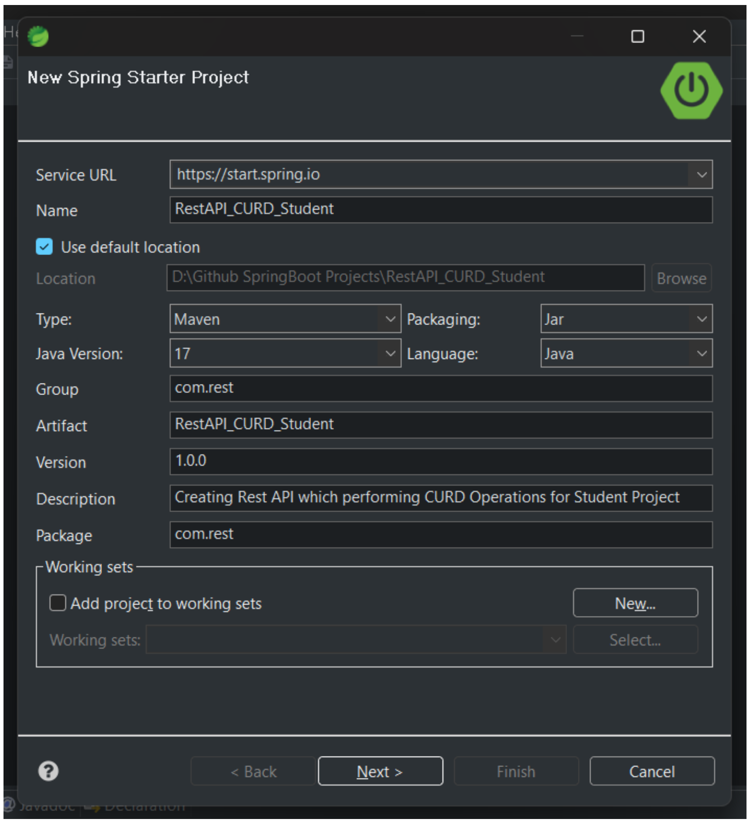
Rest API CURD Project -Student

1) Create new spring project of RestAPI_CURD_Student



New Spring Starter Project

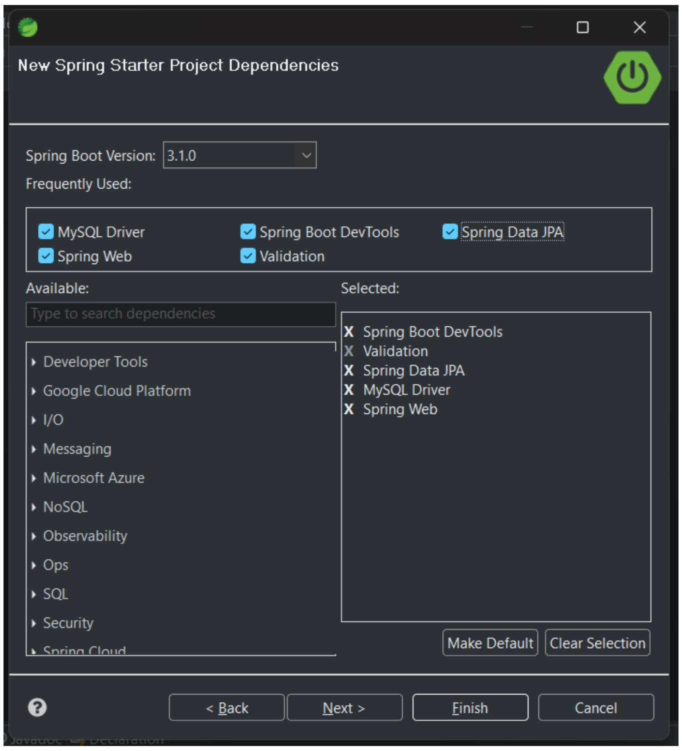
Service URL	https://start.spring.io
Name	RestAPI_CURD_Student
☑ Use default location	
Location	D:\Github SpringBoot Projects\RestAPI_CURD_Student
Type:	Maven
Packaging:	Jar
Java Version:	17
Language:	Java
Group	com.rest
Artifact	RestAPI_CURD_Student
Version	1.0.0
Description	Creating Rest API which performing CURD Operations for Student Project
Package	com.rest

Working sets

☐ Add project to working sets

Working sets:

2) Add 5 dependecies : devtools, spring web, validations, Mysql driver, spring JPA



New Spring Starter Project Dependencies

Spring Boot Version: 3.1.0

Frequently Used:

☑ MySQL Driver ☑ Spring Boot DevTools ☑ Spring Data JPA
☑ Spring Web ☑ Validation

Available:

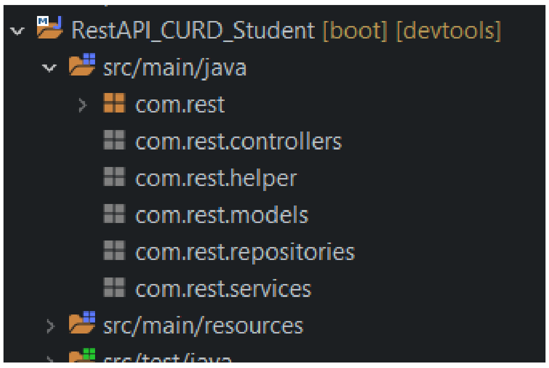
Type to search dependencies

- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops
- SQL
- Security
- Spring Cloud

Selected:

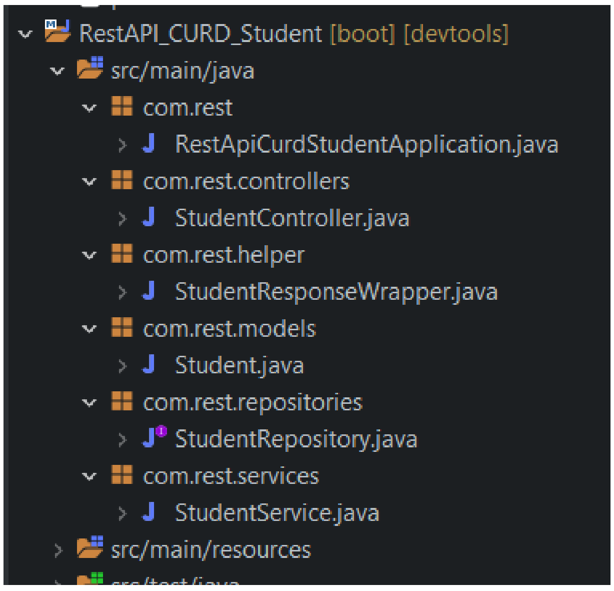
X Spring Boot DevTools
X Validation
X Spring Data JPA
X MySQL Driver
X Spring Web

3) Create 5 packages : models, repositories, services, helper, controllers



```
RestAPI_CURD_Student [boot] [devtools]
  src/main/java
    com.rest
    com.rest.controllers
    com.rest.helper
    com.rest.models
    com.rest.repositories
    com.rest.services
  src/main/resources
```

4) Create 4 classes and 1 interface in respective packages



```
RestAPI_CURD_Student [boot] [devtools]
  src/main/java
    com.rest
      RestApiCurdStudentApplication.java
    com.rest.controllers
      StudentController.java
    com.rest.helper
      StudentResponseWrapper.java
    com.rest.models
      Student.java
    com.rest.repositories
      StudentRepository.java
    com.rest.services
      StudentService.java
  src/main/resources
```

5) Create Student.java (model class) in models package

```
package com.rest.models;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.validation.constraints.Max;
import jakarta.validation.constraints.Min;

@Entity
public class Student {

@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int id;

@Column(nullable = false)
private String student_name;

@Column(nullable = false)
@Min(value = 6, message = "The student must be above 6 years of age.")
@Max(value = 28, message = "The student must be under 28 years of age.")
```

```
	private int age;

	@Column(nullable = false)
	private String city;

	@Column(nullable = false)
	private int total_marks;

	Student(){};

	// Contructor with all fields
	public Student(int id, String student_name, int age, String city, int total_marks) {
		super();
		this.id = id;
		this.student_name = student_name;
		this.age = age;
		this.city = city;
		this.total_marks = total_marks;
	}

	// Getter, Setters
	public int getId() {
		return id;
	}

	public void setId(int id) {
		this.id = id;
	}

	public String getStudent_name() {
		return student_name;
	}

	public void setStudent_name(String student_name) {
		this.student_name = student_name;
	}

	public int getAge() {
		return age;
	}

	public void setAge(int age) {
		this.age = age;
	}

	public String getCity() {
		return city;
	}

	public void setCity(String city) {
		this.city = city;
	}

	public int getTotal_marks() {
		return total_marks;
	}

	public void setTotal_marks(int total_marks) {
		this.total_marks = total_marks;
	}
}
```

6) Create StudentRepository interface in repositories package

```
	package com.rest.repositories;

	import org.springframework.data.repository.CrudRepository;
	import com.rest.models.Student;

	public interface StudentRepository extends CrudRepository<Student, Integer>{ }
```

7) Create StudentService.java in services package

```
package com.rest.services;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Service;
import org.springframework.web.server.ResponseStatusException;
import com.rest.models.Student;
import com.rest.repositories.StudentRepository;

@Service
public class StudentService {

        @Autowired
        StudentRepository studentRepository;

        // Get All Students
        public Iterable<Student> getAllStudents()
        {
                return studentRepository.findAll();
        }

        // Create Student
        public Student createStudent(Student student)
        {
                Student inserted_student= studentRepository.save(student);
                return inserted_student;
        }

        // Get Student by Id
        public Student getStudentById(int id)
        {
                Student founded_student= studentRepository.findById(id).orElseThrow ( ()-> {

                        throw new ResponseStatusException(HttpStatus.NOT_FOUND, "There is no student
of this id");
                        });
                return founded_student;
        }

        // Update Student by Id
        public Student updateStudentById(int id, Student student)
        {
                Student founded_student= getStudentById(id);
                student.setId(founded_student.getId());
                Student updated_student = studentRepository.save(student);
                return updated_student;
        }

        // Delete Student by Id
        public void deleteStudentById(int id)
        {
                getStudentById(id);
                studentRepository.deleteById(id);
        }
}
```

8) Prepare StudentResponseWrapper.java to wrap responses

```
package com.rest.helper;

public class StudentResponseWrapper {

        String message;
        Object data;

        public String getMessage() {
                return message;
```

```
		}
		public void setMessage(String message) {
			this.message = message;
		}
		public Object getData() {
			return data;
		}
		public void setData(Object data) {
			this.data = data;
		}
	}
```

9) Create REST API to perform all CRUD operations — StudentController.java

```
package com.rest.controllers;

import java.util.Iterator;
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.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.server.ResponseStatusException;
import com.rest.models.Student;
import com.rest.services.StudentService;
import com.rest.helper.StudentResponseWrapper;
import jakarta.validation.Valid;

@RestController
@RequestMapping("/students")
public class StudentController {

	@Autowired
	StudentService studentService;

	@GetMapping("")
	public ResponseEntity<?> getAllStudents()
	{
		Iterable<Student> student= studentService.getAllStudents();
		Iterator<Student>all_student=student.iterator();

		if(all_student.hasNext())
			{
			StudentResponseWrapper srw = new StudentResponseWrapper();
			srw.setMessage("Student data found successfully");
			srw.setData(all_student);
			return new ResponseEntity<>(srw,HttpStatus.FOUND);
			}
		else
			{
			throw new  ResponseStatusException(HttpStatus.NOT_FOUND, "There is no student data available. Please 
add some.");
			}
	}

	@GetMapping("/{id}")
	public ResponseEntity<?> getStudentById(@PathVariable int id)
	{
		Student founded_student=studentService.getStudentById(id);
		StudentResponseWrapper srw = new StudentResponseWrapper();
		srw.setMessage("Student data found successfully");
		srw.setData(founded_student);
		return new ResponseEntity<>(srw,HttpStatus.FOUND);
	}
```

```
    @PostMapping("")
    public ResponseEntity<?> creatstudent(@RequestBody @Valid Student student)
    {
            Student created_student= studentService.createStudent(student);
            StudentResponseWrapper srw= new StudentResponseWrapper();
            srw.setMessage("Student data created succesfully");
            srw.setData(created_student);
            return new ResponseEntity<>(srw,HttpStatus.OK);
    }

    @PutMapping("/{id}")
    public ResponseEntity<?> updateStudentById(@PathVariable int id, @RequestBody @Valid Student student)
    {
            Student updated_student=studentService.updateStudentById(id, student);
            StudentResponseWrapper srw = new StudentResponseWrapper();
            srw.setMessage("Student data updated by Id successfully");
            srw.setData(updated_student);
            return new ResponseEntity<>(srw,HttpStatus.FOUND);

    }

    @DeleteMapping("/{id}")
    public ResponseEntity<?> deleteStudentById(@PathVariable int id)
    {
            studentService.deleteStudentById(id);
            StudentResponseWrapper srw = new StudentResponseWrapper();
            srw.setMessage("Student data deleted by Id successfully");
            return new ResponseEntity<>(srw,HttpStatus.OK);
    }
}
```

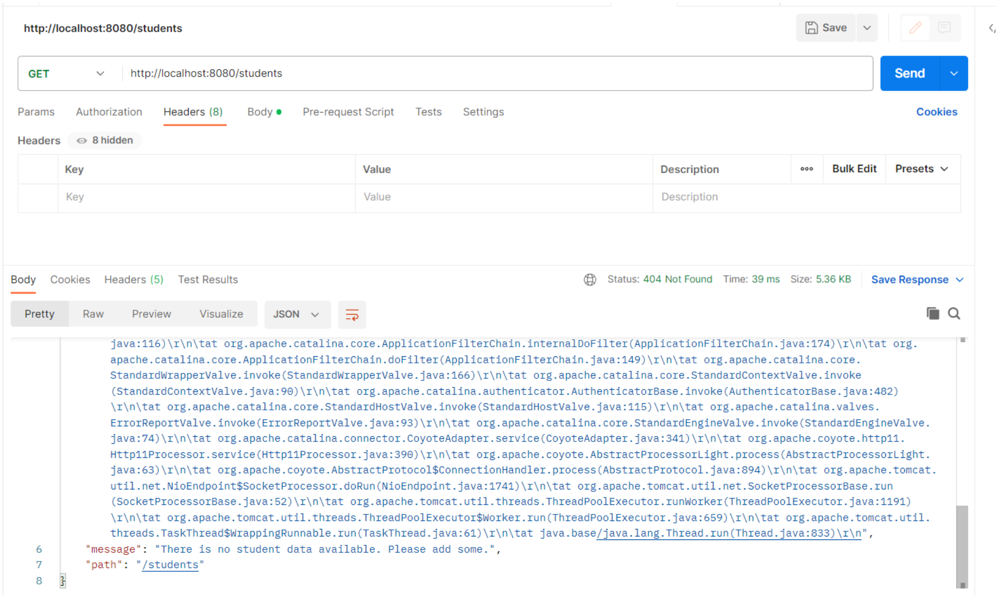
10) Setting application.properties to use MYSQL database

```
spring.datasource.url=jdbc:mysql://localhost:3306/student_db
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
```

11) Run project and test for all APIs (I have tested all APIs on postman)

OUTPUTS from POSTMAN :

1) **GET** before inserting data (throwing associated error message)



http://localhost:8080/students

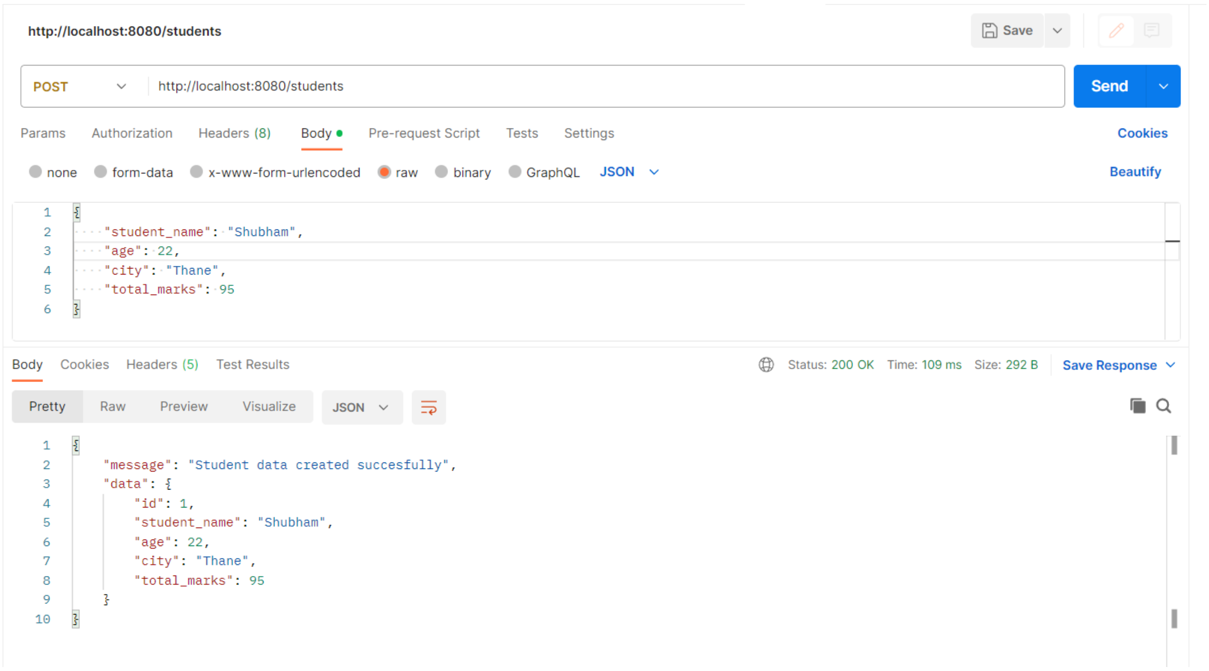
GET http://localhost:8080/students

Key	Value	Description
Key	Value	Description

Status: 404 Not Found Time: 39 ms Size: 5.36 KB

```
        java:116)\r\n\tat org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:174)\r\n\tat org.
        apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:149)\r\n\tat org.apache.catalina.core.
        StandardWrapperValve.invoke(StandardWrapperValve.java:166)\r\n\tat org.apache.catalina.core.StandardContextValve.invoke
        (StandardContextValve.java:90)\r\n\tat org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:482)
        \r\n\tat org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:115)\r\n\tat org.apache.catalina.valves.
        ErrorReportValve.invoke(ErrorReportValve.java:93)\r\n\tat org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.
        java:74)\r\n\tat org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:341)\r\n\tat org.apache.coyote.http11.
        Http11Processor.service(Http11Processor.java:390)\r\n\tat org.apache.coyote.AbstractProcessorLight.process(AbstractProcessorLight.
        java:63)\r\n\tat org.apache.coyote.AbstractProtocol$ConnectionHandler.process(AbstractProtocol.java:894)\r\n\tat org.apache.tomcat.
        util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1741)\r\n\tat org.apache.tomcat.util.net.SocketProcessorBase.run
        (SocketProcessorBase.java:52)\r\n\tat org.apache.tomcat.util.threads.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1191)
        \r\n\tat org.apache.tomcat.util.threads.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:659)\r\n\tat org.apache.tomcat.util.
        threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)\r\n\tat java.base/java.lang.Thread.run(Thread.java:833)\r\n",
    "message": "There is no student data available. Please add some.",
    "path": "/students"
}
```

2) **POST** / Insert



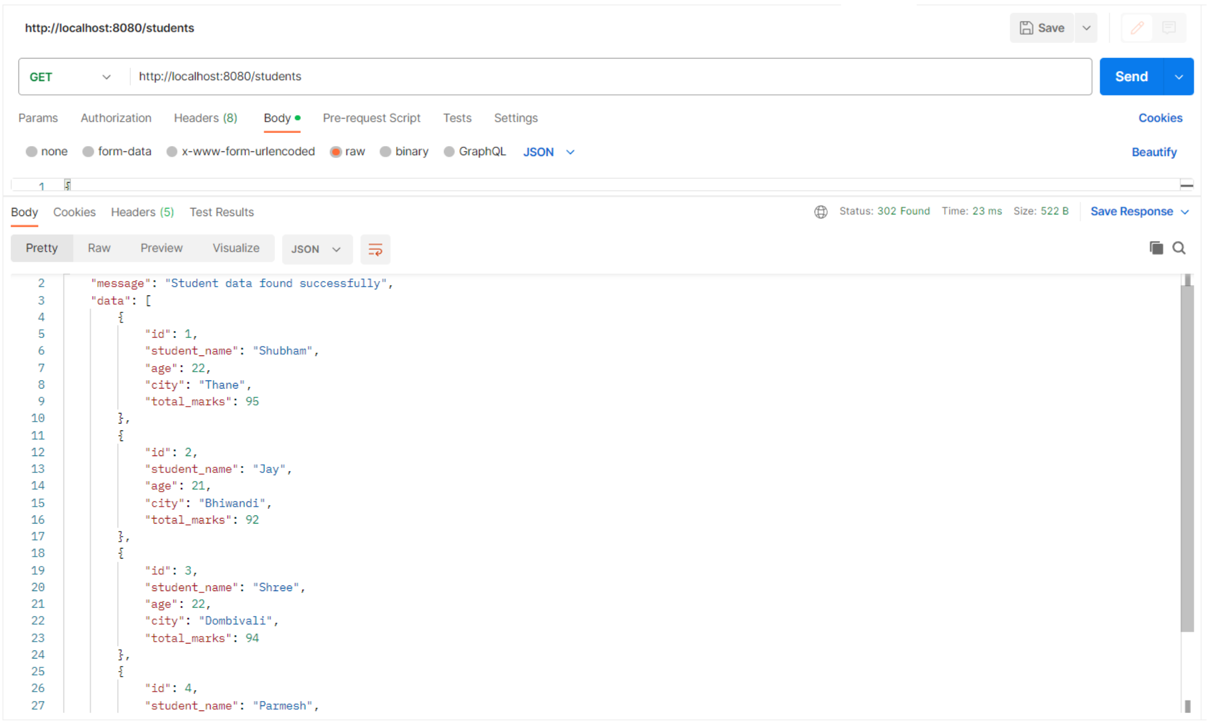
POST http://localhost:8080/students

```
{
    "student_name": "Shubham",
    "age": 22,
    "city": "Thane",
    "total_marks": 95
}
```

Status: 200 OK Time: 109 ms Size: 292 B

```
{
    "message": "Student data created succesfully",
    "data": {
        "id": 1,
        "student_name": "Shubham",
        "age": 22,
        "city": "Thane",
        "total_marks": 95
    }
}
```

3) **Get** after inserting data

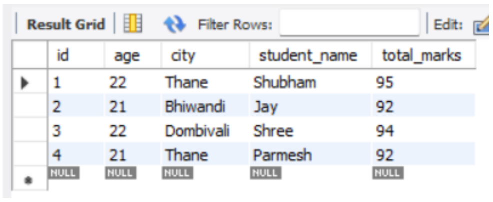


GET http://localhost:8080/students

Status: 302 Found Time: 23 ms Size: 522 B

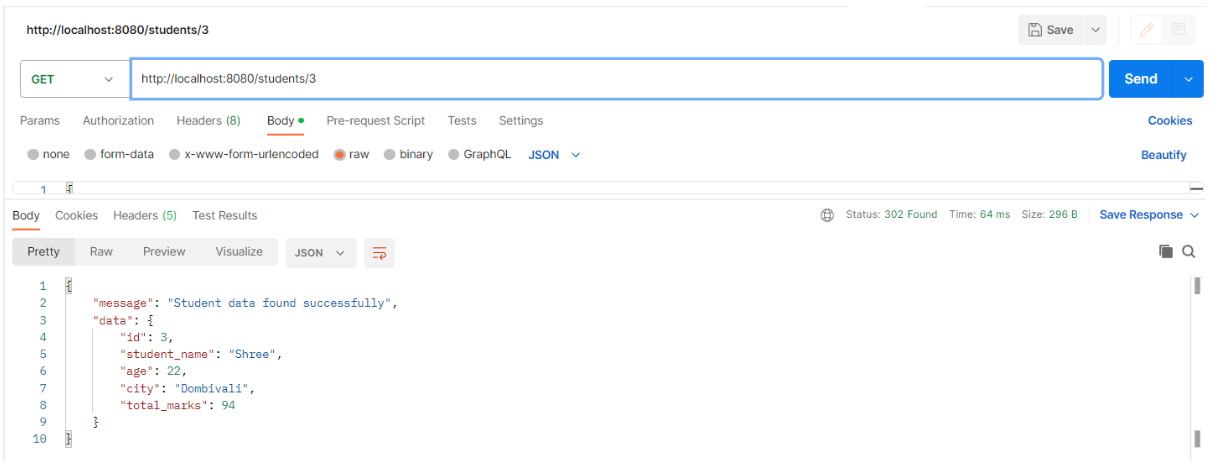
```
    "message": "Student data found successfully",
    "data": [
        {
            "id": 1,
            "student_name": "Shubham",
            "age": 22,
            "city": "Thane",
            "total_marks": 95
        },
        {
            "id": 2,
            "student_name": "Jay",
            "age": 21,
            "city": "Bhiwandi",
            "total_marks": 92
        },
        {
            "id": 3,
            "student_name": "Shree",
            "age": 22,
            "city": "Dombivali",
            "total_marks": 94
        },
        {
            "id": 4,
            "student_name": "Parmesh",
```

Data in student table (MySQL Workbench) :



id	age	city	student_name	total_marks
1	22	Thane	Shubham	95
2	21	Bhiwandi	Jay	92
3	22	Dombivali	Shree	94
4	21	Thane	Parmesh	92
NULL	NULL	NULL	NULL	NULL

4) **Get** by Id



http://localhost:8080/students/3

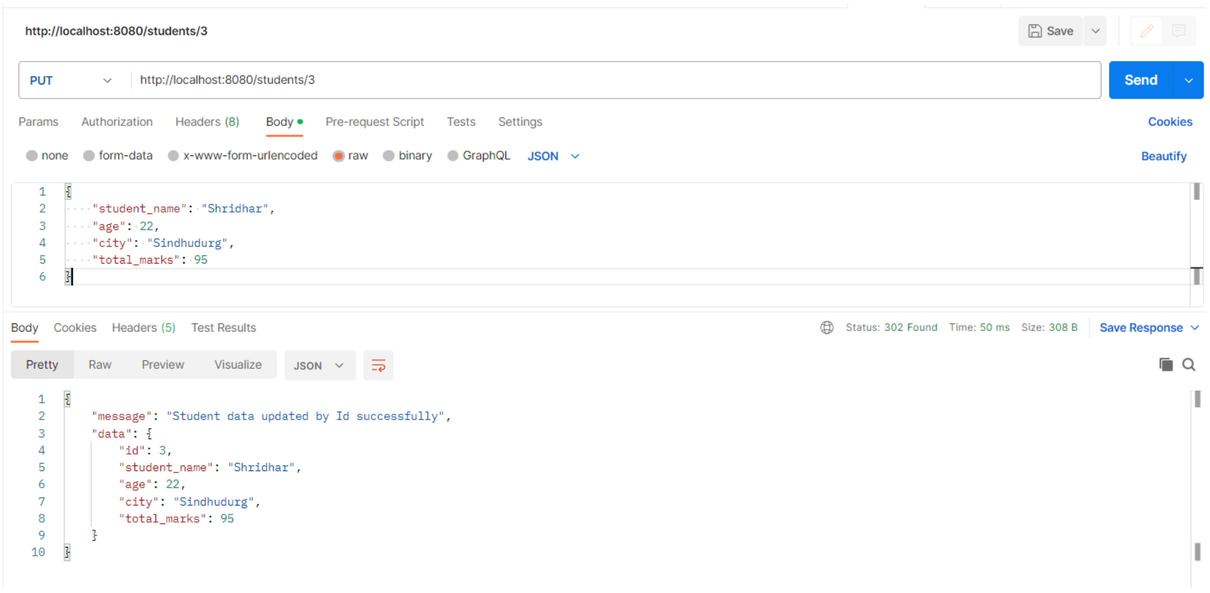
GET http://localhost:8080/students/3

Body: raw JSON

Status: 302 Found Time: 64 ms Size: 296 B

```
{
    "message": "Student data found successfully",
    "data": {
        "id": 3,
        "student_name": "Shree",
        "age": 22,
        "city": "Dombivali",
        "total_marks": 94
    }
}
```

5) **Update** by ID (PUT)



http://localhost:8080/students/3

PUT http://localhost:8080/students/3

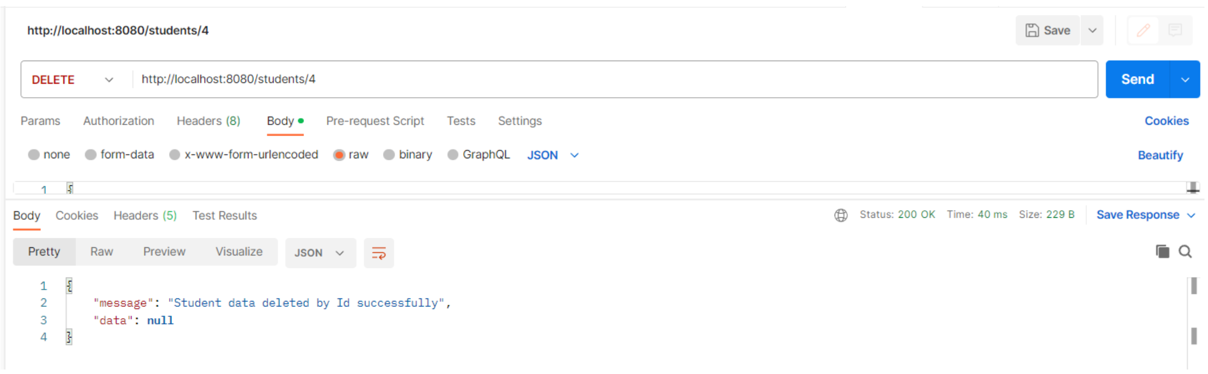
Body: raw JSON

```
{
    "student_name": "Shridhar",
    "age": 22,
    "city": "Sindhudurg",
    "total_marks": 95
}
```

Status: 302 Found Time: 50 ms Size: 308 B

```
{
    "message": "Student data updated by Id successfully",
    "data": {
        "id": 3,
        "student_name": "Shridhar",
        "age": 22,
        "city": "Sindhudurg",
        "total_marks": 95
    }
}
```

6) **Delete** by Id



http://localhost:8080/students/4

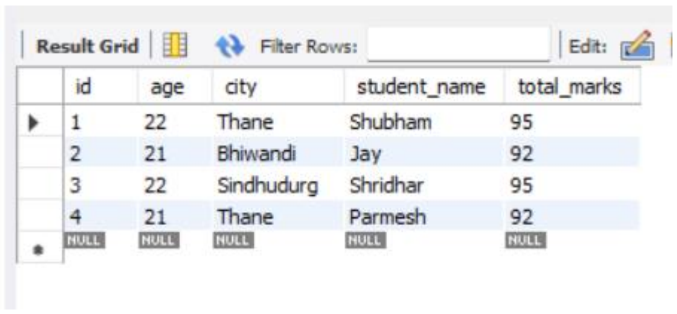
DELETE http://localhost:8080/students/4

Body: raw JSON

Status: 200 OK Time: 40 ms Size: 229 B

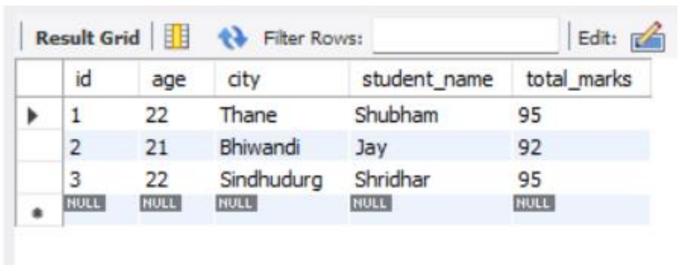
```
{
    "message": "Student data deleted by Id successfully",
    "data": null
}
```

Table data before deleting student data :



id	age	city	student_name	total_marks
1	22	Thane	Shubham	95
2	21	Bhiwandi	Jay	92
3	22	Sindhudurg	Shridhar	95
4	21	Thane	Parmesh	92
NULL	NULL	NULL	NULL	NULL

Table data after deleting student data :



id	age	city	student_name	total_marks
1	22	Thane	Shubham	95
2	21	Bhiwandi	Jay	92
3	22	Sindhudurg	Shridhar	95
NULL	NULL	NULL	NULL	NULL