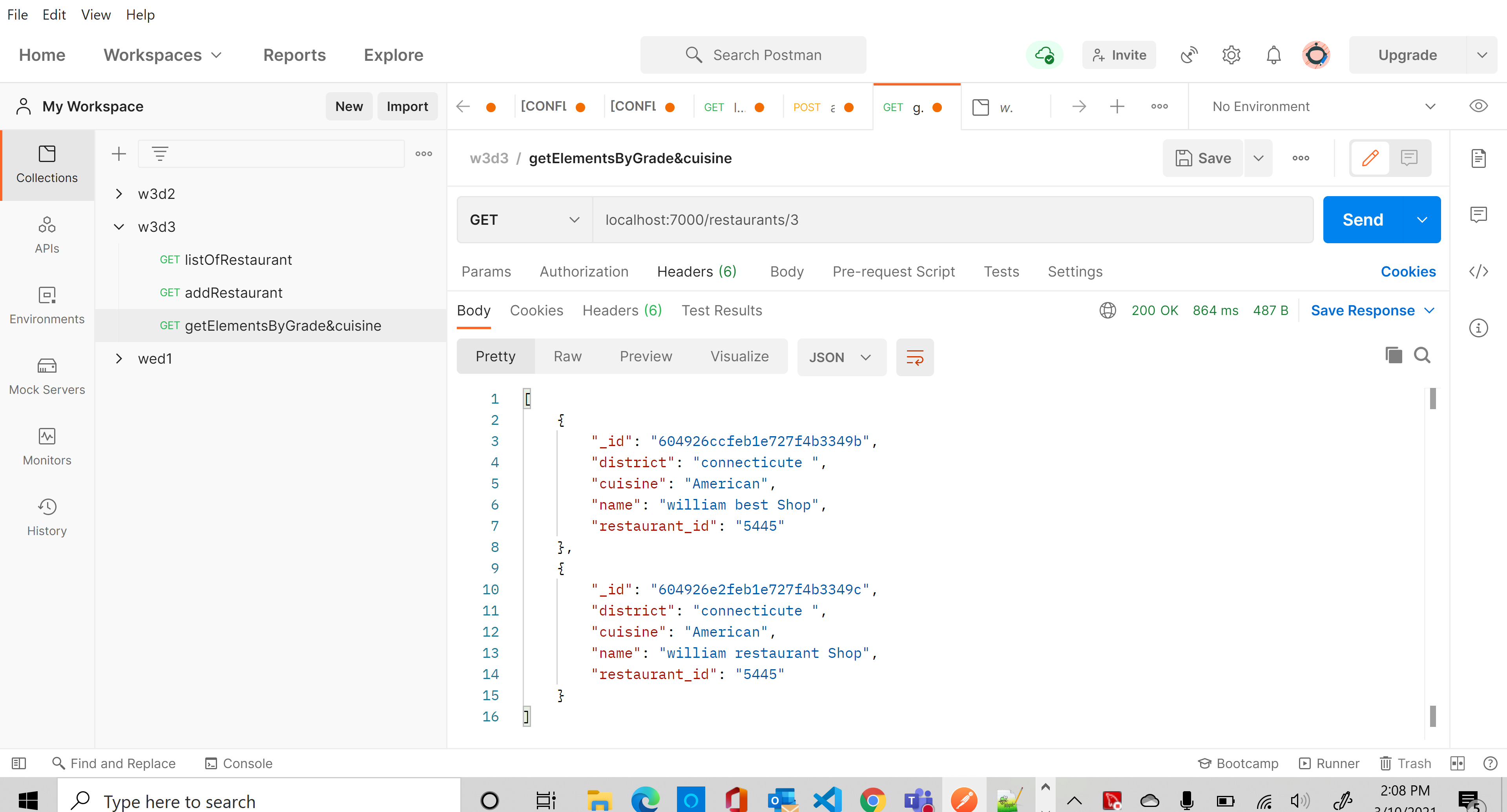
        })

        .catch(err => {

            res.json(err)

        })

})



4- Write a MongoDB query to find the restaurant\_id, name, district and cuisine for

those restaurants which contains 'Reg' as three letters somewhere in its name

router.get('/4', (req, res) => {

req.db.collection('restaurants').find({ 'name': /.\*Reg.\*/ }).project({ 'restaurant\_id': 1, 'name': 1, 'district': 1, 'cuisine': 1 }).toArray()

        .then(data => {

            res.json(data)

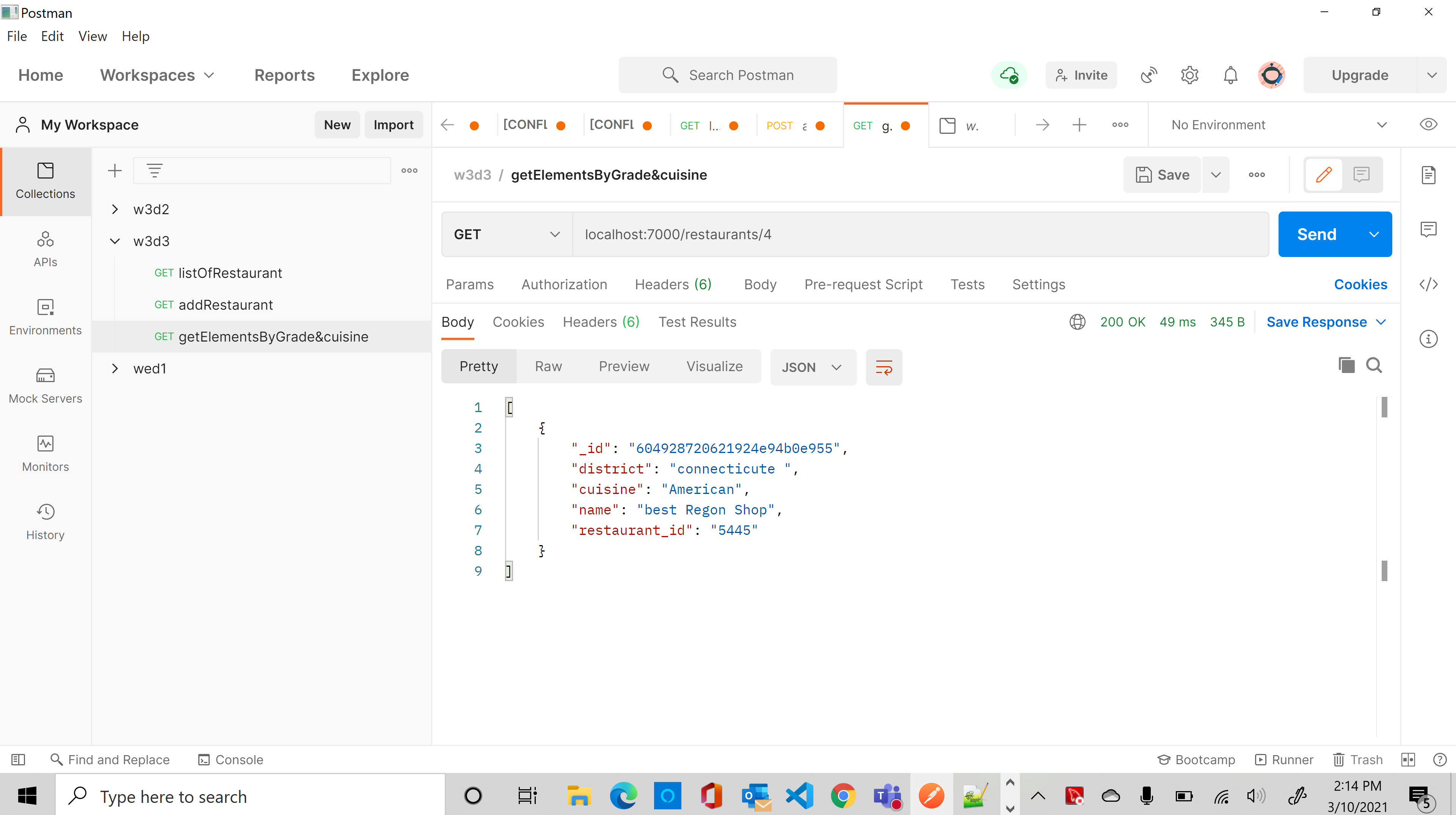
        })

        .catch(err => {

            res.json(err)

        })

})



5- Write a MongoDB query to find the restaurants which belongs to

the district "Bronx" and prepared either American or Chinese dish.

router.get('/5', (req, res) => {

    // req.db.collection('restaurants').find({ 'district': 'Bronx', $or: [{ 'cuisine': 'American','cuisine': 'Chinese'}] }).project({ 'restaurant\_id': 1, 'name': 1, 'district': 1, 'cuisine': 1 }).toArray()

    req.db.collection('restaurants').find({ 'district': 'Bronx', 'cuisine': { $in: ['American', 'Chinese'] } }).toArray()

        .then(data => {

            res.json(data)

        })

        .catch(err => {

            res.json(err)

        })

})

[

    {

        "\_id": "60492aba5bf6f84704f9e6f7",

        "address": {

            "building": "1007",

            "coord": [

                -73.856077,

                40.848447

            ],

            "street": "connecticute Park Ave",

            "zipcode": "52257"

        },

        "district": "Bronx",

        "cuisine": "American",

        "grades": [

            {

                "date": "2014-03-03T00:00:00.000Z",

                "grade": "A",

                "score": 12

            },

            {

                "date": "2013-09-11T00:00:00.000Z",

                "grade": "A",

                "score": 16

            },

            {

                "date": "2013-01-24T00:00:00.000Z",

                "grade": "A",

                "score": 13

            },

            {

                "date": "2011-11-23T00:00:00.000Z",

                "grade": "A",

                "score": 934

            },

            {

                "date": "2011-03-10T00:00:00.000Z",

                "grade": "B",

                "score": 124

            }

        ],

        "name": "best Regon Shop",

        "restaurant\_id": "5445"

    }

]

6- Write a MongoDB query to find the restaurant\_id, name, district and cuisine for

those restaurants which belongs to the district "Staten

Island" or "Queens" or "Bronx" or "Brooklyn".

router.get('/6', (req, res) => {

    req.db.collection('restaurants').find({ 'district': { $in: ['Bronx', 'Queens', 'Brooklyn', 'Staten Island'] } })

        .project({ "restaurant\_id": 1, "name": 1, "district": 1, "cuisine": 1 }).toArray()

        .then(data => {

            res.json(data)

        })

        .catch(err => {

            res.json(err)

        })

})

[

    {

        "\_id": "604914a66ac197e48885c1c6",

        "district": "Bronx",

        "cuisine": "Bakery",

        "name": "Morris Park Bake Shop",

        "restaurant\_id": "30075445"

    },

    {

        "\_id": "60492aba5bf6f84704f9e6f7",

        "district": "Bronx",

        "cuisine": "American",

        "name": "best Regon Shop",

        "restaurant\_id": "5445"

    },

    {

        "\_id": "60492ac75bf6f84704f9e6f8",

        "district": "Bronx",

        "cuisine": "chinese best",

        "name": "best Regon Shop",

        "restaurant\_id": "5445"

    },

    {

        "\_id": "60495743eac5bb3298779afa",

        "district": "Brooklyn",

        "cuisine": "Japan best",

        "name": "best Regon Shop",

        "restaurant\_id": "5445"

    }

]

7- Write a MongoDB query to find the restaurant\_id, name, district and cuisine for

those restaurants which are not belonging to the district "Staten

Island" or "Queens" or "Bronx" or "Brooklyn".

router.get('/7', (req, res) => {

    req.db.collection('restaurants').find({ 'district': { $nin: ['Bronx', 'Queens', 'Brooklyn', 'Staten Island'] } })

        .project({ "restaurant\_id": 1, "name": 1, "district": 1, "cuisine": 1 }).toArray()

        .then(data => {

            res.json(data)

        })

        .catch(err => {

            res.json(err)

        })

})

[

    {

        "\_id": "60491927e627cb391c51770e",

        "district": "washington",

        "cuisine": "Ethiopian",

        "name": "Georgia Shop",

        "restaurant\_id": "660075445"

    },

    {

        "\_id": "6049199de627cb391c51770f",

        "district": "silver spring",

        "cuisine": "Ertria",

        "name": "Atlanta Shop",

        "restaurant\_id": "12660075445"

    },

    {

        "\_id": "60491a08e627cb391c517710",

        "district": "connecticute ",

        "cuisine": "American",

        "name": "best Shop",

        "restaurant\_id": "5445"

    },

    {

        "\_id": "60491a62e627cb391c517711",

        "district": "connecticute ",

        "cuisine": "American",

        "name": "best Shop",

        "restaurant\_id": "5445"

    },

    {

        "\_id": "604926ccfeb1e727f4b3349b",

        "district": "connecticute ",

        "cuisine": "American",

        "name": "william best Shop",

        "restaurant\_id": "5445"

    },

    {

        "\_id": "604926e2feb1e727f4b3349c",

        "district": "connecticute ",

        "cuisine": "American",

        "name": "william restaurant Shop",

        "restaurant\_id": "5445"

    },

    {

        "\_id": "604928500621924e94b0e953",

        "district": "connecticute ",

        "cuisine": "American",

        "name": "best reginal Shop",

        "restaurant\_id": "5445"

    },

    {

        "\_id": "604928630621924e94b0e954",

        "district": "connecticute ",

        "cuisine": "American",

        "name": "best Reagon Shop",

        "restaurant\_id": "5445"

    },

    {

        "\_id": "604928720621924e94b0e955",

        "district": "connecticute ",

        "cuisine": "American",

        "name": "best Regon Shop",

        "restaurant\_id": "5445"

    }

]

8- Write a MongoDB query to find the restaurant\_id, name, district and cuisine for

those restaurants which achieved a score which is not more than 10.

router.get('/8', (req, res) => {

    req.db.collection('restaurants').find({ 'grades.score': { $lte: 10 } })

        .project({ "restaurant\_id": 1, "name": 1, "district": 1, "cuisine": 1 }).toArray()

        .then(data => {

            res.json(data)

        })

        .catch(err => {

            res.json(err)

        })

})

[

    {

        "\_id": "604914a66ac197e48885c1c6",

        "district": "Bronx",

        "cuisine": "Bakery",

        "name": "Morris Park Bake Shop",

        "restaurant\_id": "30075445"

    },

    {

        "\_id": "60491927e627cb391c51770e",

        "district": "washington",

        "cuisine": "Ethiopian",

        "name": "Georgia Shop",

        "restaurant\_id": "660075445"

    },

    {

        "\_id": "6049199de627cb391c51770f",

        "district": "silver spring",

        "cuisine": "Ertria",

        "name": "Atlanta Shop",

        "restaurant\_id": "12660075445"

    }

]

b-) Create an Express application that implements a Restful Stateless API for an entity

called students as following:

const students = [{id: 1, name: "Asaad Saad", course: "CS572",grade: 95}]

• Write routes for the following CRUD operations and use the proper HTTP verbs

(GET one and all, POST, and DELETE).

Get all

router.get('/', (req, res) => {

    req.db.collection('students').find().toArray()

        .then(data => {

            res.json(data)

        })

        .catch(err => {

            res.json(err)

        })

})

[

    {

        "\_id": "60495eb24c67dca38c9b33c0",

        "id": "1",

        "name": "Asaad Saad",

        "course": "CS572",

        "grade": "95"

    },

    {

        "\_id": "604960c94c67dca38c9b33c1",

        "id": "2",

        "name": "Umur Inan",

        "course": "CS477",

        "grade": "98"

    }

]

Get one

router.get('/:id', (req, res) => {

    console.log('amama', req.params.id)

    req.db.collection('students').findOne({ '\_id': new ObjectID(req.params.id) })

        .then(data => {

            res.json(data)

        })

        .catch(err => {

            res.json(err)

        })

})

{

    "\_id": "604960c94c67dca38c9b33c1",

    "id": "2",

    "name": "Umur Inan",

    "course": "CS477",

    "grade": "98"

}

Post

router.post('/', (req, res) => {

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

        .then(data => {

            res.json({ status: "Added successfully" })

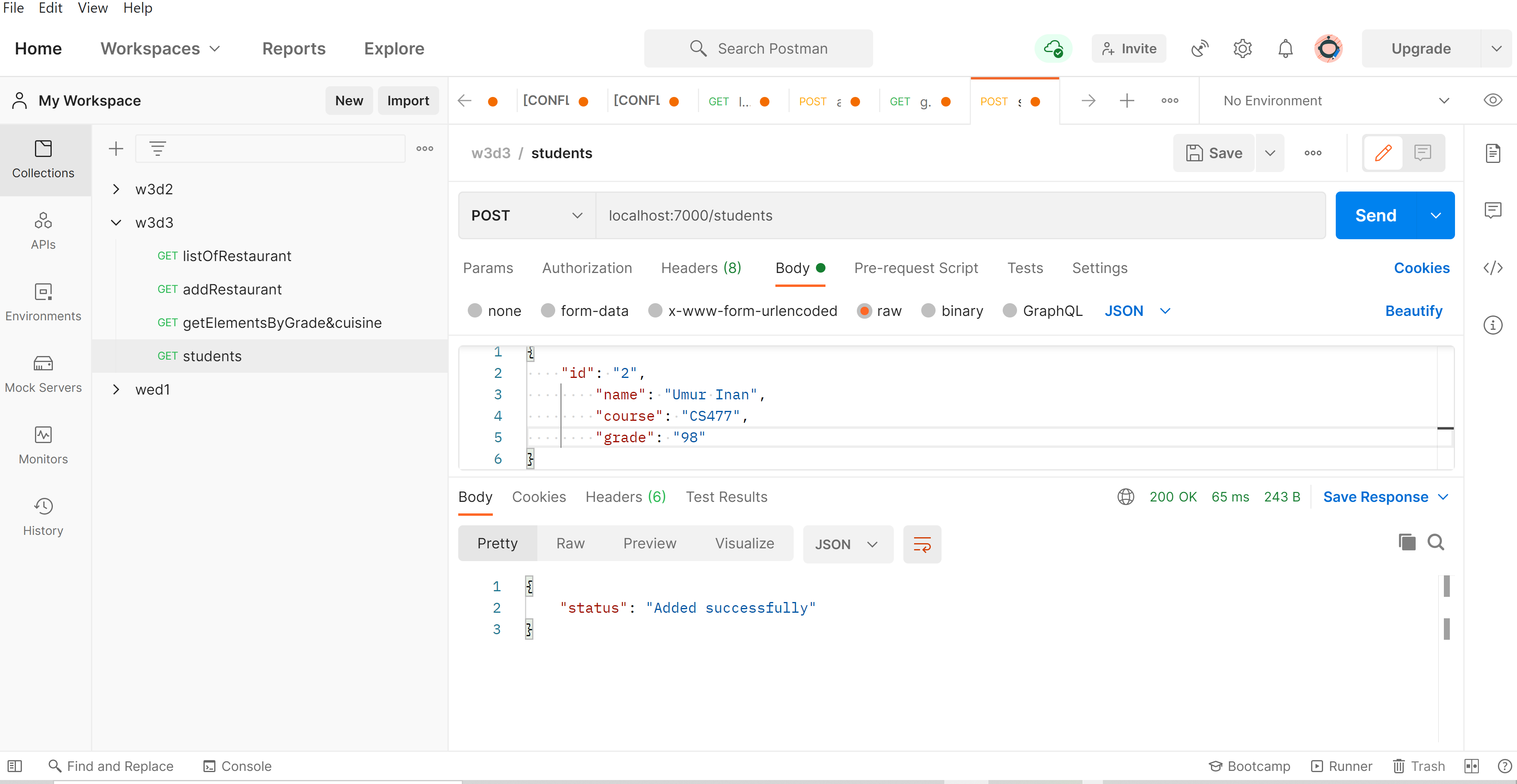
        })

        .catch(err => {

            res.json(err)

        })

})



Delete

router.delete('/:id', (req, res) => {

    console.log('amama', req.params.id)

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

        .then(data => {

            res.json({ status: "Deleted successfuly " })

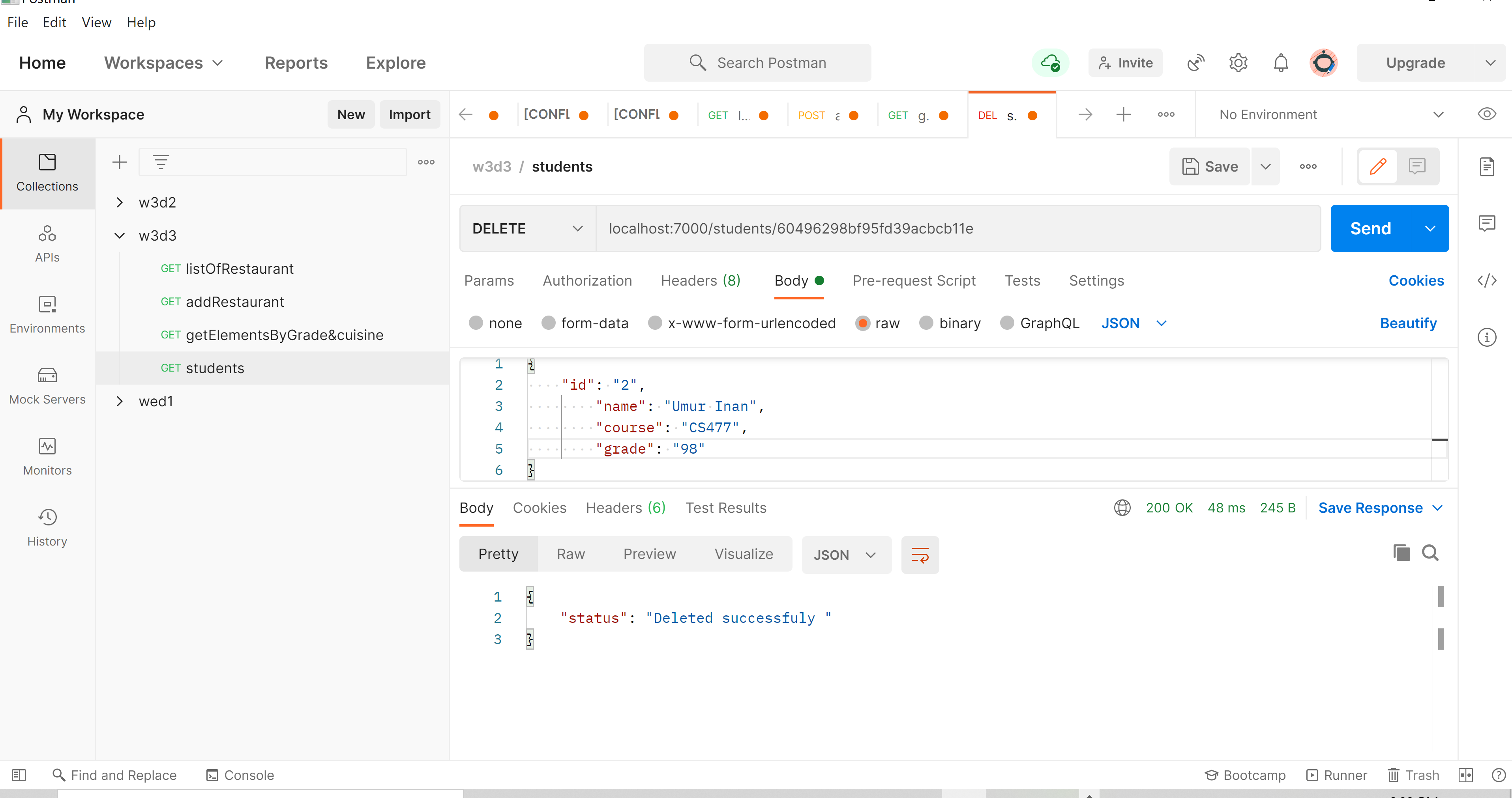
        })

        .catch(err => {

            res.json(err)

        })

})

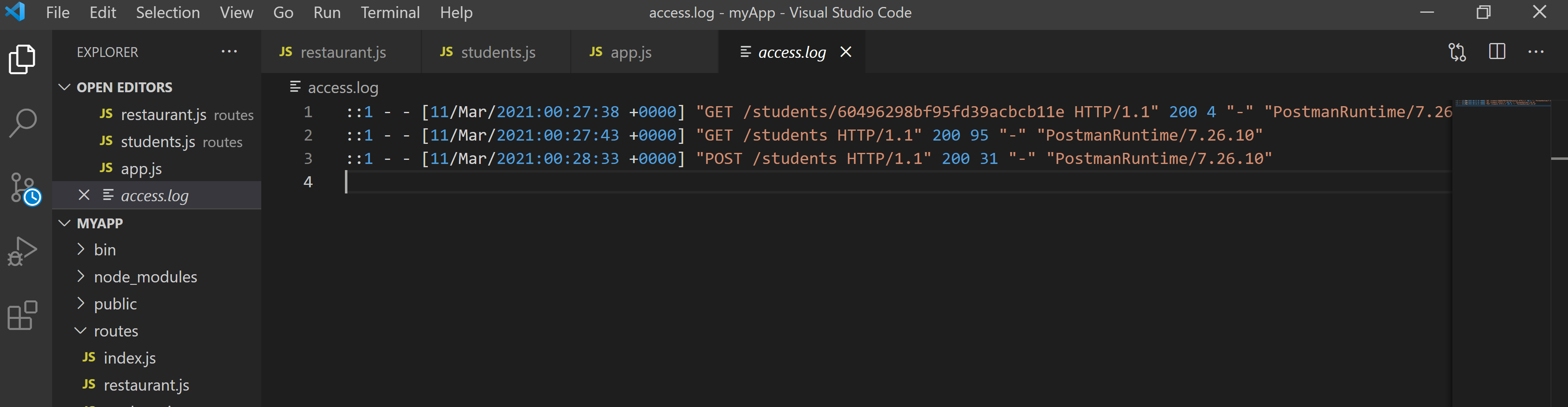


• Log all requests to a file access.log using morgan middleware.

const accessLogStream = fs.createWriteStream(

  path.join(\_\_dirname, 'access.log'), { flags: 'a' });

app.use(morgan('combined', { stream: accessLogStream }));



• For your POST route, assign a custom middleware to verify if a user passes

a JSON object that contains id, name, course and grade, otherwise send back an

error

function checkValidity(req, res, next) {

    if (req.body.id === "" || req.body.name === "" || req.body.course === "" || req.body.grade === "") {

        res.json({ Status: "invalid body" })

    } else {

        next()

    }

}

router.post('/', checkValidity, (req, res) => {

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

        .then(data => {

            res.json({ status: "Added successfully" })

        })

        .catch(err => {

            res.json(err)

        })

})

