Task 1, REST API in Java – by Nikhil Kumar

This is a simple REST API built with Spring Boot that can handle basic CRUD, Create, Read, Update, Delete requests. The API allows user to interact with a student database by providing endpoints for creating, reading, deleting, and updating products.

Response screenshots attached at the end.

Postman workspace: https://www.postman.com/supply-operator-97850319/workspace/nikhilkumar-rest-api

Getting Started

To run this project, you need:

Java 11+

Maven 3.6+

Postman or any other REST client

Installation

git clone https://github.com/NikhilKumar2444/Kaiburr-Task-1.git

mvn clean install

mvn package java -jar target/crud-0.0.1-SNAPSHOT.jar

API Endpoints

Get:

Get("/students") – Outputs all the entries in the database

Get("/students/{ID}") – Outputs the student with the ID provided in the parameter
Get("/students/name/{Name}" – Outputs all the entries where the fullname matches Name given in parameter.

Post:

Post("/students/register") – This endpoint is used to create a new entry. The json object obtained here is then stored in the database if the ID is not used previously. If the ID is used previously, a METHOD_NOT_ALLOWED (405) status is returned.

Delete:

Delete("/students/delete/{ID}") – This endpoint is used to remove the entry based on the ID that we receive. The API checks if we have an entry with the given ID and deletes it if the entry is found. If no entry with such ID is found we return status 404.

PUT:

Put("/students/edit/{ID}") – This endpoint receives a JSON object along with a parameter ID. The parameter ID is used to locate the entry that needs editing and the JSON file is then edited.

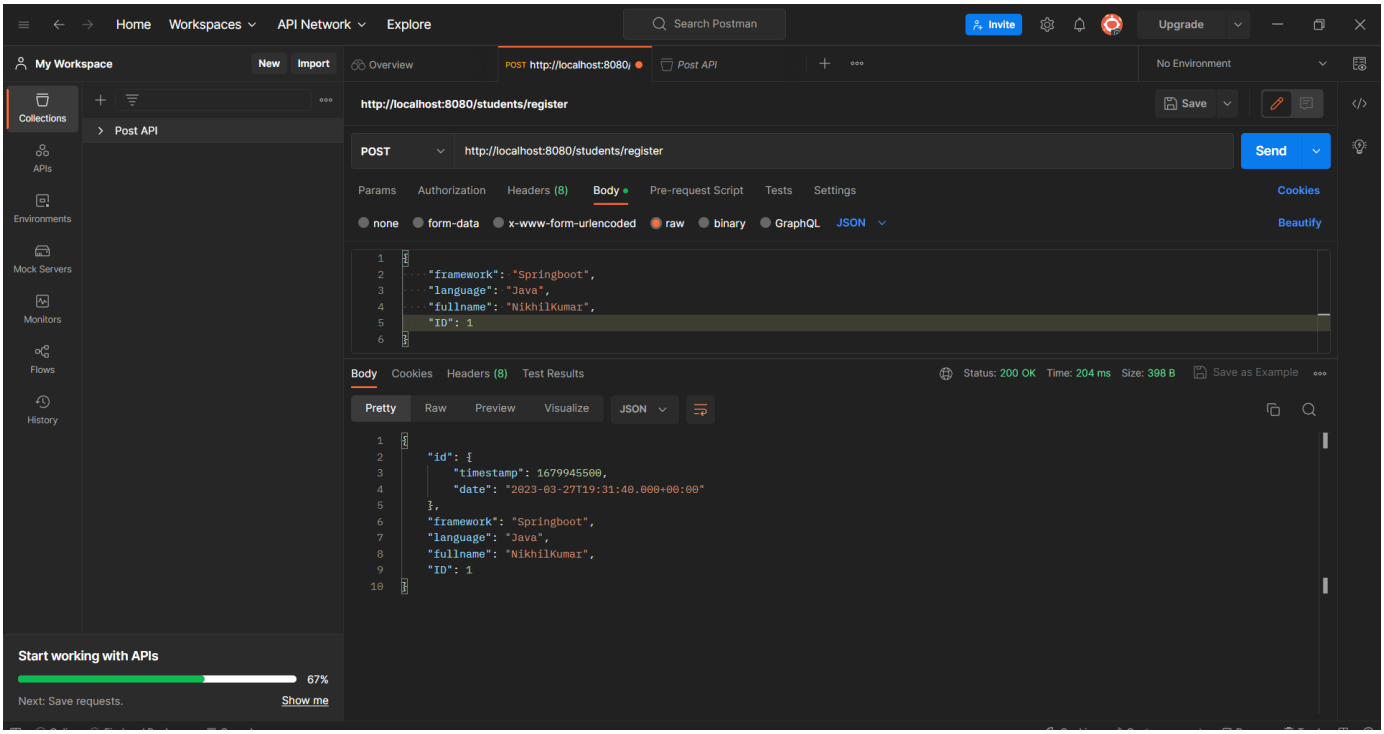
Output and screenshots

POST:

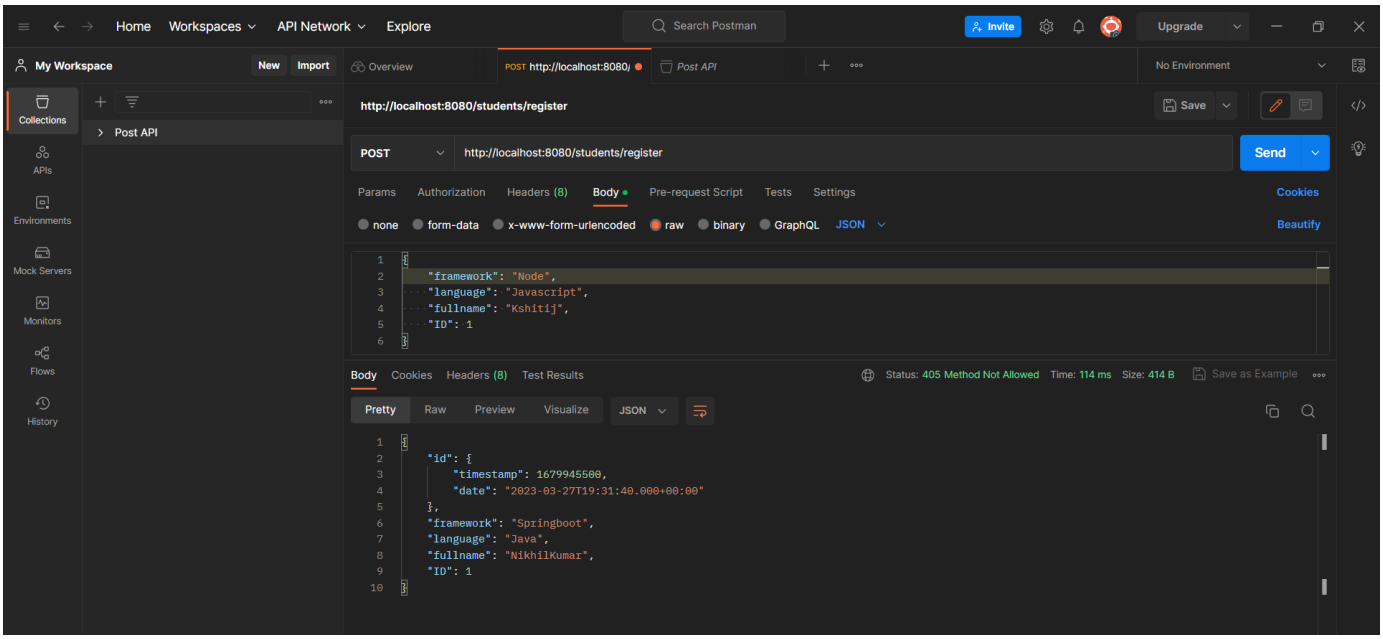
Endpoint: "/students/register"

Requestbody contains a student object (in json format).

Creating a fresh student:



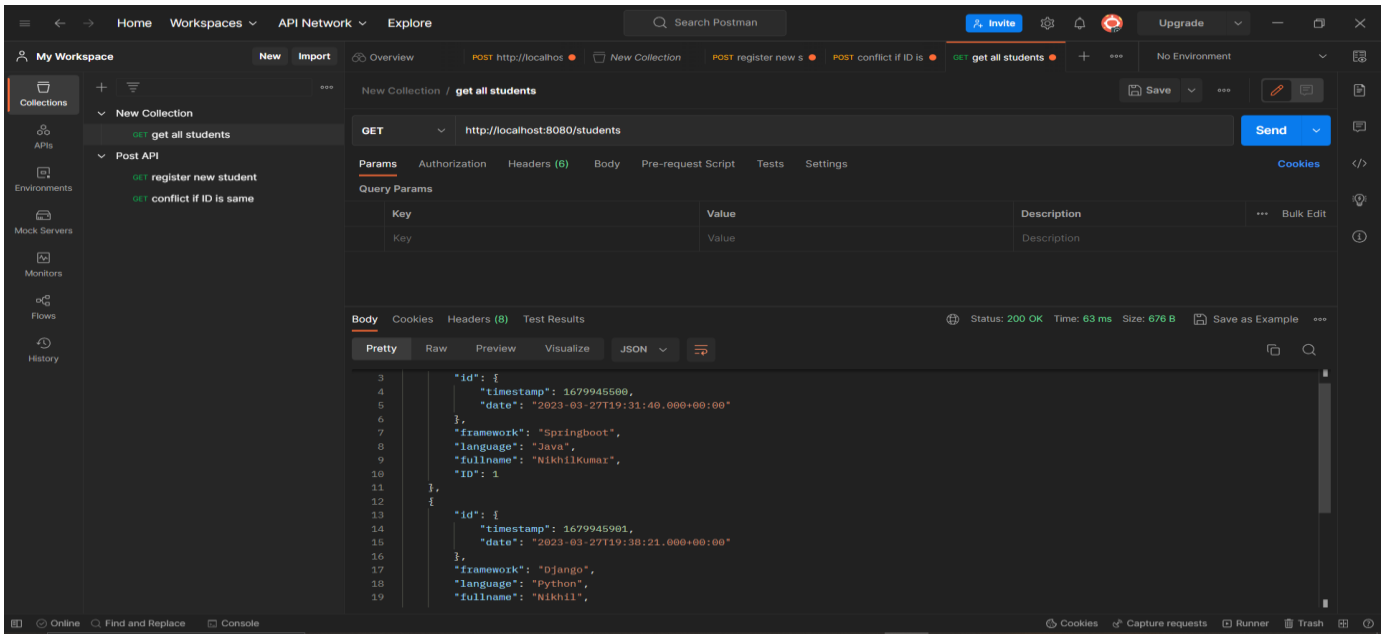
If we try to create another entry with same ID, the server responds with a status of 405 and we are returned the entry that is already present with that ID:



GET:

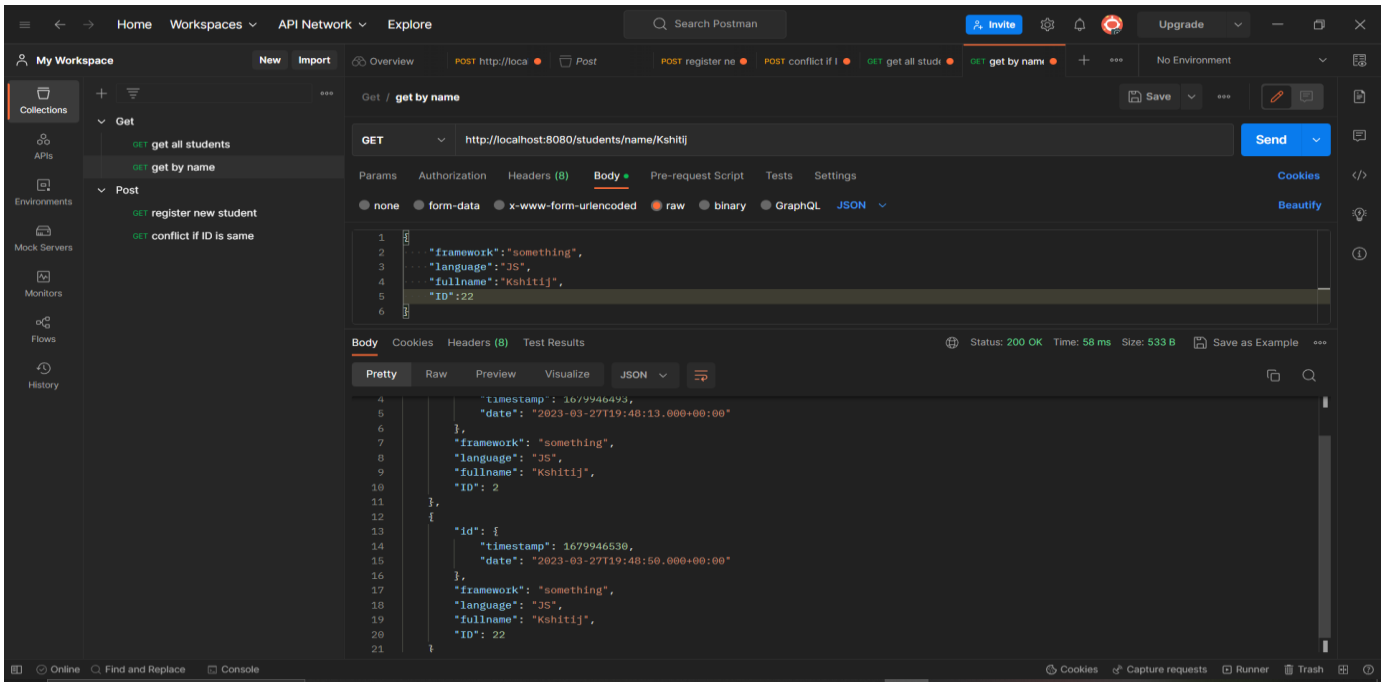
1) Endpoint: “students/”

This end point fetches all the students in the database:

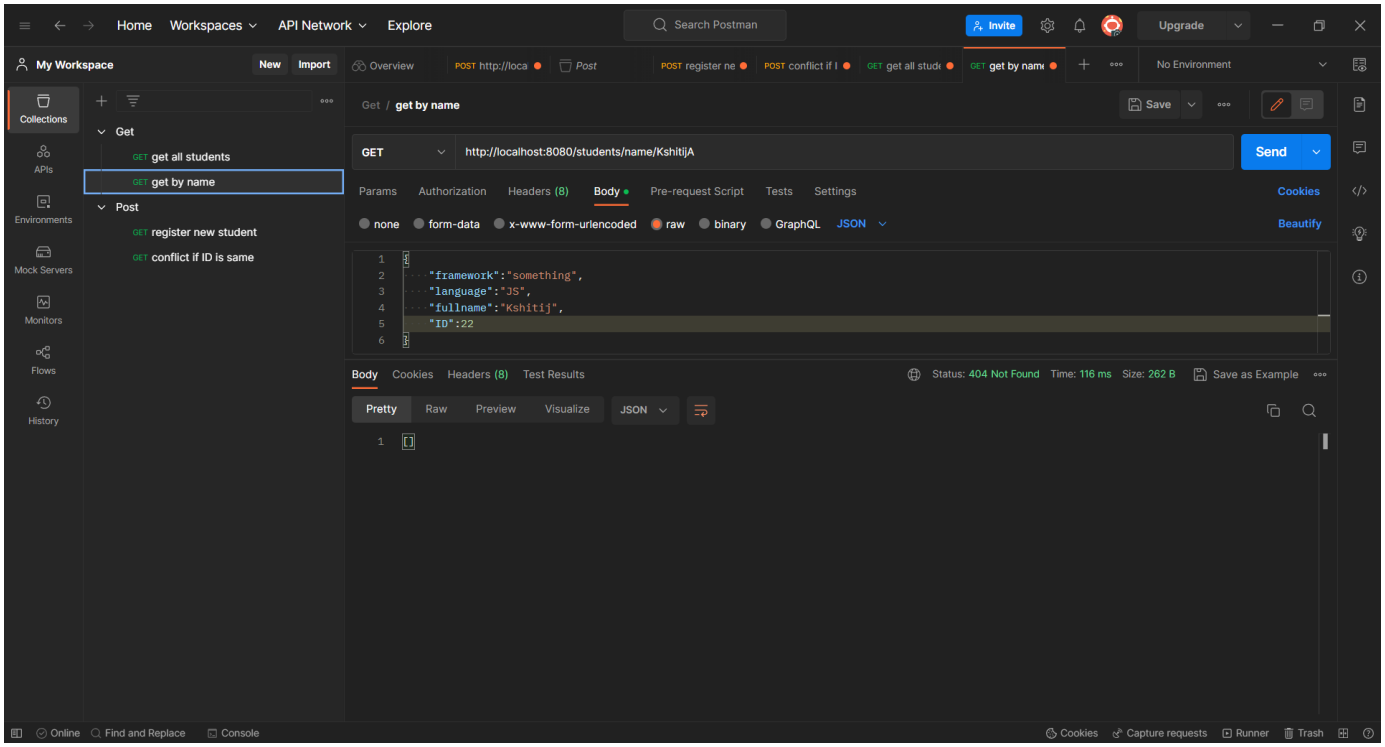


2) Endpoint: “students/name/{Name}”

Return all the entries where the fullname matches Name provided in request (case sensitive)

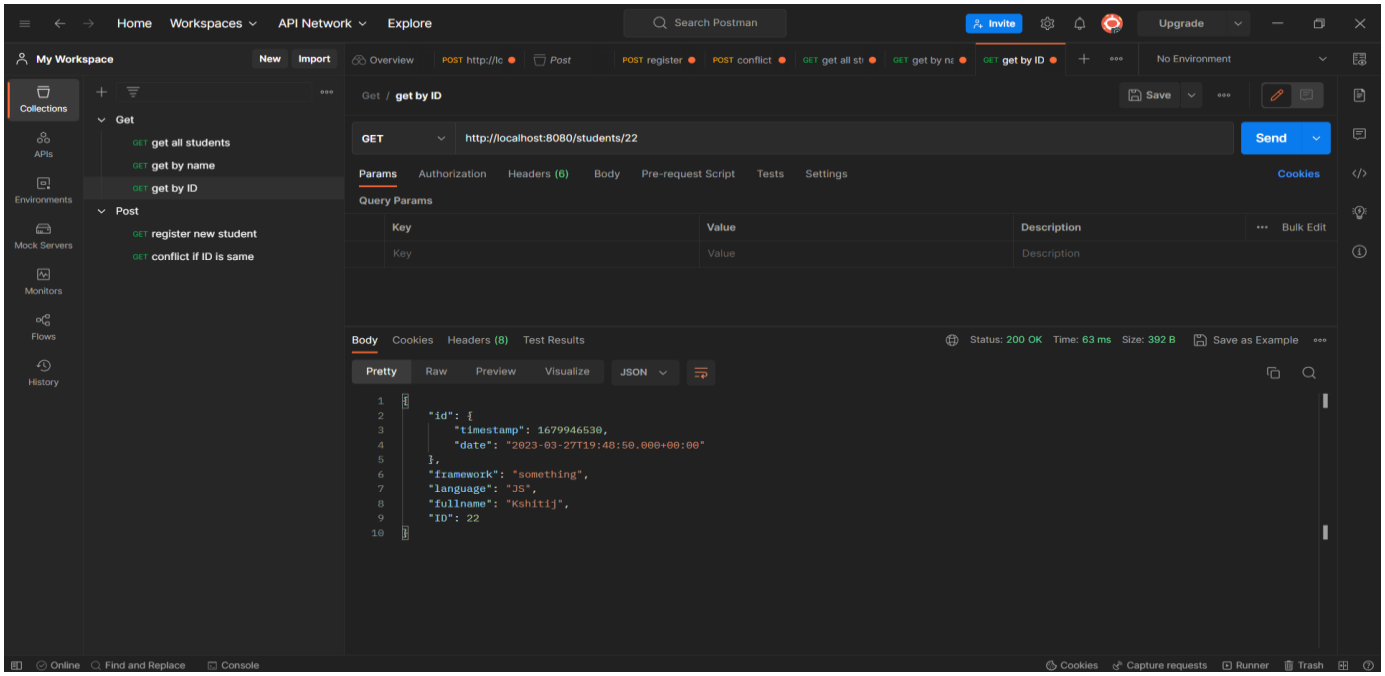


If we do not have any entry with the given name, the server responds with status 404:

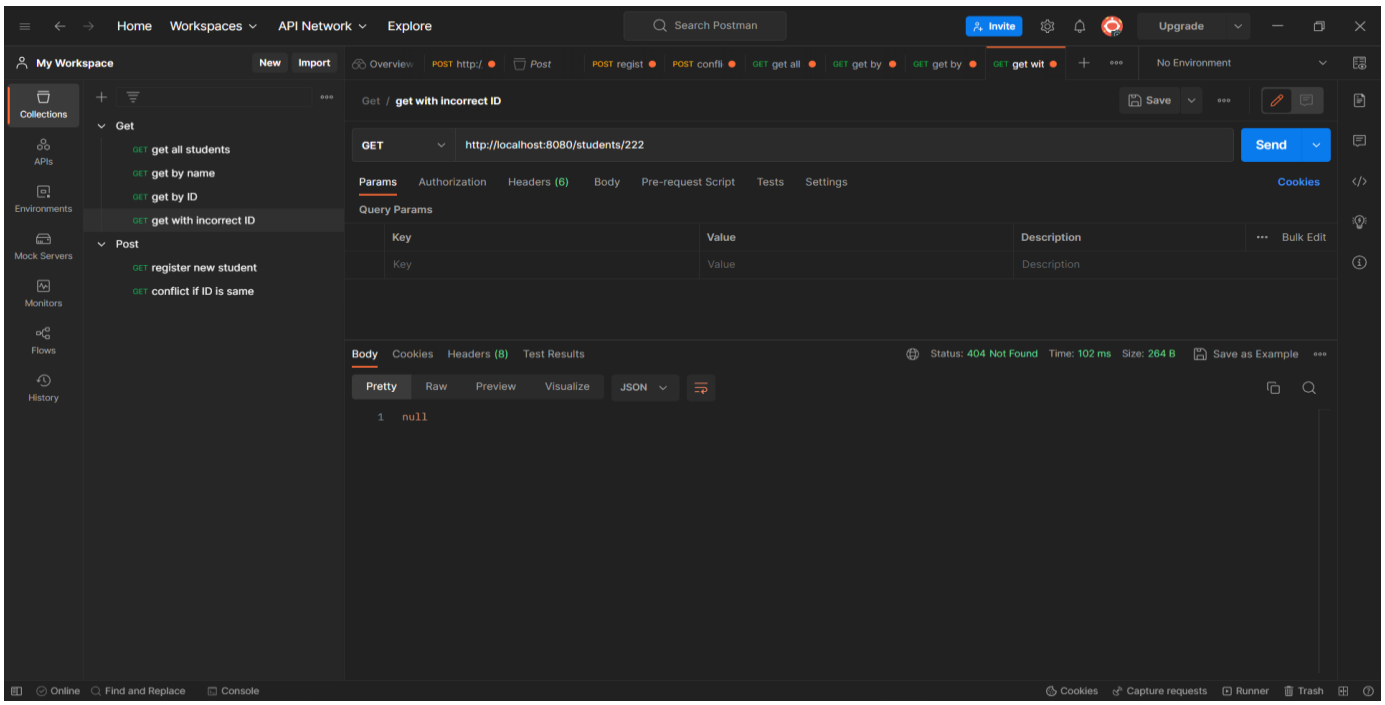


3) Endpoint: “students/{ID}”

This endpoint fetches a student with the ID provided:



If no student is found with the given ID, status of 404 is returned

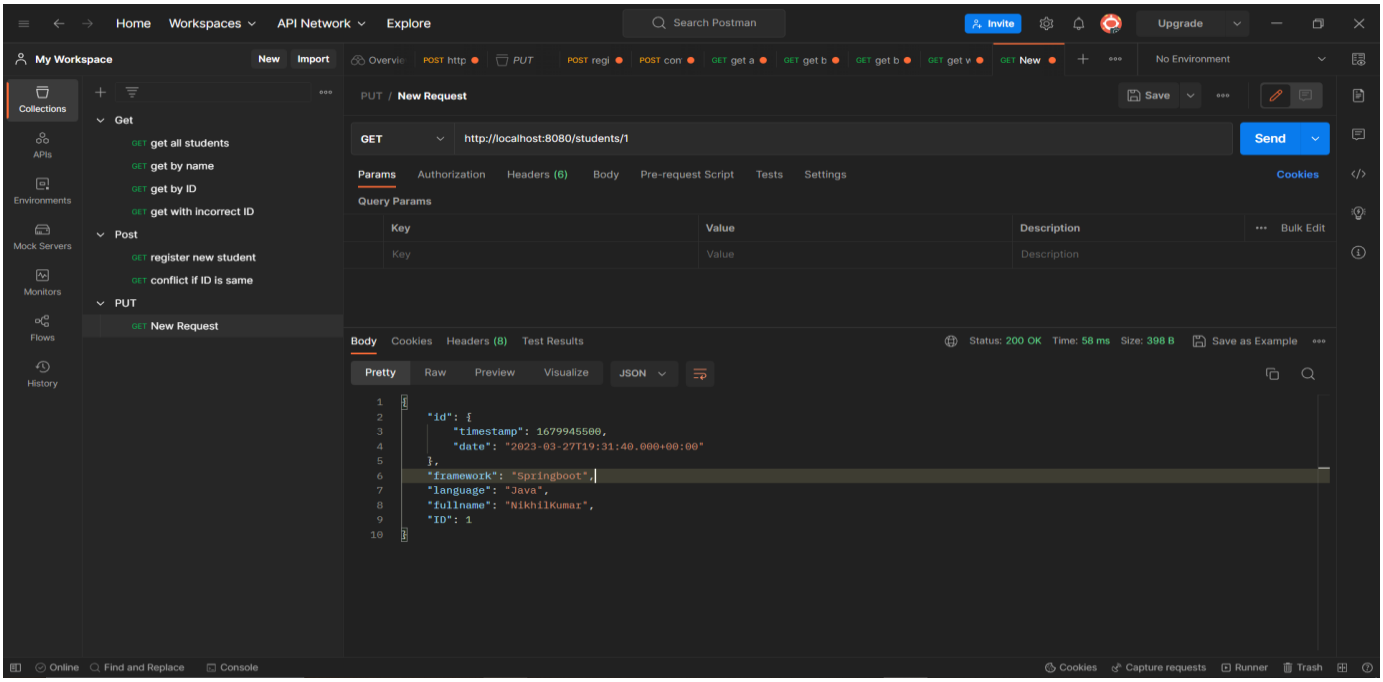


PUT:

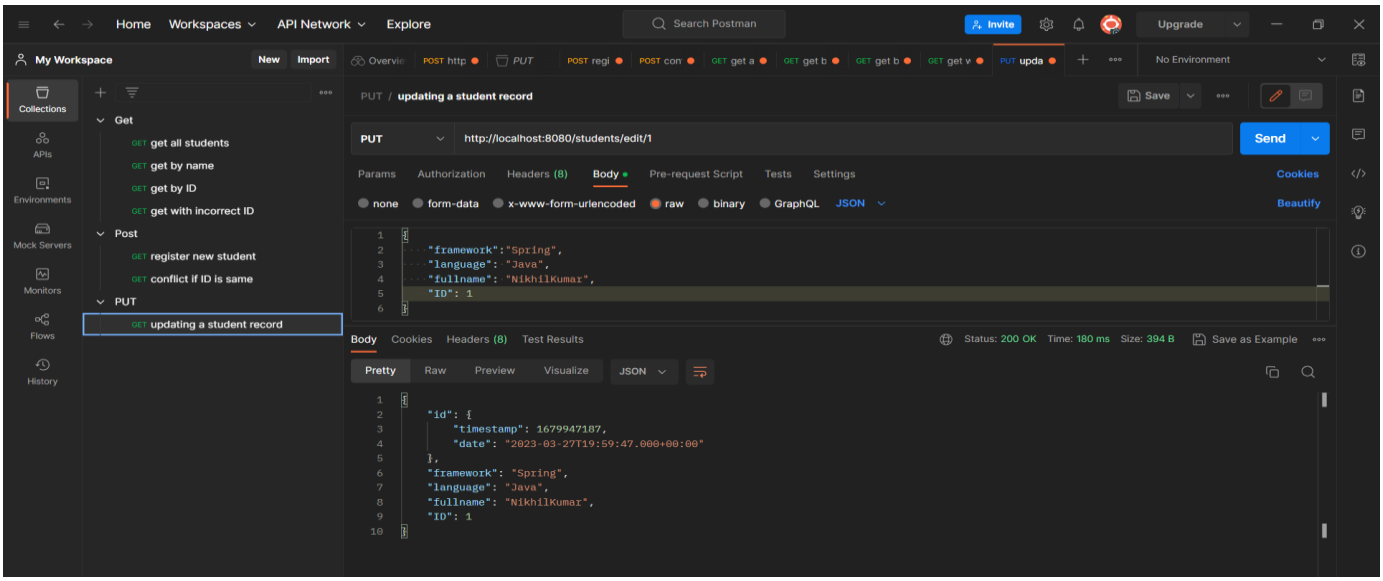
Endpoint: “students/edit/{ID}”

Request body contains the student that needs to be updated.

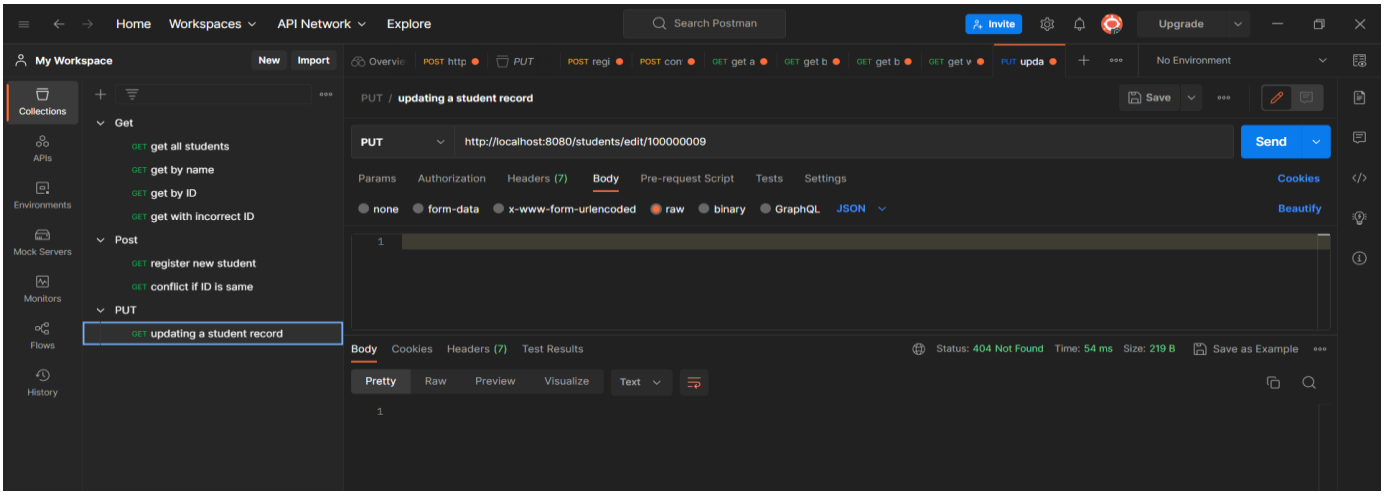
We will update student with ID =1 current state:



After Updating “framework” to Spring:



If no student with such ID is found, server responds with status code of 404

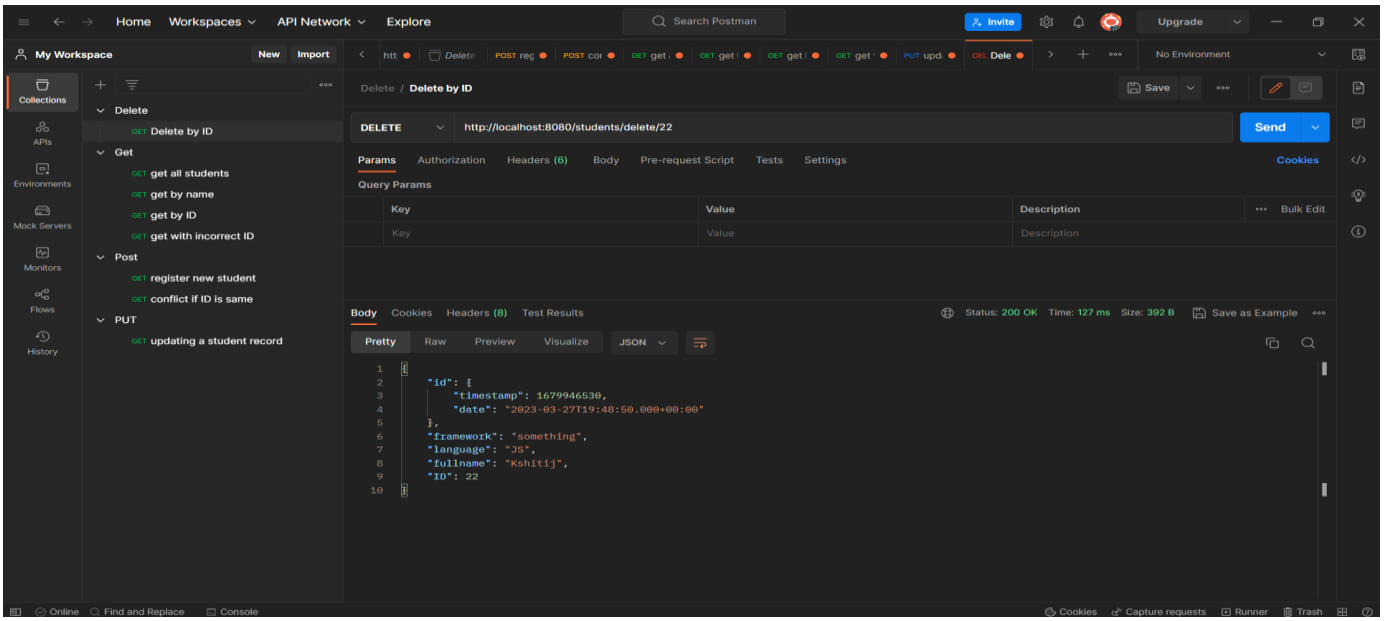


DELETE:

Endpoint: “students/delete/{ID}

The ID obtained from the parameters is used to delete the student where ID matches.

Deleting student with ID=22



If the server is provided with an ID does not match any student record, a status code of 405 is returned: