

21EM5A0513

Vallu.Vijaya

Swarnandhra Institute
of Engineering and
Engineering

Spring Restful API Web Service

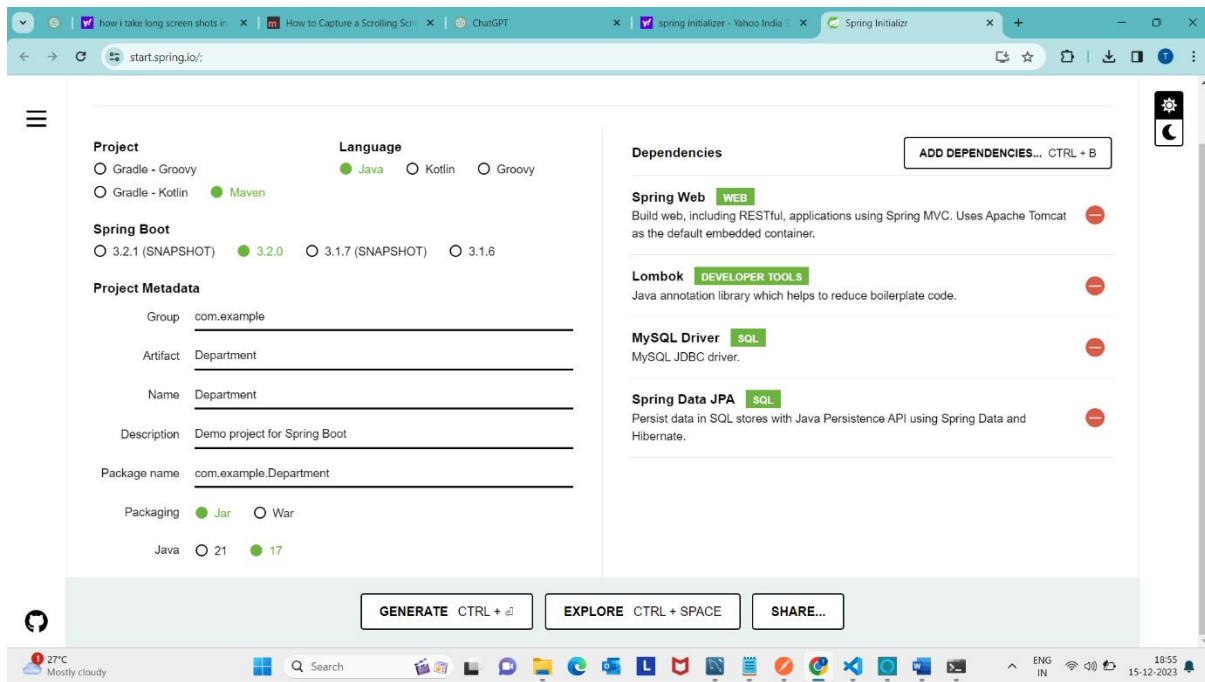
REQUIREMENTS:

- JAVA 17
- SPRING TOOL SUITE IDE
- POSTMAN

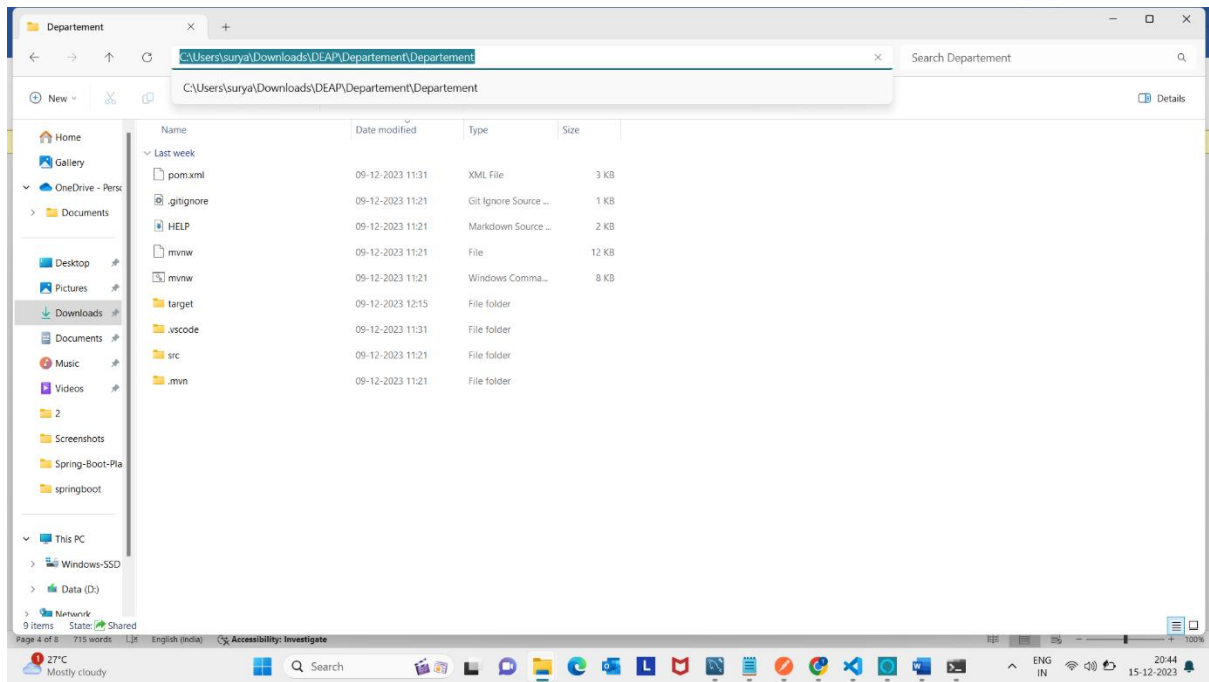
Steps for implementing Basic Spring REST API Service :

Initializing the Spring Boot Application :

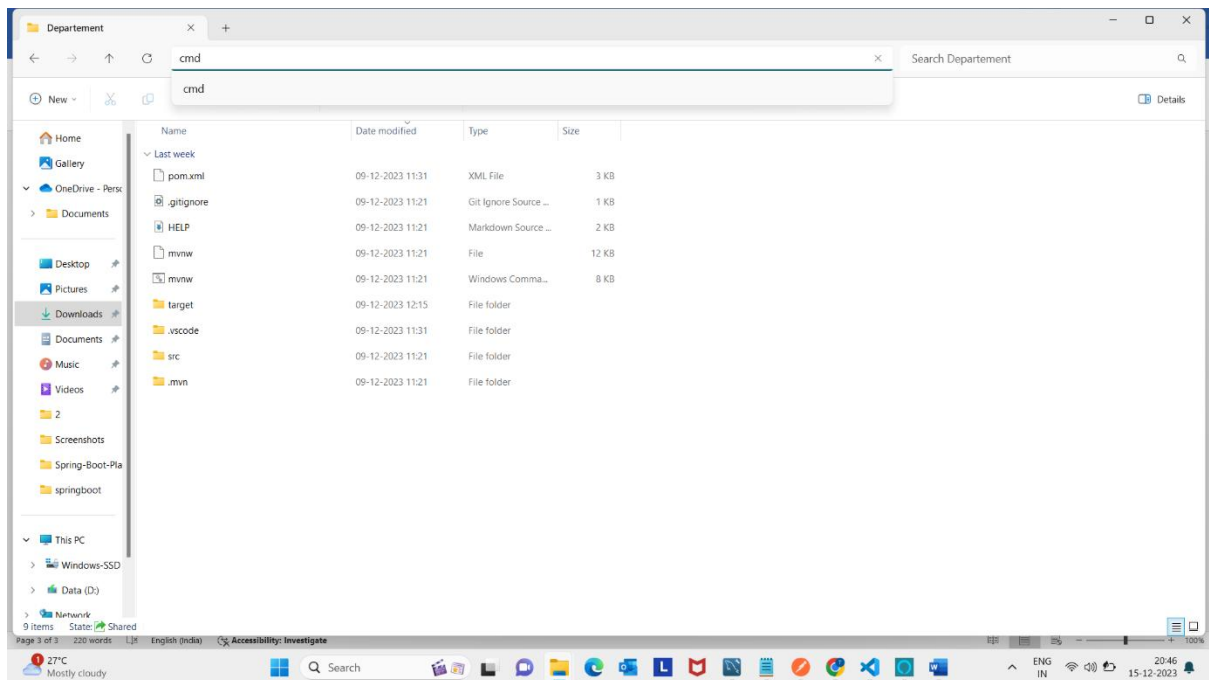
- The first thing you should do is familiarize yourself with the basics of Spring and set up a Spring Boot application.
- Open Google chrome or any other browser and type <https://start.spring.io/> and browse then the following page will be displayed
- Then choose Lang, and Add Dependencies as shown below



- Select the project as Maven and language as java and go with the spring boot version 3.2.0 then name your project metadata as you like and do the packaging with jar(Java Archive) file and with the java version 17.
- Then you need to add some dependencies that are needed for our project.
- As we can see there is a Generate button in the above picture. Click on that Generate button then a **Department** file will be downloaded automatically.
- Then extract that **Department** zip file
- After Extracting the zip file a file folder will be generated and in that folder there are some files like pom.XML , Sr C , git ignore file ,MVP files etc.Then go the folder path and remove the particular path present in it .

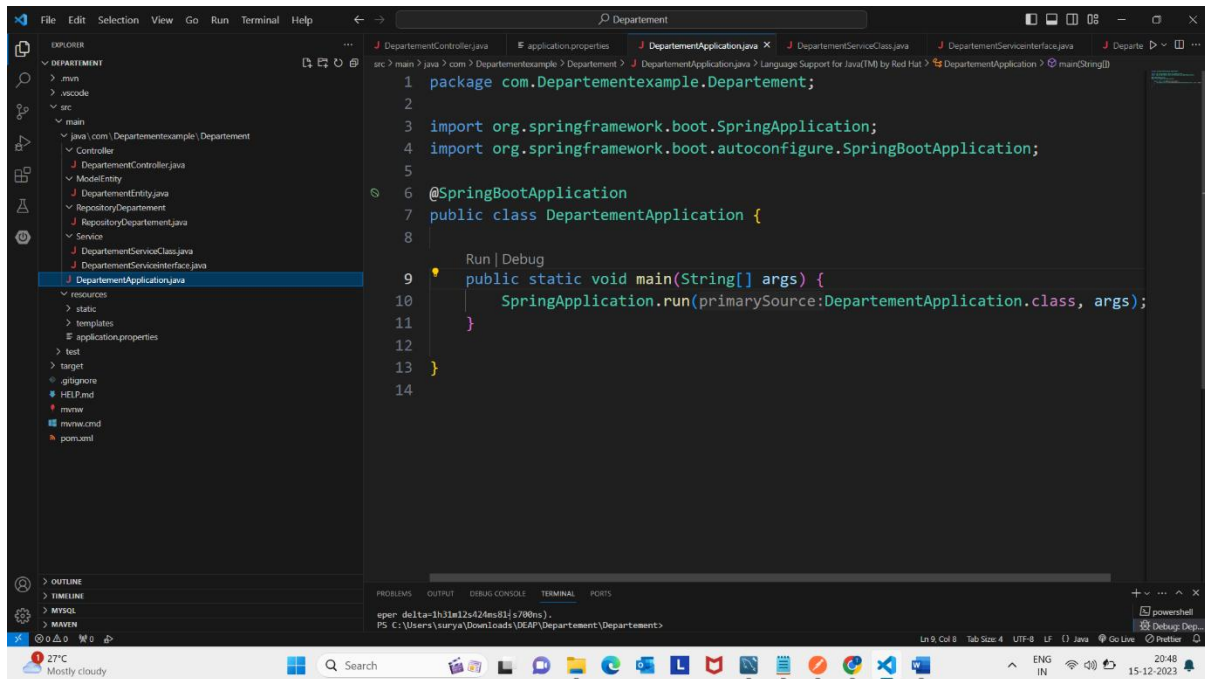


➤ Remove that particular path and type command prompt .



- After clicking enter you will be able to see a command prompt window with that respective path.
- Then type **code .** command then you will be able to see the vscode(Visual Studio Code) window with **Department** folder creation with that particular files.

- After that I created a Model folder and Controller folder inside the java/com/example/curd Demo and inside that Model folder I departmentalization.**java** and inside that Controller folder I created Department Controller.**java** file and inside the Repository and Service folder I created Repository Department.**java** *and Department Service*



Model class:

In Spring MVC, the model works a container that contains the data of the application. Here, a data can be in any form such as objects, strings, information from the database, etc.

Controller class:

In Spring Boot, the controller class is responsible for processing incoming REST API requests, preparing a model, and returning the view to be rendered as a response. The controller classes in Spring are annotated either by the @Controller or the @RestController annotation.

Annotations used in Tutorial Controller.java :

In Spring Boot, are `@RestController`, `@RequestMapping`, and `@GetMapping` annotations commonly used for building Restful API. Here's an explanation of each:

`@RestController`:

- The `@RestController` annotation is a specialized version of the `@Controller` annotation. It is used to indicate that the class defines a Restful API endpoint.
- When you annotate a class with `@RestController`, it implies that every method inside the class is treated as a controller method and returns the response in a format suitable for Restful services (typically JSON).

`@RequestMapping`:

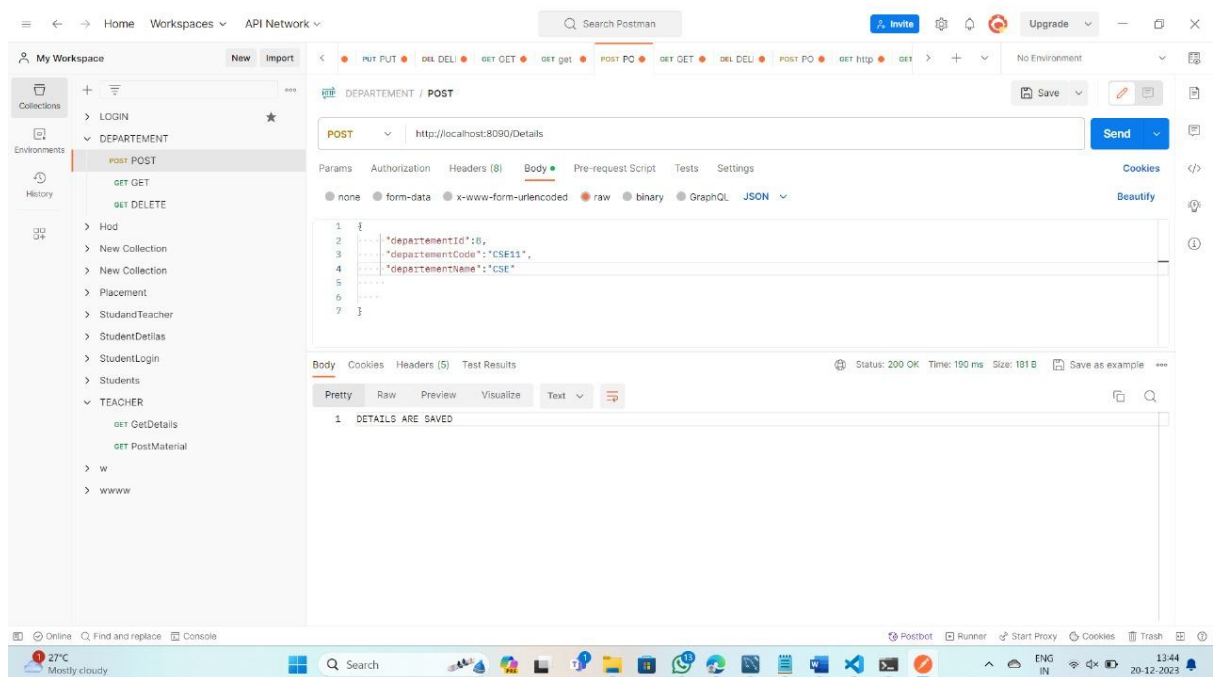
- The `@RequestMapping` annotation is used to map web requests to specific methods in a controller class. It can be applied at the class level and/or method level.
- It allows you to define the base URI for all the methods in the class and then further refine the URI for each method.
- It can specify the HTTP method (GET, POST, PUT, DELETE) and other request parameters.

`@GetMapping`:

- The `@GetMapping` annotation is a specialized version of `@RequestMapping` focused on HTTP GET requests. It is a shortcut for `@RequestMapping(method = RequestMethod.GET)`.
- It simplifies the code by making it more concise when you're dealing specifically with GET requests.
- Here I used only `@RestController`, `@RequestMapping`, and `@GetMapping` annotations and by using the vendor Id I will retrieve the details.

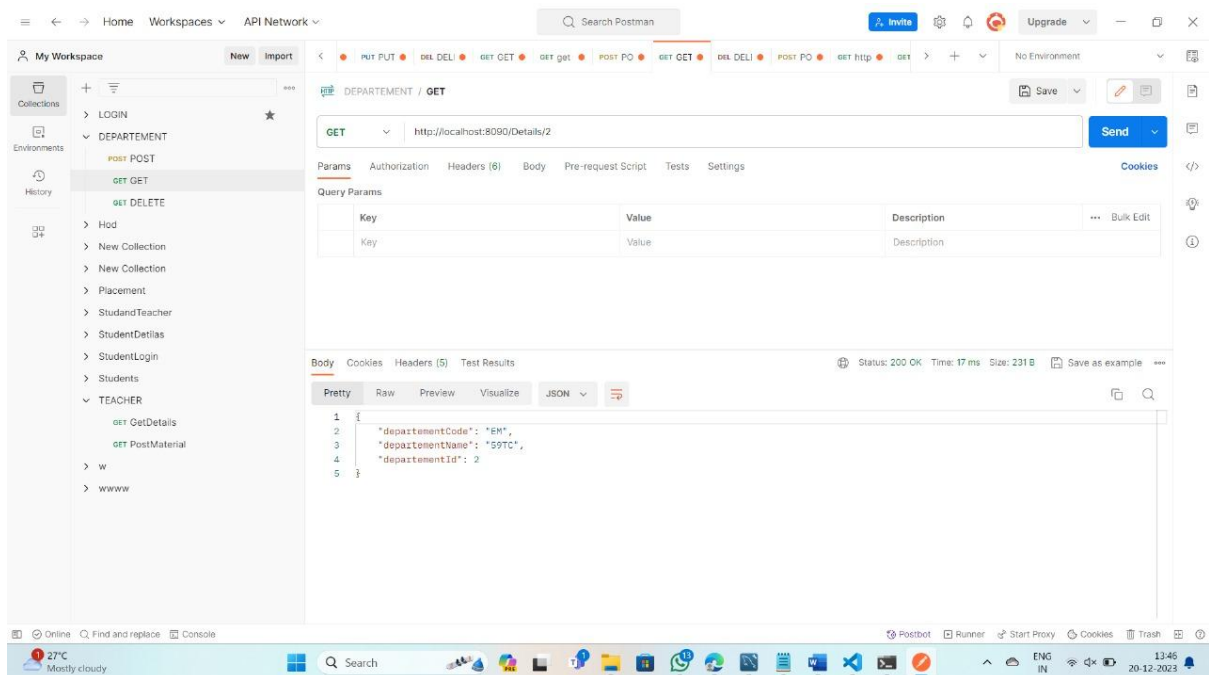
- This code appears to be a Java class representing a Restful API for managing cloud vendor details using the Spring Framework's `@Rest` Controller and `@Request Mapping` annotations.
- Then after run the application. As default the application will run on port 8080 in Apache Tomcat server.
-

@Post mapping:



- Create a new request file in ***** and name it as Put .Then the details are updated.

@Get Mapping :

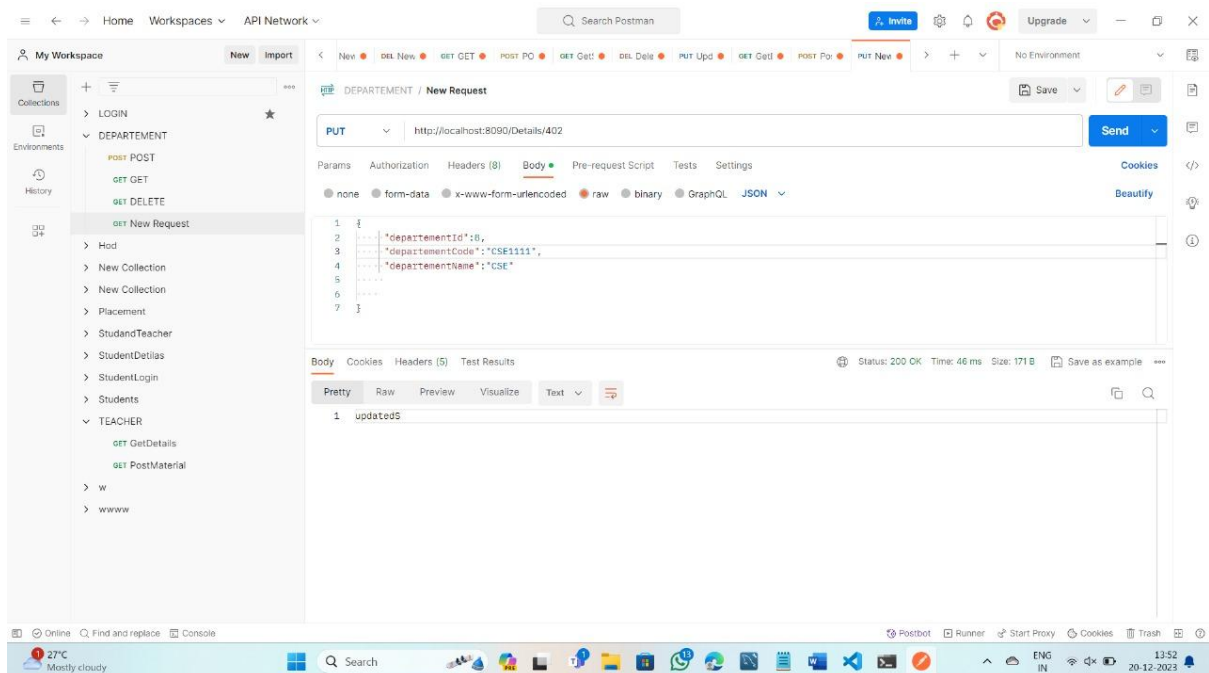


The screenshot shows the Postman interface with a GET request configured. The URL is `http://localhost:8090/Details/2`. The response is a JSON object with the following structure:

```
1 {
2   "departementCode": "EM",
3   "departementName": "597C",
4   "departementId": 2
5 }
```

The status bar indicates a 200 OK response with a time of 17 ms and a size of 231 B.

@Put Mapping :



The screenshot shows the Postman interface with a PUT request configured. The URL is `http://localhost:8090/Details/402`. The request body is a JSON object with the following structure:

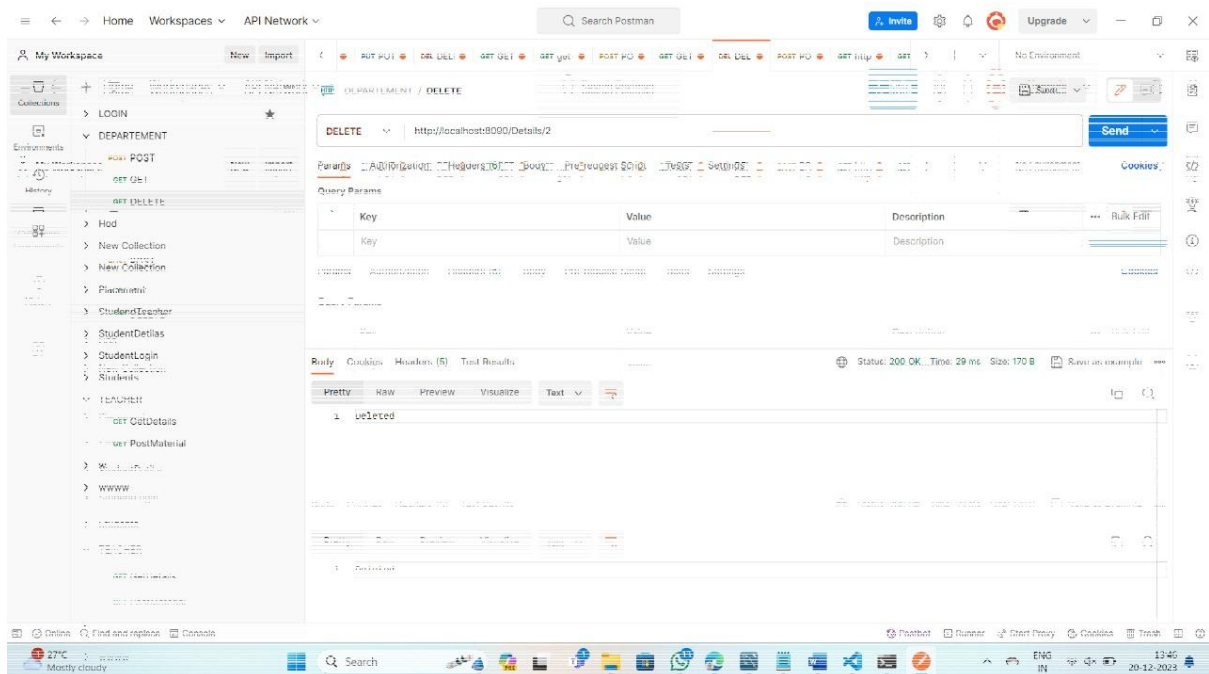
```
1 {
2   "departementId": 8,
3   "departementCode": "CSE1111",
4   "departementName": "CSE"
5 }
```

The response is a JSON object with the following structure:

```
1 {
2   "updated": true
3 }
```

The status bar indicates a 200 OK response with a time of 46 ms and a size of 171 B.

@Delete Mapping :



.....THANK YOU.....