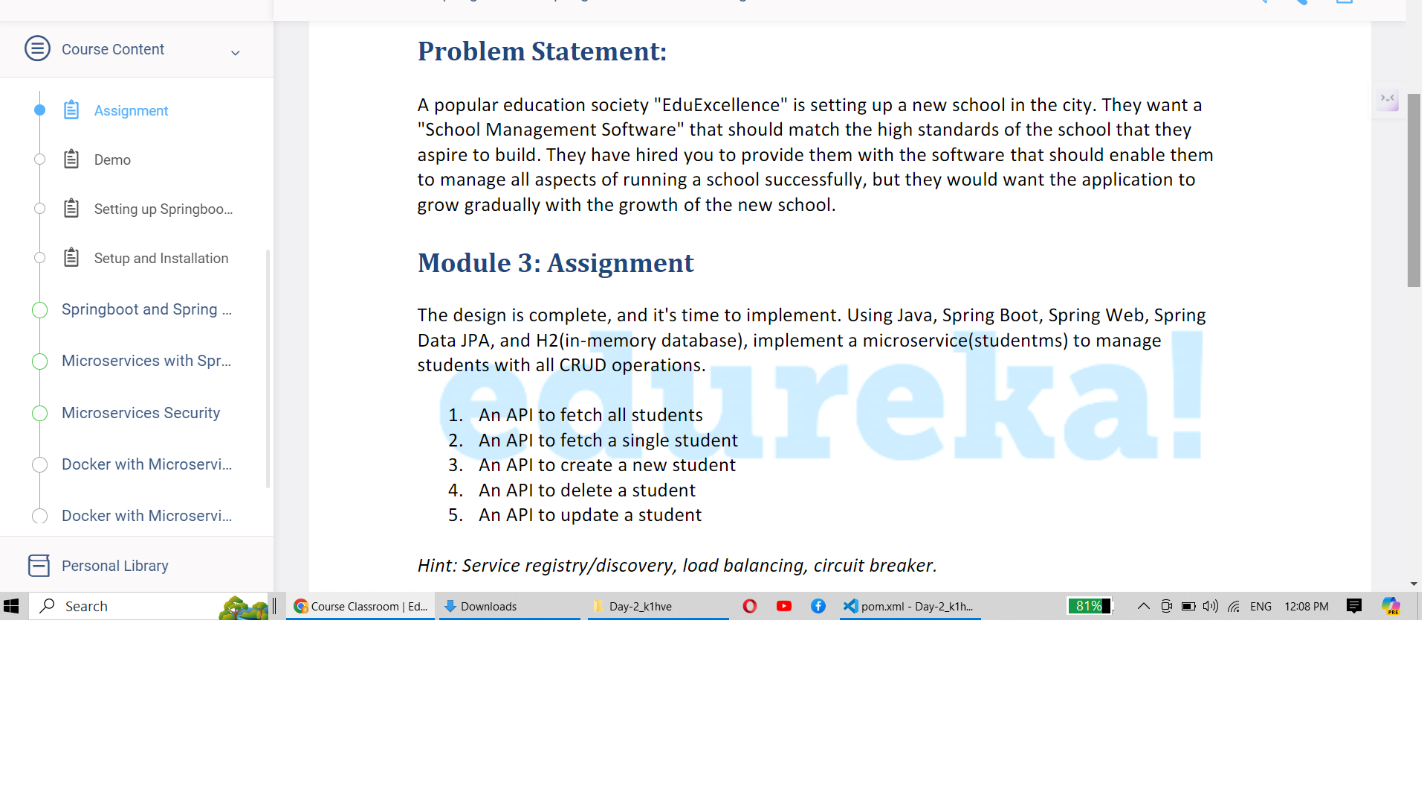
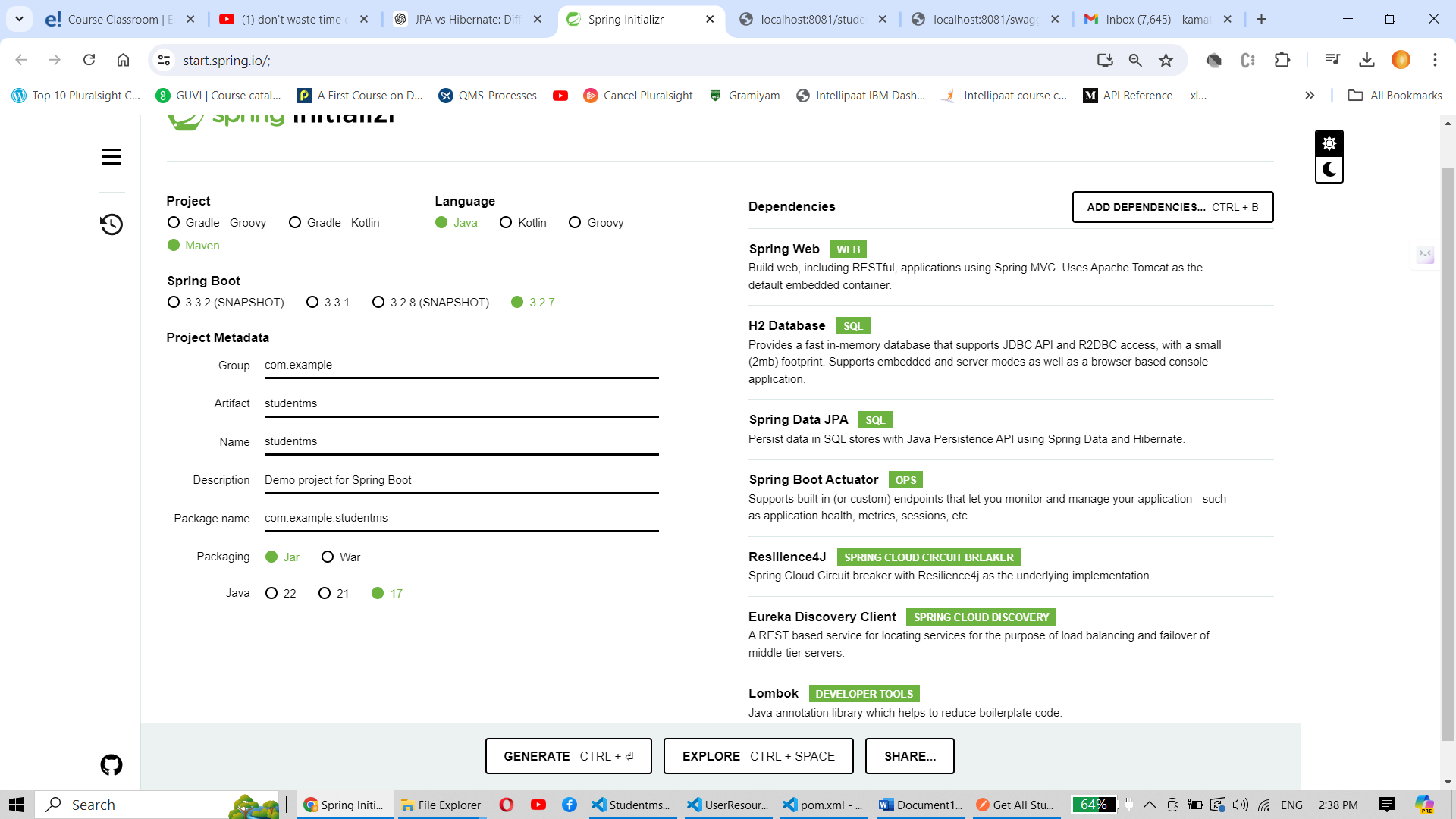
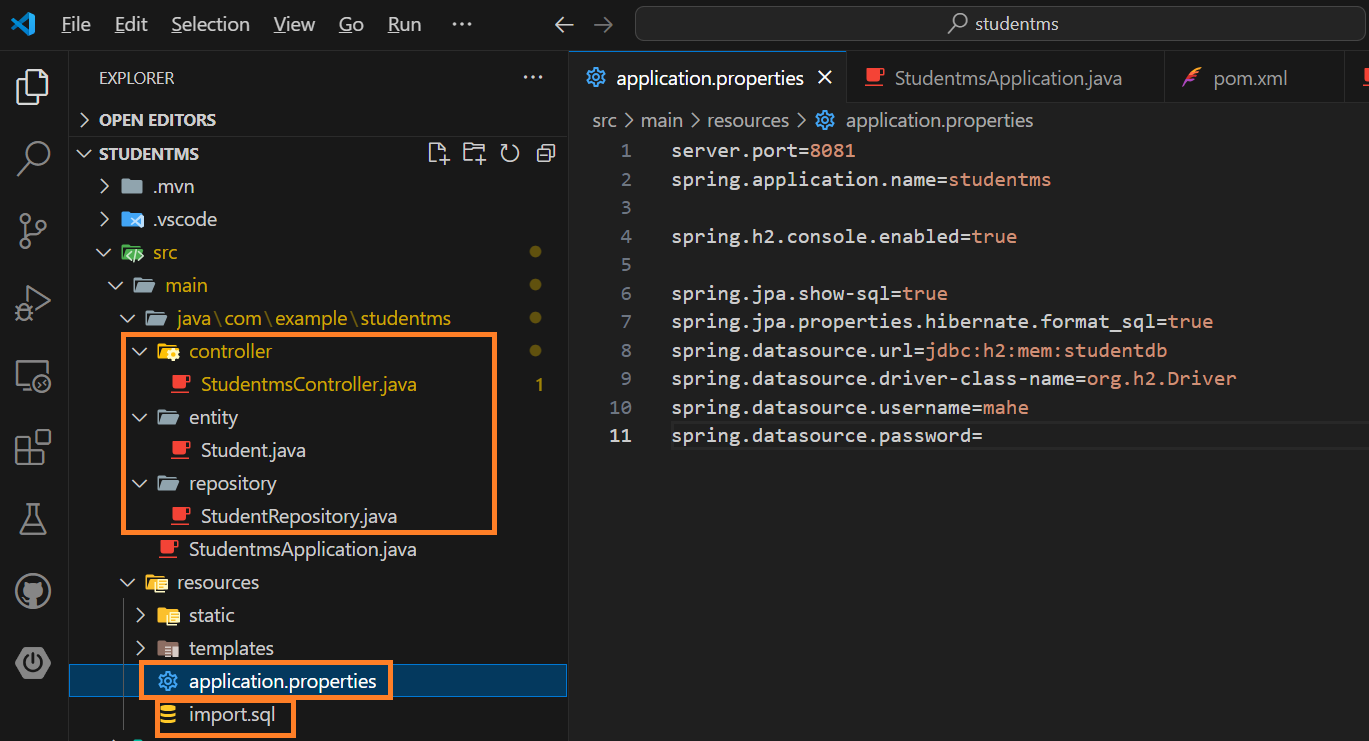
Assignment 3 

Step 1: Created a project studentms with the below mentioned dependencies. Considering dependencies for future assignments, added eureka client,resilience4j, actuator dependencies



2: In order to support H2 and to add other settings, updated application.properties.



3. The new files added were 1. StudentmsController.java 2.Student.java entity class 3.StudentRepository.java 4.import.sql

StudentmsController.java content

package com.example.studentms.controller;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

import com.example.studentms.entity.Student;

import com.example.studentms.repository.StudentRepository;

import jakarta.validation.constraints.Null;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

@RestController

public class StudentmsController {

    @Autowired

    StudentRepository studRepo;

    @GetMapping("/students") // An api to fetch all students

    public ResponseEntity<List<Student>> getStudents() {

        List<Student> students = studRepo.findAll();

        return new ResponseEntity<>(students, HttpStatus.OK);

    }

    @GetMapping("/student/{id}") //An api to fetch a single student

    public ResponseEntity<Student> getStudent(@PathVariable Integer id)

    {

        Optional<Student> st = studRepo.findById(id);

        return st.map(student -> new ResponseEntity<>(student, HttpStatus.OK))

            .orElseGet(() -> new ResponseEntity<>(HttpStatus.NOT\_FOUND));

    }

    @PostMapping("/student") //An api to create a new student

    public ResponseEntity<Student> addStudent(@RequestBody Student students) {

        Student savedStudent = studRepo.save(students);

        return new ResponseEntity<>(savedStudent, HttpStatus.OK);

    }

    @DeleteMapping("/student/{id}") //An api to delete a new student

    public ResponseEntity<Student> deleteStudent(@PathVariable Integer id) {

        Optional<Student> st = studRepo.findById(id);

        if (st.isPresent()) {

            studRepo.delete(st.get());

            return new ResponseEntity<>(HttpStatus.OK);

        } else {

            return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

        }

    }

    @PutMapping("/student") //An api to update a new student

    public ResponseEntity<Student> updateStudent(@RequestBody Student student) {

        Optional<Student> st = studRepo.findById(student.getId());

        if (st.isPresent()) {

            Student stud = st.get();

            stud.setName(student.getName());

            stud.setCourse(student.getCourse());

            stud.setBatch\_year(student.getBatch\_year());

            studRepo.save(stud);

            return new ResponseEntity<>(HttpStatus.OK);

        } else {

            return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

        }

    }

}

Student.java content

package com.example.studentms.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import lombok.Getter;

import lombok.Setter;

@Entity

@Table(name = "students\_data")

@Getter

@Setter

public class Student {

    @Id

    @GeneratedValue()

    Integer id;

    private String name;

    private String course;

    private Integer batch\_year;

}

StudentRepository.java content

package com.example.studentms.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.example.studentms.entity.Student;

@Repository

public interface StudentRepository extends JpaRepository<Student, Integer> {

}

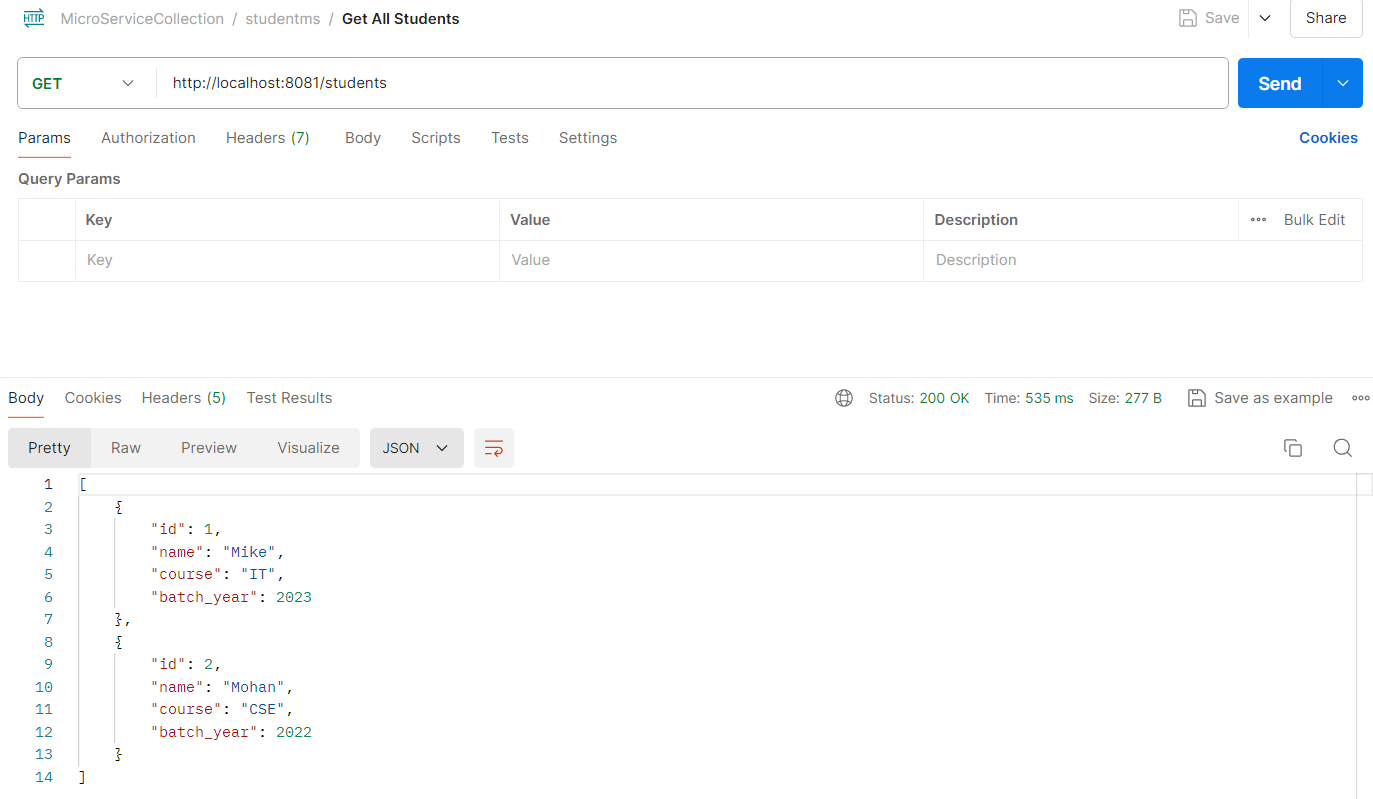
Import.sql content

insert into students\_data (id, name,course,batch\_year) values (1, 'Mike','IT', 2023);

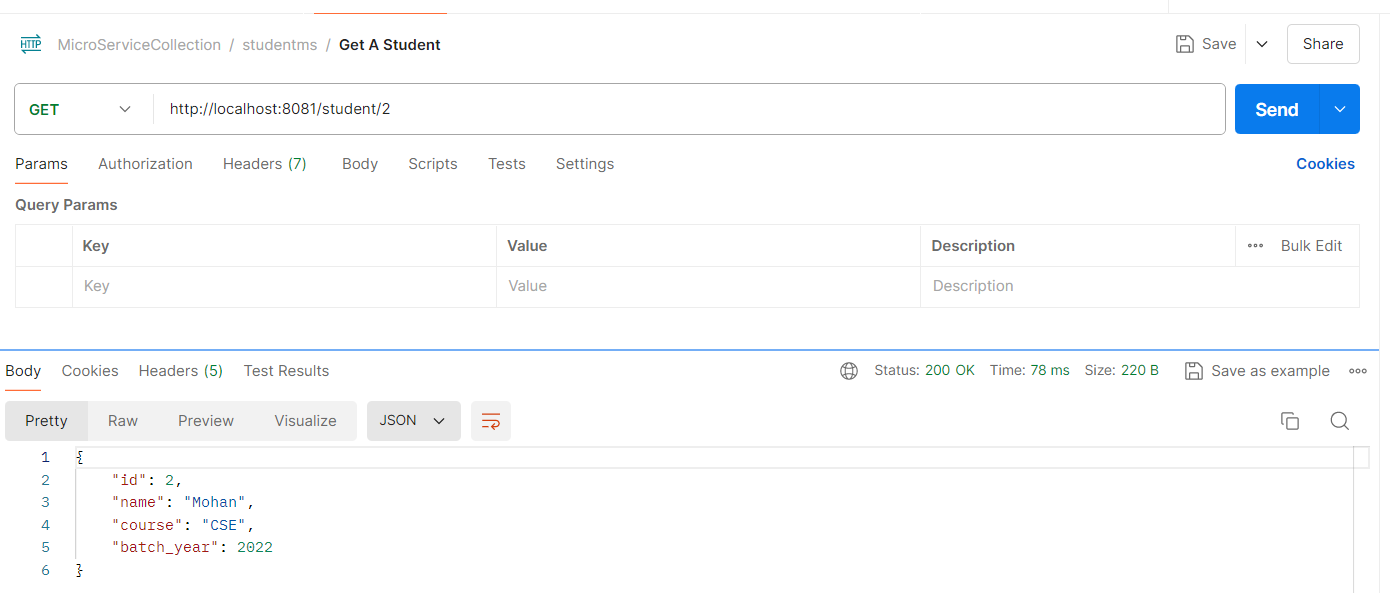
insert into students\_data (id, name,course,batch\_year) values (2, 'Mohan','CSE', 2022);

Step 4: Solution Execution Sequence

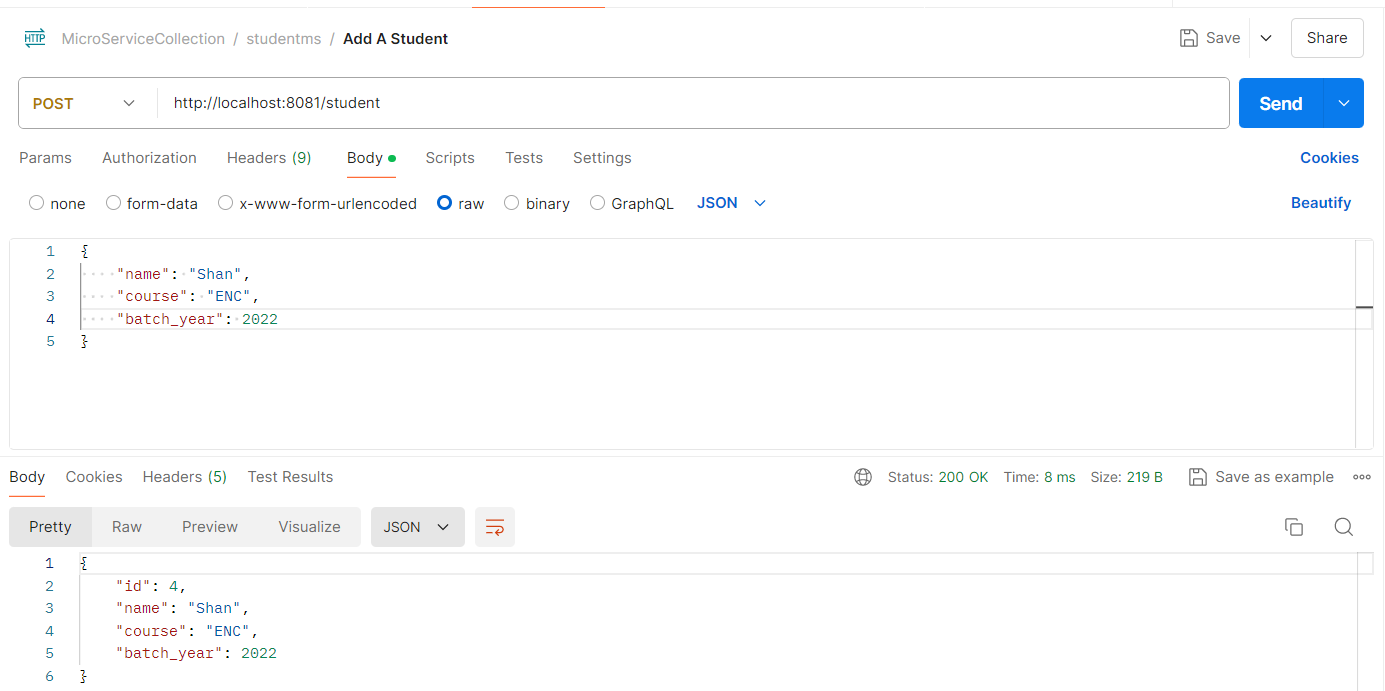
1. An API to fetch all students



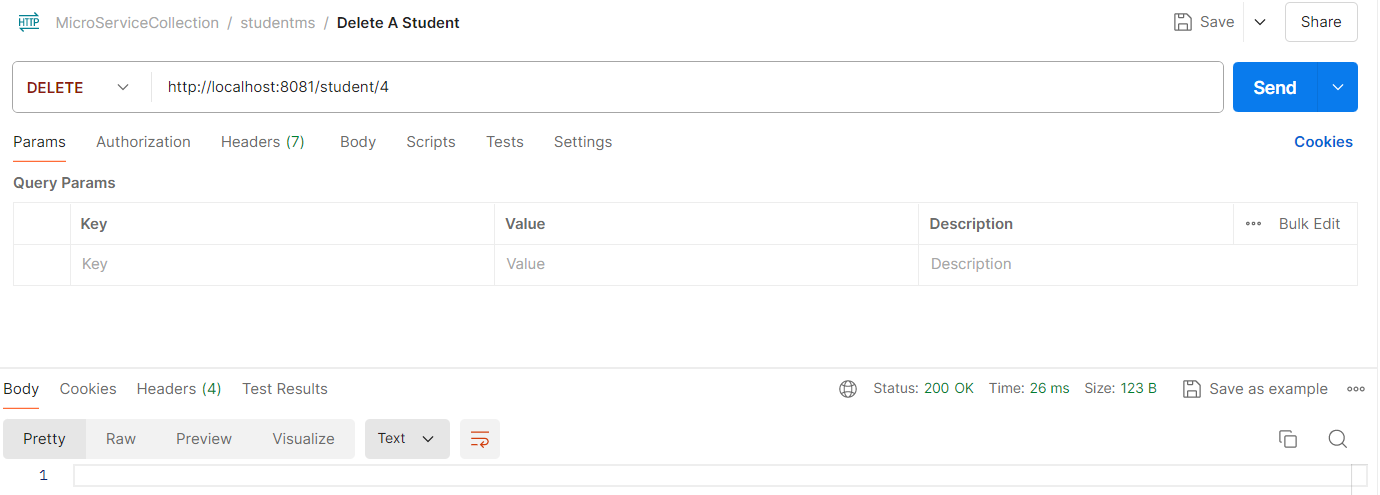
2. An API to fetch a single student



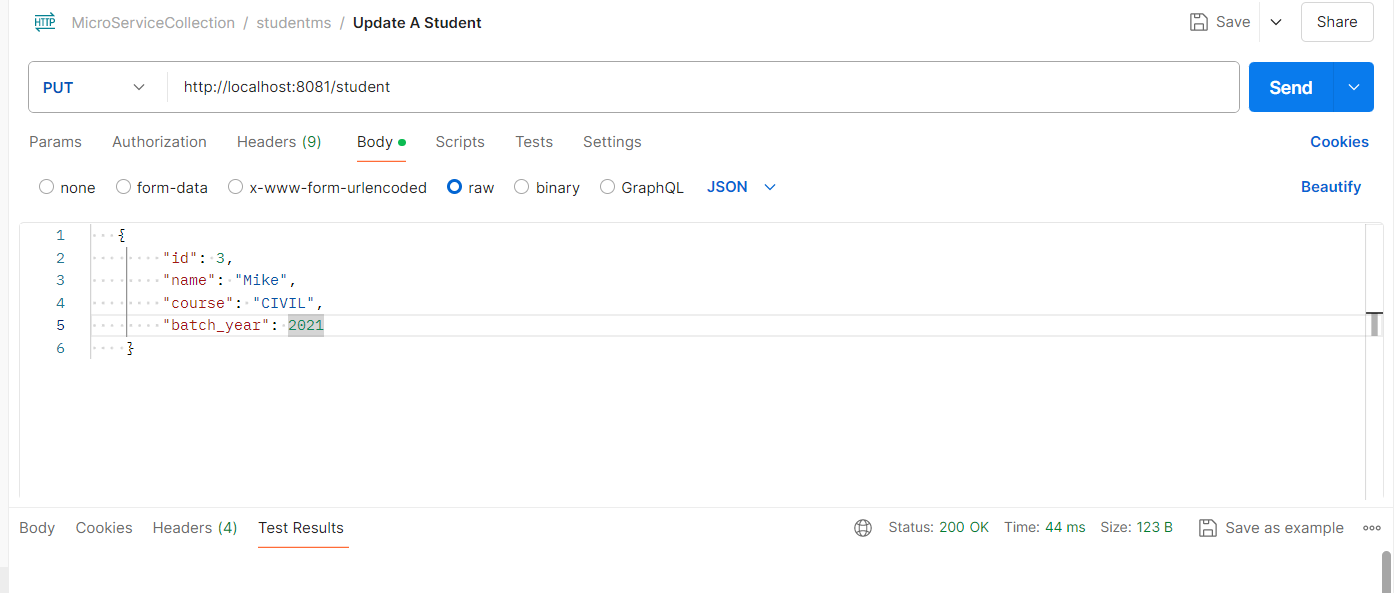
3. An API to create a new student



4. An API to delete a student



5. An API to update a student



Final list after addition, updation, deletion operations

