**Create a NodeJS application to provide endpoints for the collections studentInfo and studentAcademicInfo with a smellier structure of objects created in Assignment 6. Install required packages like – express, mongoose, nodemon and perform the following operations. Handle asynchronous behaviour and exceptions properly.**

1. **Define schema and model for studnetInfo where every field must be required and rollno, mobile, and email must be unique.**

**Ans-**

const mongoose = require("mongoose");

const StudentSchema = new mongoose.Schema(

{

name: {

type: String,

required: true,

},

rollno: {

type: Number,

required: true,

unique: true,

},

mobile: {

type: String,

required: true,

unique: true,

},

email: {

type: String,

required: true,

unique: true,

},

address: {

city: {

type: String,

required: true,

},

state: {

type: String,

required: true,

},

pin: {

type: String,

required: true,

},

},

},

{

timestamps: true,

}

);

const StudentModel = new mongoose.model("StudentInfo", StudentSchema);

module.exports = StudentModel;

**2. Define schema and model for studentAcademicInfo where every field is required and rollno is unique again.**

**Ans-**

const mongoose = require("mongoose");

const StudentAcademicSchema = new mongoose.Schema(

{

rollno: {

type: Number,

unique: true,

ref: "StudentInfo",

},

program:{

type:String,

required:true

},

branch:{

type:String,

required:true

},

cgpa:{

type:String,

required:true

}

},

{ timestamps: true }

);

const StudentAcademicModel = new mongoose.model("StudentAcademicInfo", StudentAcademicSchema);

module.exports = StudentAcademicModel;

**3. Create a file to establish the connection with MongoDB and select the database.**

**Ans –**

//connect to mongo db

const mongoose = require("mongoose");

const dbConnect = async () => {

const DB\_URL =

"mongodb+srv://aniket96:pupulaniket420@cluster0.mksf4r6.mongodb.net";

const DB = "csea";

try {

await mongoose.connect(DB\_URL + "/" + DB);

console.log("MongoDB Connected");

} catch (error) {

console.error(error);

}

};

module.exports = dbConnect;

**4. Create a controller file for studentInfo and add functionalities for –**

**a. Add a student’s info**

**Ans-**

const addStudentInfo = async(req,res) => {

try {

console.log("add a student");

let data = req.body;

await StudentModel.create(data);

res.status(201).json({ message: "Data Added Successfully!!" });

} catch (error) {

res.status(400).json({ message: "Error in sending Data!!" + "\n" + error });

}

}

**b. Retrieve all students' records**

**Ans-**

const getAllStudentInfos = async (req, res) => {

try {

console.log("get all student info called");

const studentinfos = await StudentModel.find();

res.status(200).json(studentinfos);

} catch (error) {

res.status(500).json(error);

}

};

**c. Retrieve a single student based on roll number**

**Ans-**

const getStudentInfoByRoll = async(req,res) => {

try{

const id = req.params.id

const student = await StudentModel.find( { rollno : id } )

res.status(201).json(student)

}catch(error){

res.status(400).json({message:"Student not found!!"})

}

}

**d. Update a student based on roll number**

**Ans-**

const updateStudent = async(req,res) => {

try {

const id = req.params.id

const updatedStudent = await StudentModel.findOneAndUpdate({"rollno":id},req.body,{new:true});

if(updatedStudent){

res.status(200).json(updatedStudent);

}else{

res.status(404).json({message:'No Student with this rollno is present'});

}

} catch (error) {

res.status(500).json(error)

}

}

**e. Delete a student based on roll number**

**Ans-**

const deleteStudent = async(req,res) => {

try {

const id = req.params.id

const deletedStudent = await StudentModel.findOneAndDelete( {"rollno":id} );

if(!deletedStudent){

return res.status(404).json('No such user present')

}

res.status(200).json(deletedStudent);

} catch (error) {

res.status(500).json(error)

}

}

**5. Create a controller file for studentAcademicInfo and add functionalities for –**

**a. Add a student’s academic record**

**Ans-**

const addStudentAcademicRecord = async(req,res) => {

try{

const studentAcademicRecord = await StudentAcademicModel.create(req.body);

if(!studentAcademicRecord){

return res.status(400).send("Failed to create student Academic Record.");

}

res.status(200).json(studentAcademicRecord);

}catch(error){

res.status(500).json(error)

}

}

**b. Retrieve all students’ academic records**

**Ans-**

const getAllStudentAcademicInfos = async (req, res) => {

try {

const academicRecords = await StudentAcademicModel.find()

res.status(201).json(academicRecords);

} catch (error) {

res.status(400).json({ message: "Error in retrieving Data!!" + "\n" + error });

}

};

**c. Retrieve a single student’s academic record based on roll number**

**Ans-**

const getStudentAcademicRecordByRoll = async(req,res) => {

try {

const id = req.params.id

const studentAcademicRecord = await StudentAcademicModel.find({rollno:id})

if(studentAcademicRecord){

res.status(200).json(studentAcademicRecord)

}else{

res.status(400).json({message:"No Data Found!!"})

}

} catch (error) {

res.status(500).json(error)

}

}

**d. Update a student’s academic record based on roll number**

**Ans-**

const updateStudentAcademicRecordByRoll = async(req,res) => {

try {

const id = req.params.id

const updatedRecord = await StudentAcademicModel.findOneAndUpdate({rollno:id},req.body,{new:true})

if(updatedRecord){

res.status(200).json(updatedRecord)

}else{

res.status(400).json({message :'Data not found!'});

}

} catch (error) {

res.status(500).json(error)

}

}

**e. Delete a student’s academic record based on roll number**

**Ans-**

const deleteStudentAcademicRecordByRoll = async(req,res) => {

try {

const id = req.params.id

const deletedRecord = await StudentAcademicModel.findOneAndDelete( { rollno: id } )

if (!deletedRecord) {

return res.status(400).send("No Record with the provided ID was found.")

} else {

res.status(200).json({message:'Deletion Successful'})

}

} catch (error) {

res.status(500).json(error)

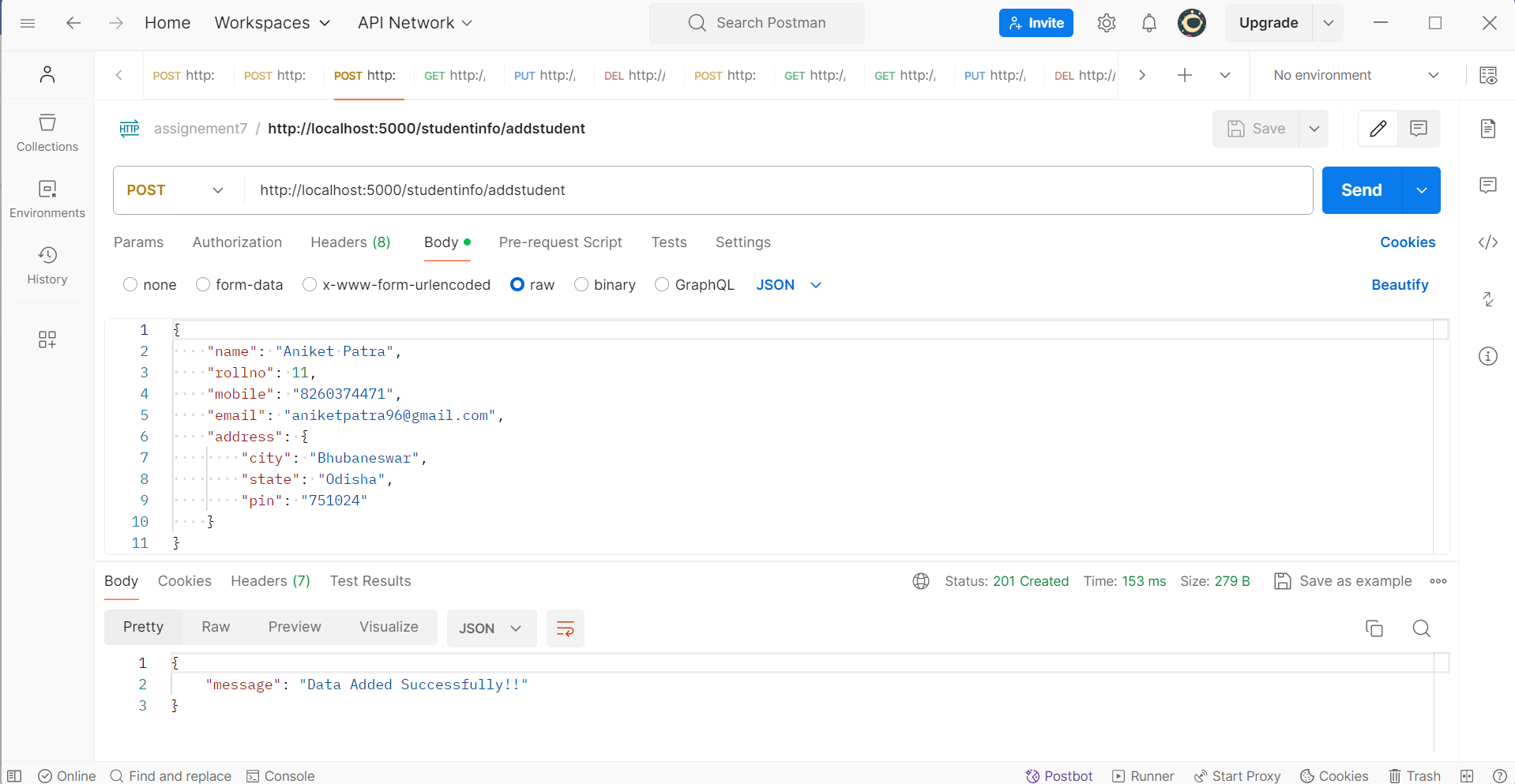
}

}

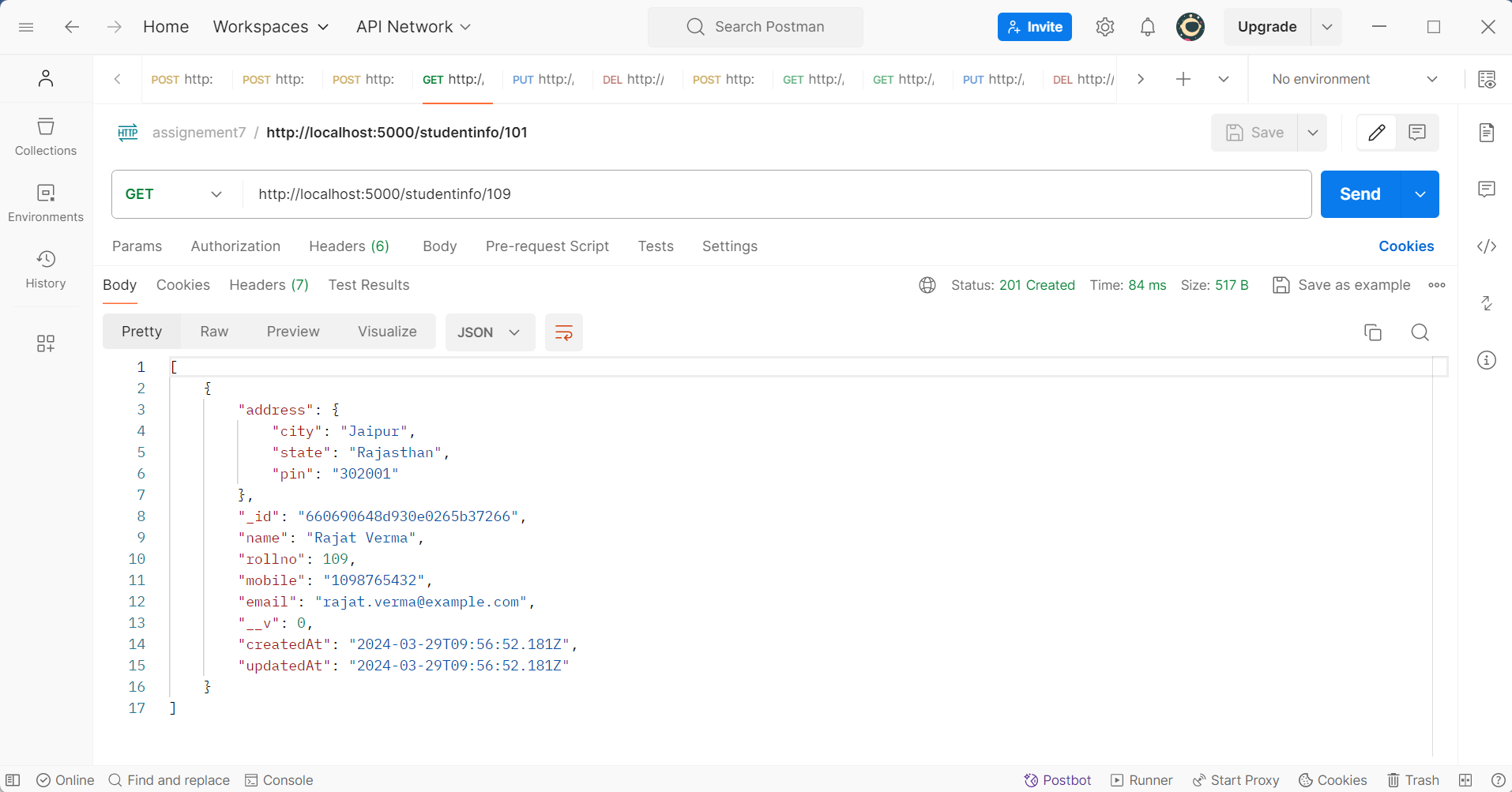
**6. Now use the controllers and provide end-points for all the functionalities and test the endpoints with Postman**

**Ans-**

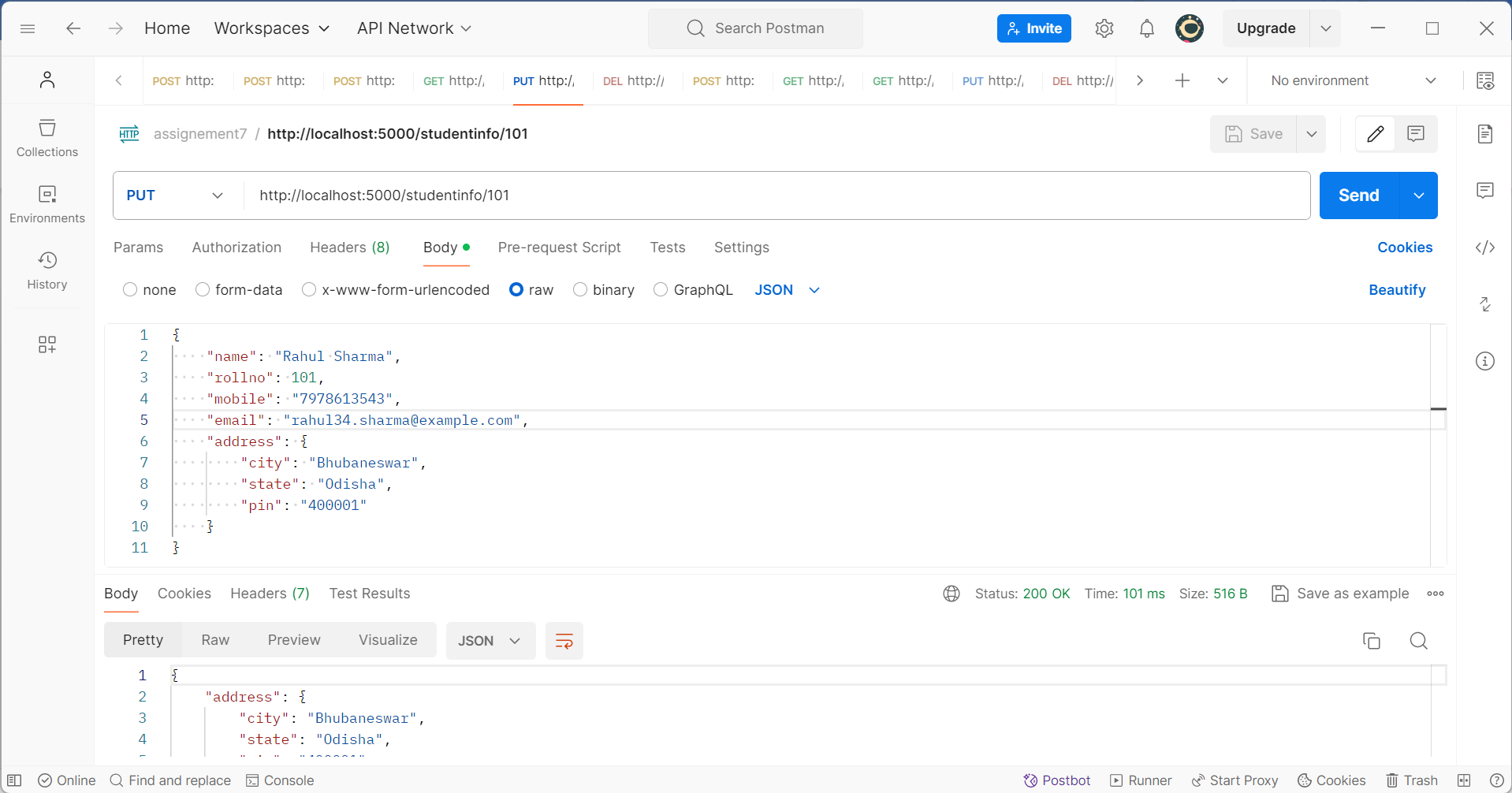
**AddStudentInfo**



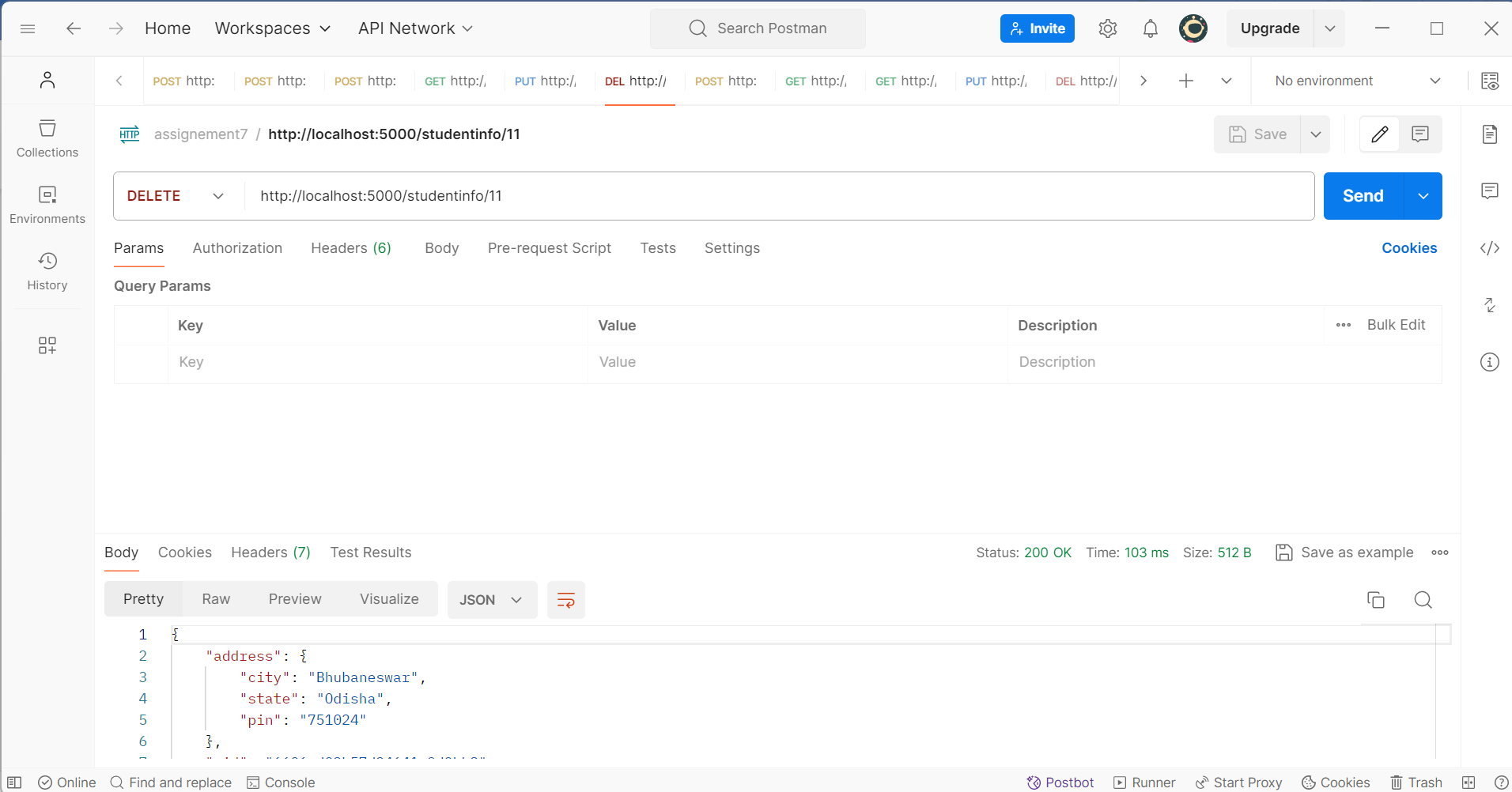
**GetStudentInfoByRoll**

****

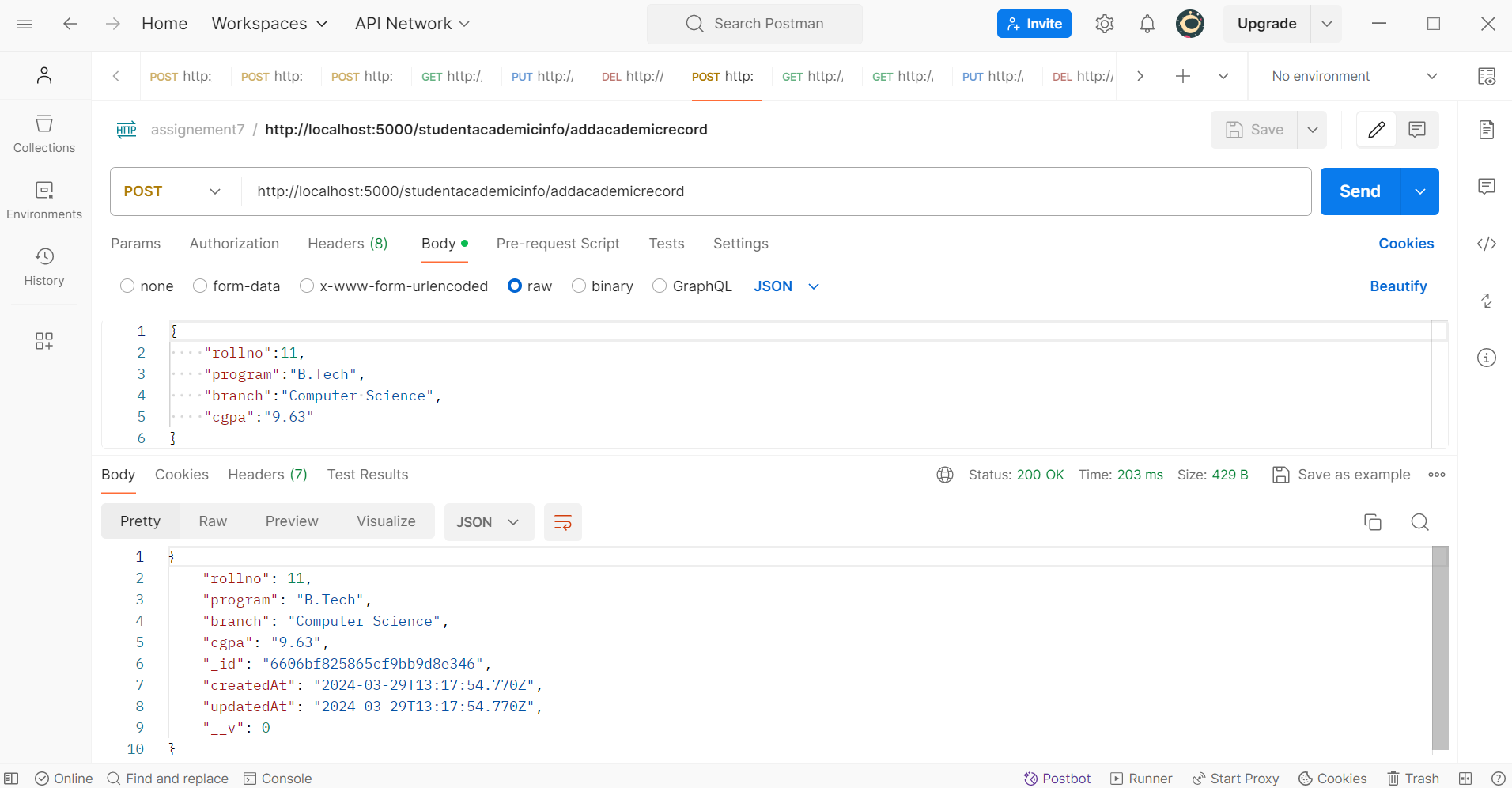
**UpdateStudentInfoByRoll**



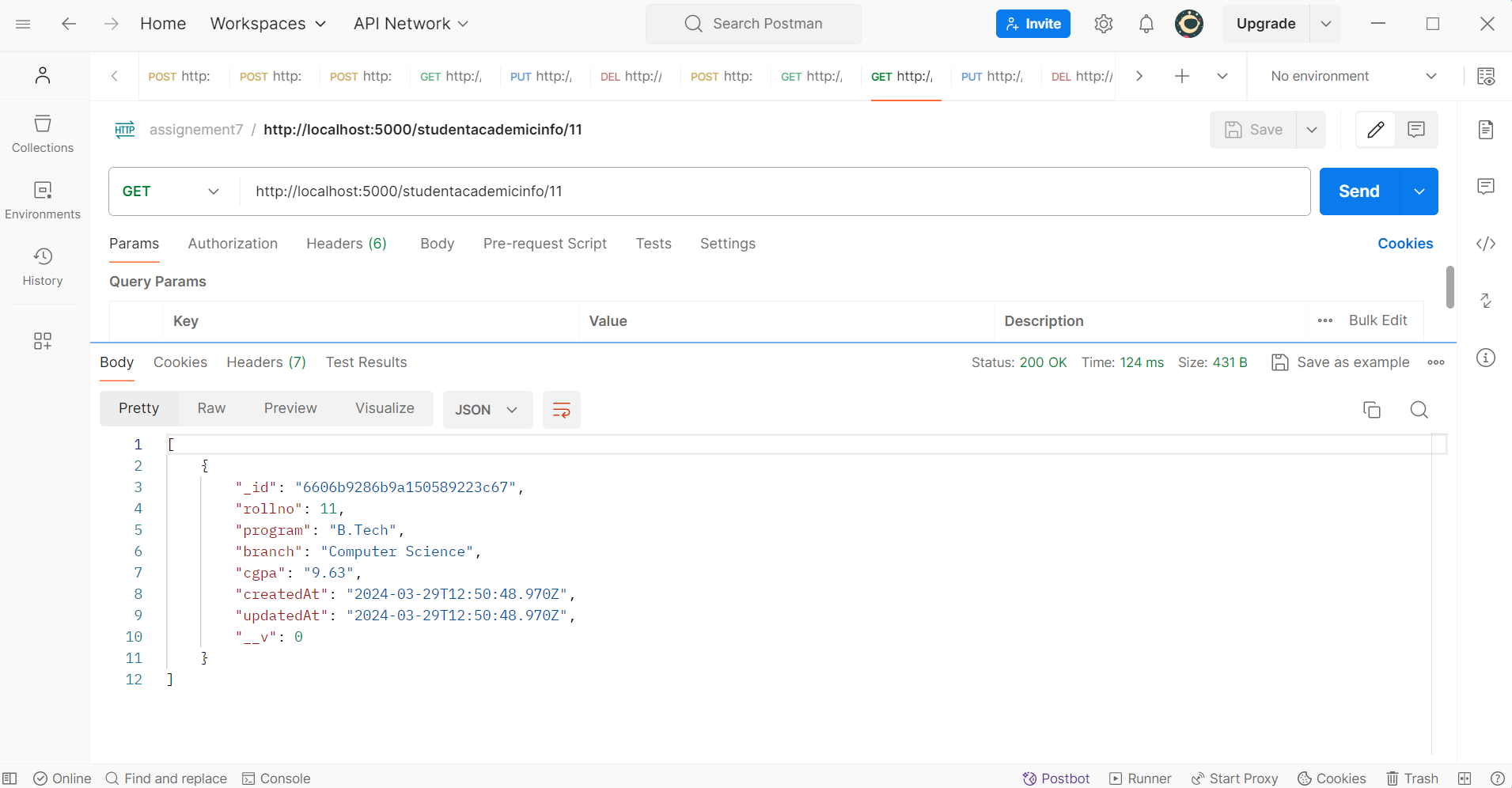
**DeleteStudentInfoByRoll**



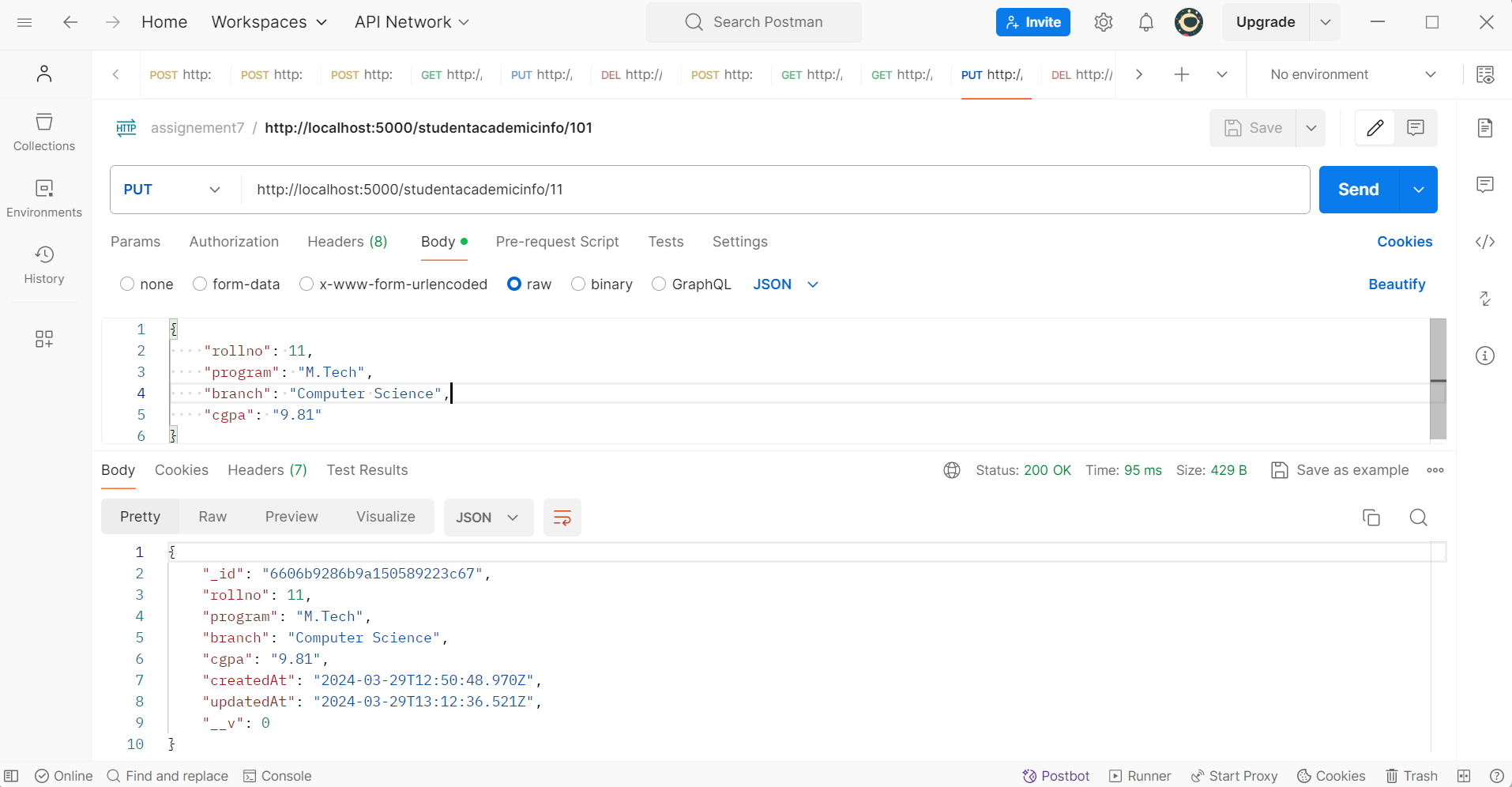
**AddStudentAcademicRecord**



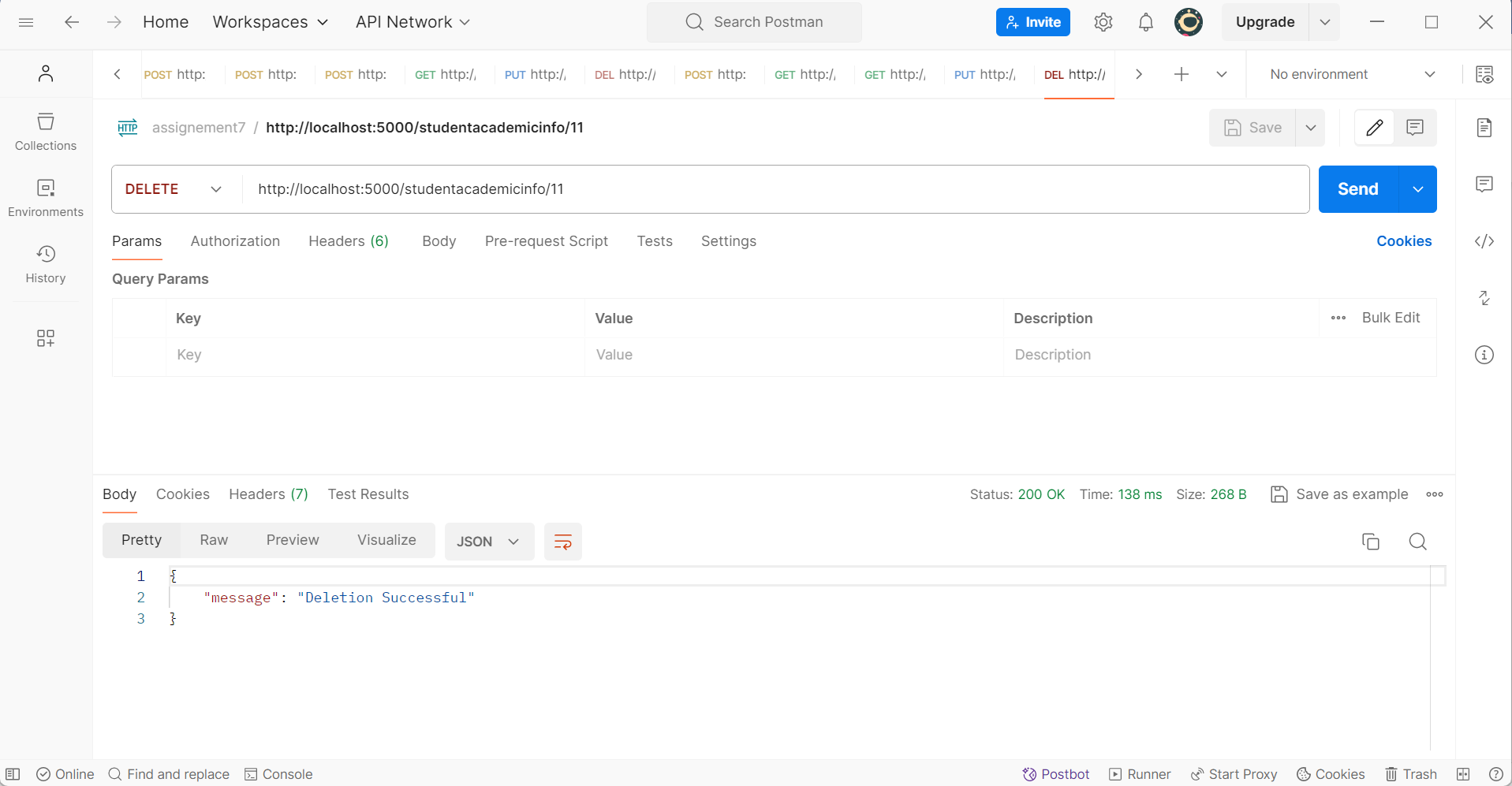
**GetStudentAcademicRecord**

****

**UpdateStudentAcademicRecord**

****

**DeleteStudentAcademicRecord**

****

**7. Add two more routes/end-points one of which will be capable of adding records to both the collections and another one will fetch the records from both the collections.**

**Ans-**

const getAllStudentDetails = async (req, res) => {

try {

const details = await StudentModel.aggregate([

{

$lookup: {

from: "studentacademicinfos",

localField: "rollno",

foreignField: "rollno",

as: "studentDetails",

},

},

]);

if (details) {

res.status(200).json(details);

} else {

res.status(404).json({ message: "No Data Found!" });

}

} catch (err) {

res.status(500).json(err);

}

};

const addAllStudentDetails = async (req, res) => {

try {

const { name, rollno, mobile, email, address, studentDetails } = req.body;

const [{program,branch,cgpa}] = studentDetails;

const studentInfo = await StudentModel.create({

name,

rollno,

mobile,

email,

address,

})

const studentAcademicInfo = await StudentAcademicModel.create({

rollno,

program,

branch,

cgpa,

})

res.status(200).json({message : "Data Inserted to both the collections!!"})

} catch (error) {

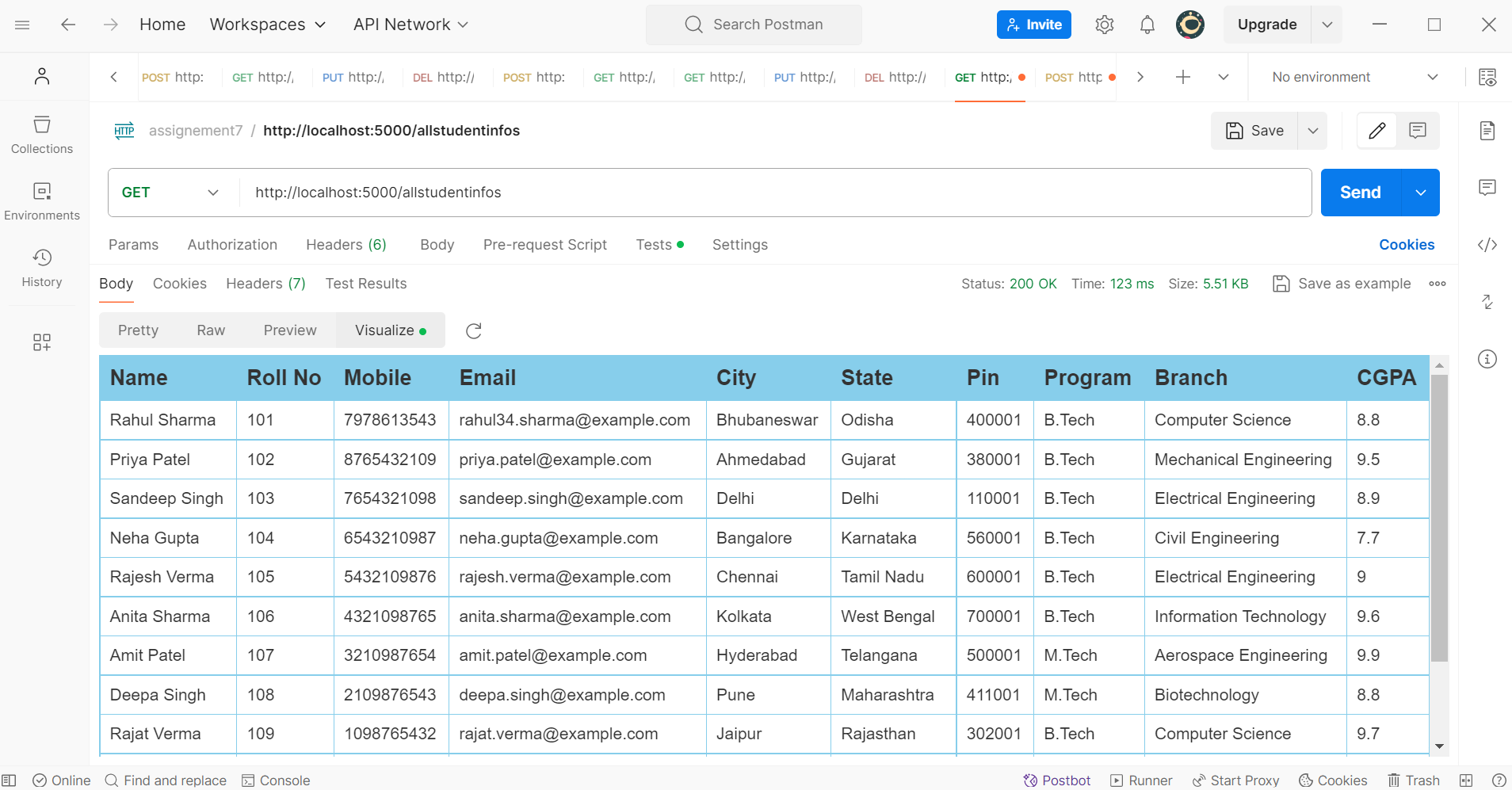
res.status(500).json({Error : "Error is : "+error})

}

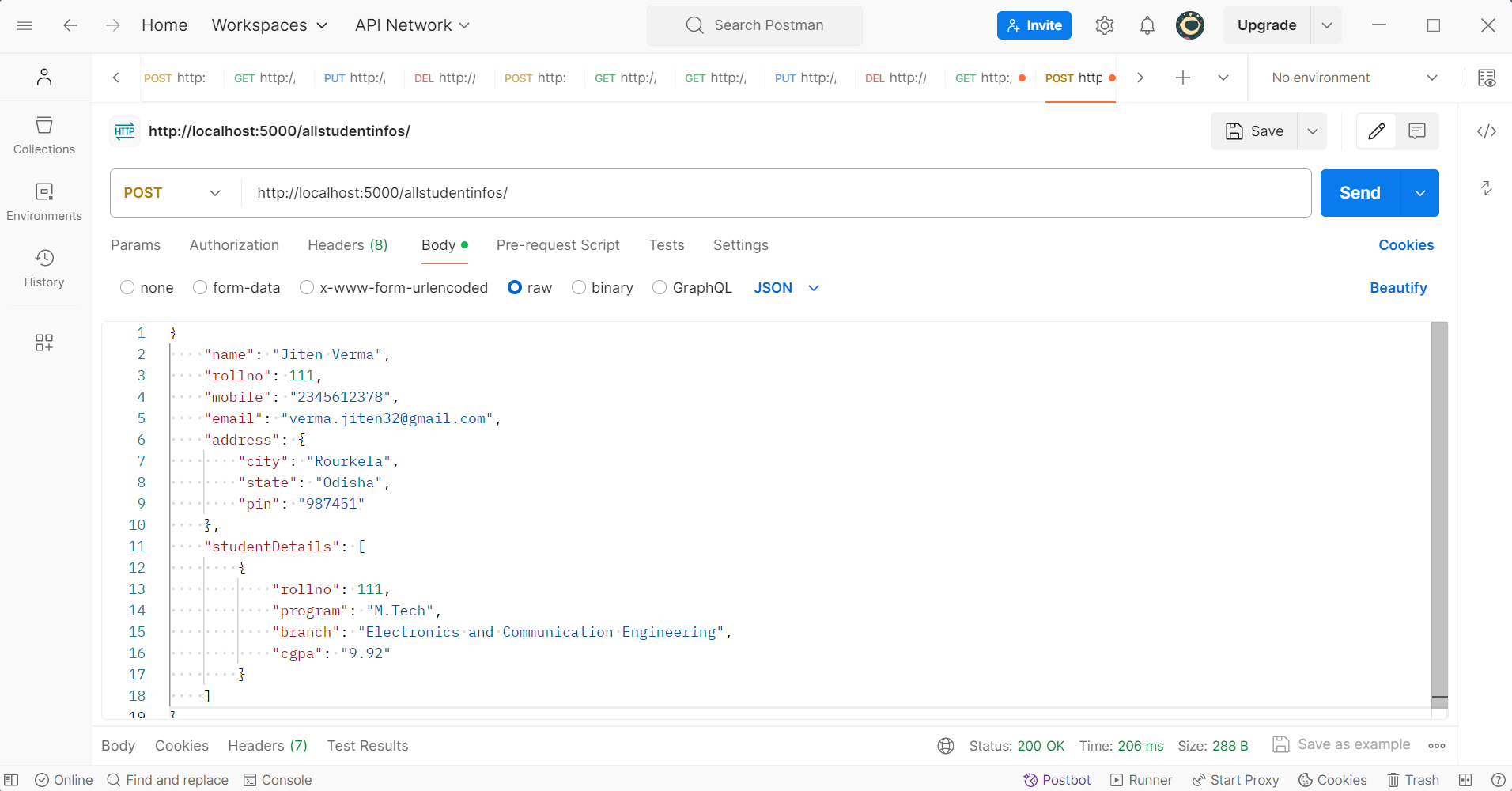
};

**Output:-**

**AllInfos**

****

**AddAllInfos**

****