# B. NOKESH

Reg no: 20EM1A0416

# Swarnandhra Institute Of Engineering And Technology

REQUIREMENTS

- IDE (VS CODE)
- SPRING INITILIZER
- POST-MAN
- JAVA 17

## STEPS FOR BUILDING RESTAPI USING SPRING BOOT IS GIVEN BELOW:

1. Open the browser and enter the url https://start.spring.io/ , then spring intializr website will open.
2. Then select the corresponding option such as ,in the project select the maven ,in the language select the java, select the spring boot version as 3.2.0. change the meta data as per the your comfort ,select the jar in packaging, and select the corresponding java version in your system ,at the last add the required dependencies in it .
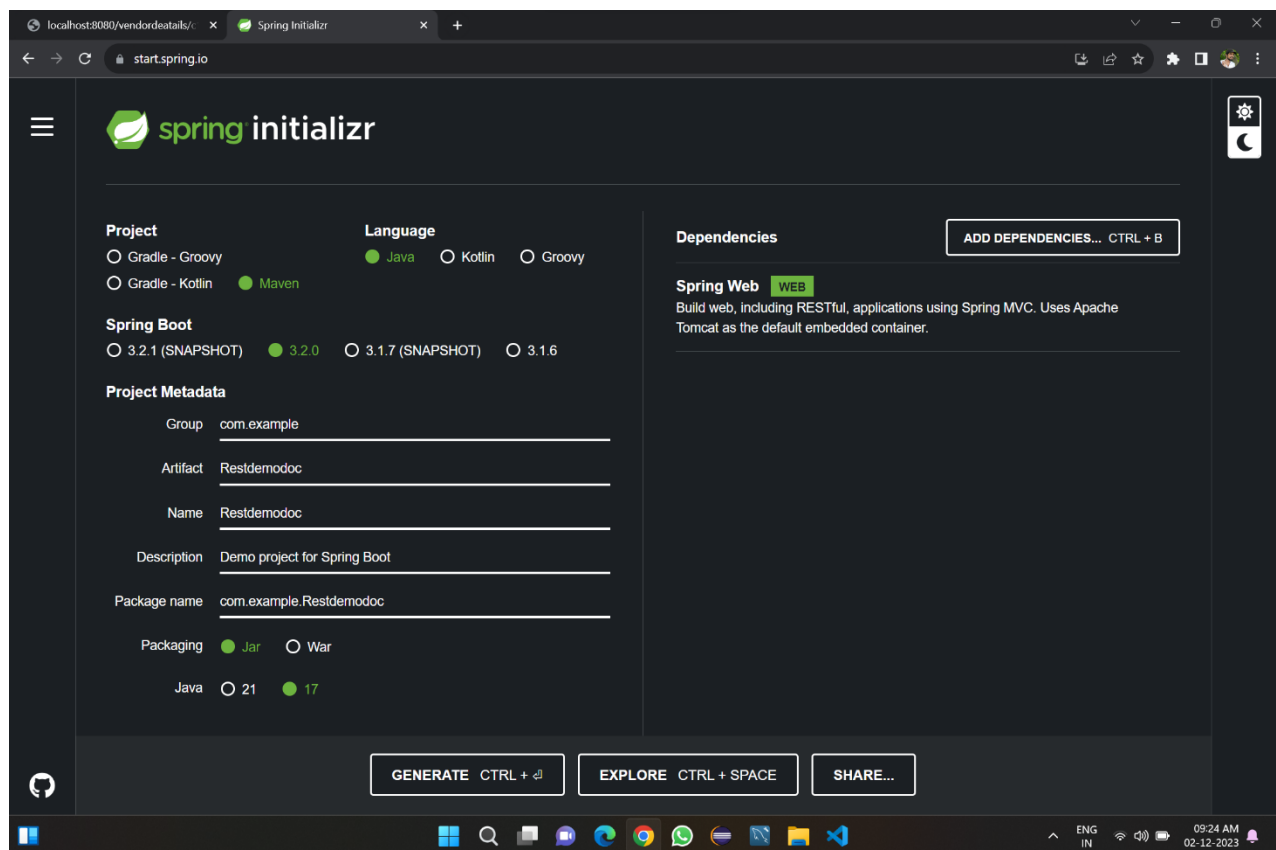
3. After click on the Generate ,then the Zip file is automatically will download and open the downloaded zip file and right click on it . Then tap Extract all and click "EXTRACT".
4. In the extracted file click on the path of search bar and enter "CMD" commad .
5. Then the COMMAND PROMPT app will open automatically , then type <<< code .
6. It directly goes to VisualStudioCode APP.
7. Then click on the folder ,in the folder there are sub folders and on one by one SRC > MAIN> JAVA , there will be default application
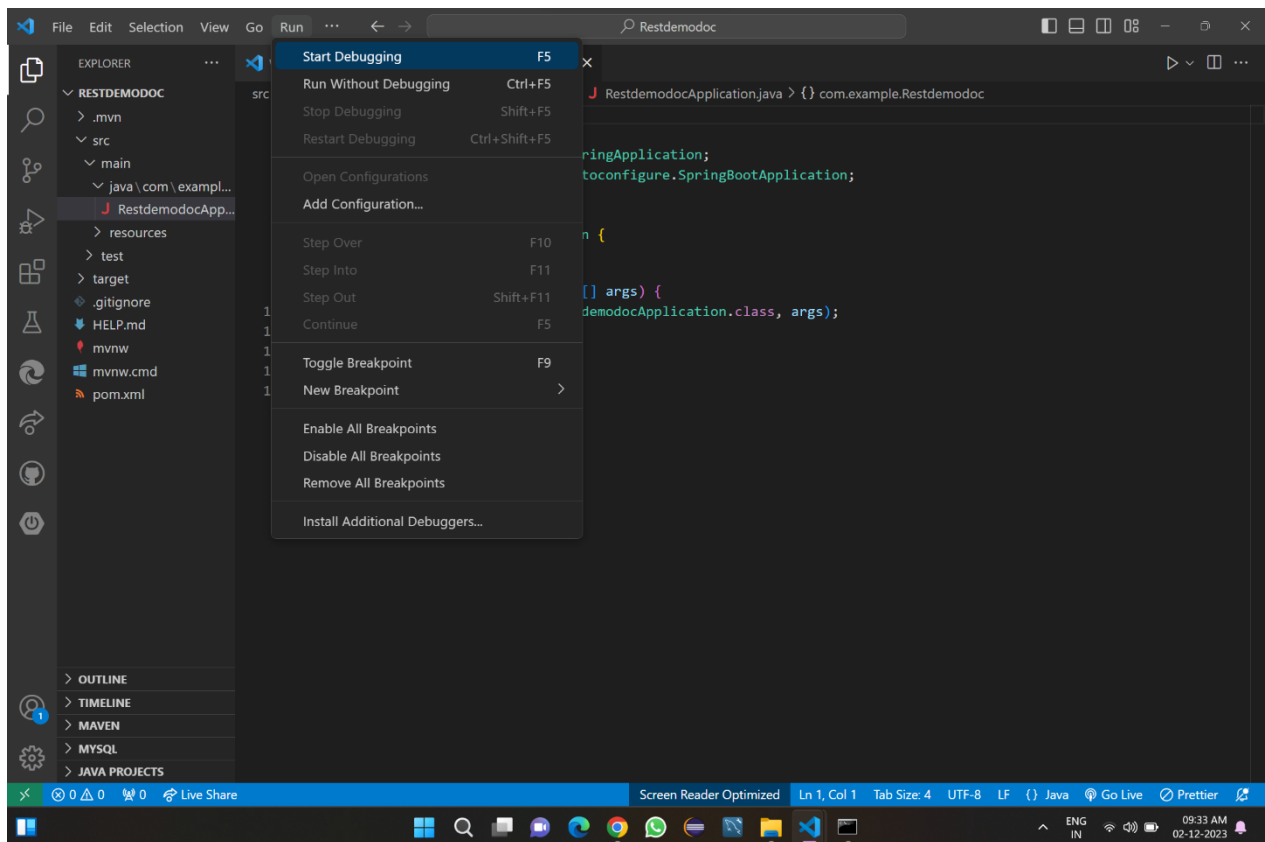
8. Then run the program and "START DEBBUGGING"., if all process done will means then it will shows a TOMCAT webserver started at port 8080.
9. Then open the browser and search in the url section as localhost:8080, then it returns WhiteLabel Error Page.
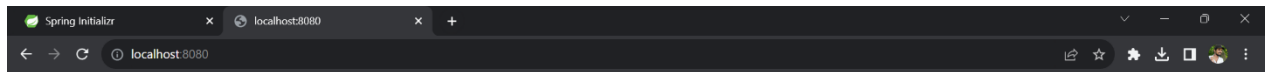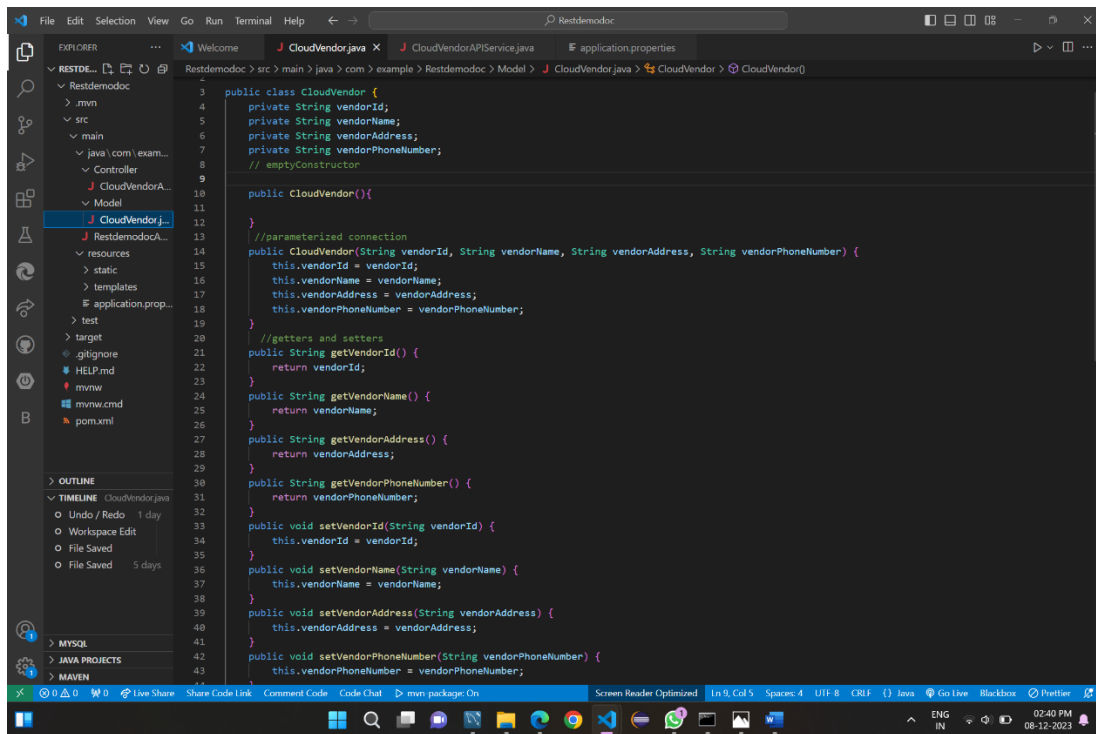

10. In that That API1 sub folder create a new folder named as "Model".

11. In that folder create a File named as "VendorDetails.java" as shown below…. and enter the code in it.

12. And create a another Folder named as "Controller", in that folder create a file name as CloudVendor.java and enter the code in it.
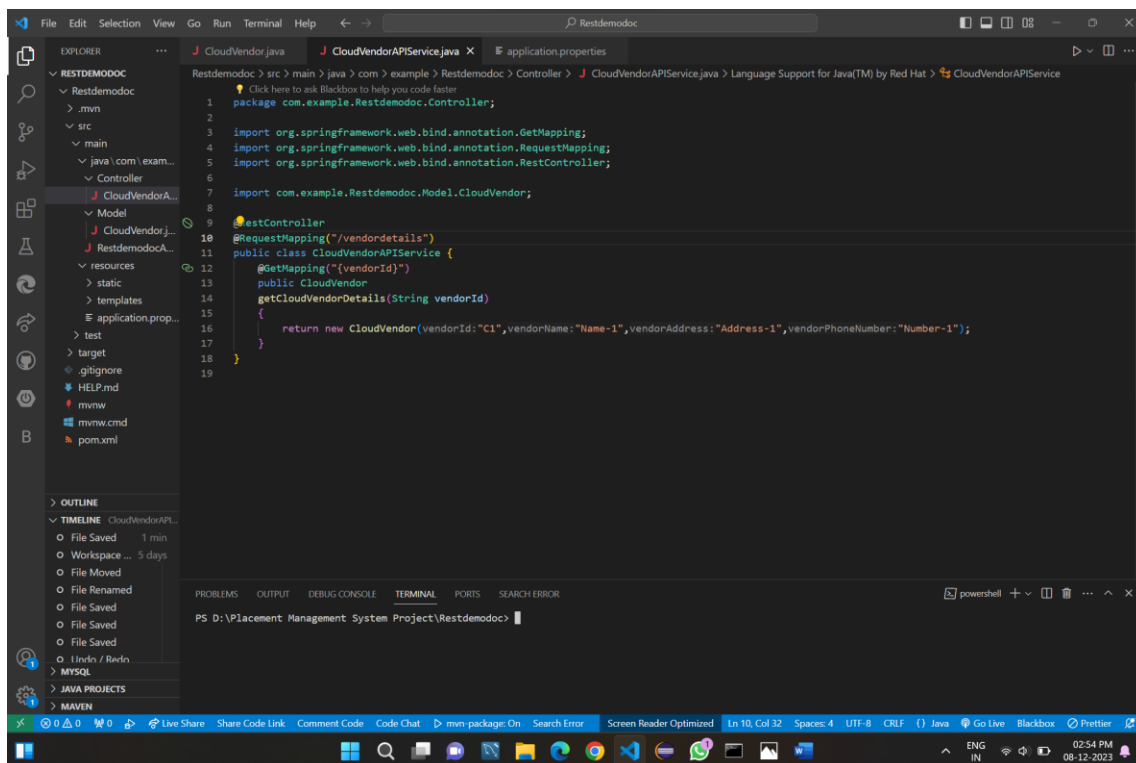
13. After completing the code then click on the RUN and "START DEBBUGGING". This shows a TOMCAT webserver at port 8080.
14. Then the web address as localhost:8080/vendordetails/c1

{"vendorId":"C1","vendorName":"Name-1","vendorAddress":"Address-1","vendorPhoneNumber":"Number-1"}
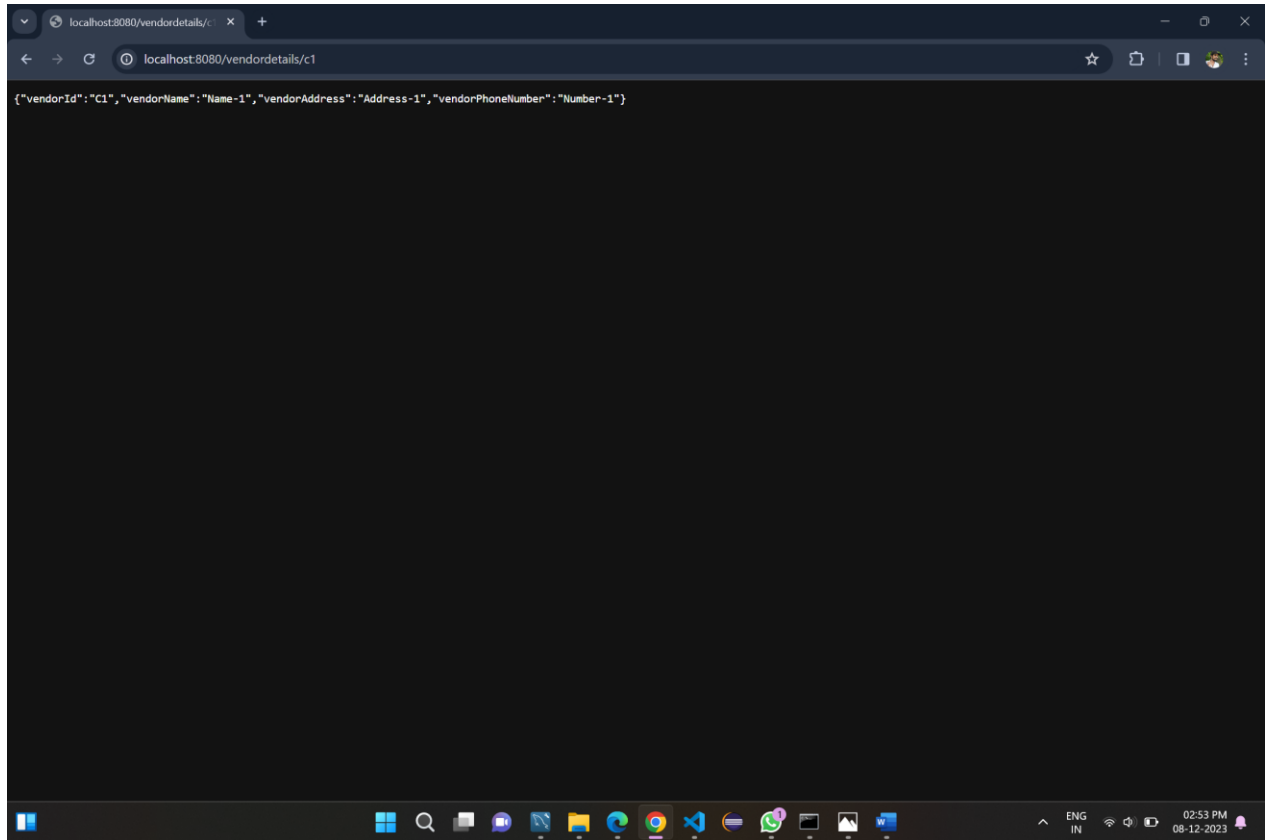
Performing CRUD Operations in Spring Boot

Postman

Postman serves as a valuable tool for interacting with and testing the RESTful APIs developed using Spring.

By using post man perform GET PUT POST DELETE operations of the cloud vendor details

Update CloudVendorAPIService.java class
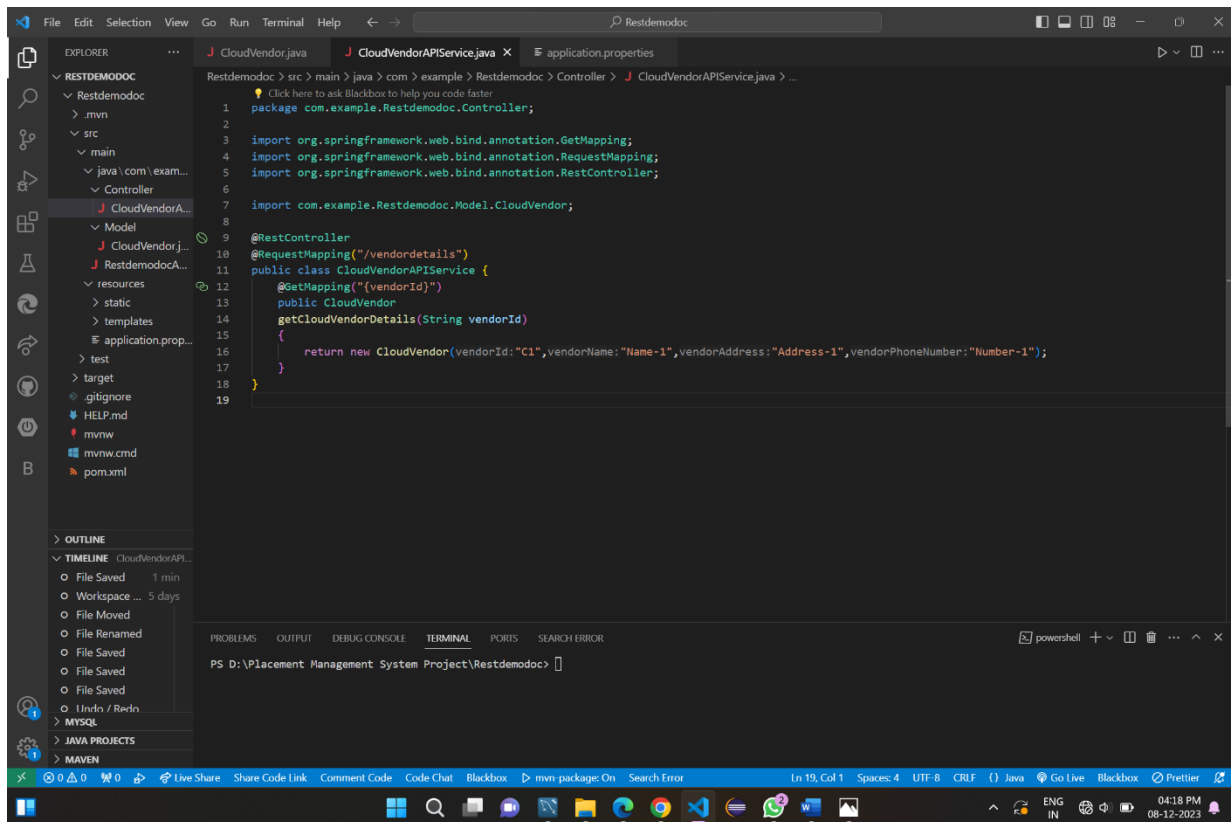
- In the class we have created different methods for getting vendor details, posting vendor details, updating vendor details and deleting vendor details.
- each method has its own mapping annotation.
- for getting the vendor details we use @GetMapping followed by the vendorId which specify to display the details of that particular vendor.

- we bound the @GetMapping with getVendorDetails.
- after this method is executed we will be displayed with the details of that user

This code defines a REST controller in a Spring Boot application that handles CRUD (Create, Read, Update, Delete) operations for a CloudVendor entity.

Controller Class Overview:

@RestController: Indicates that this class is a REST controller, and its methods return data to the client rather than rendering views.
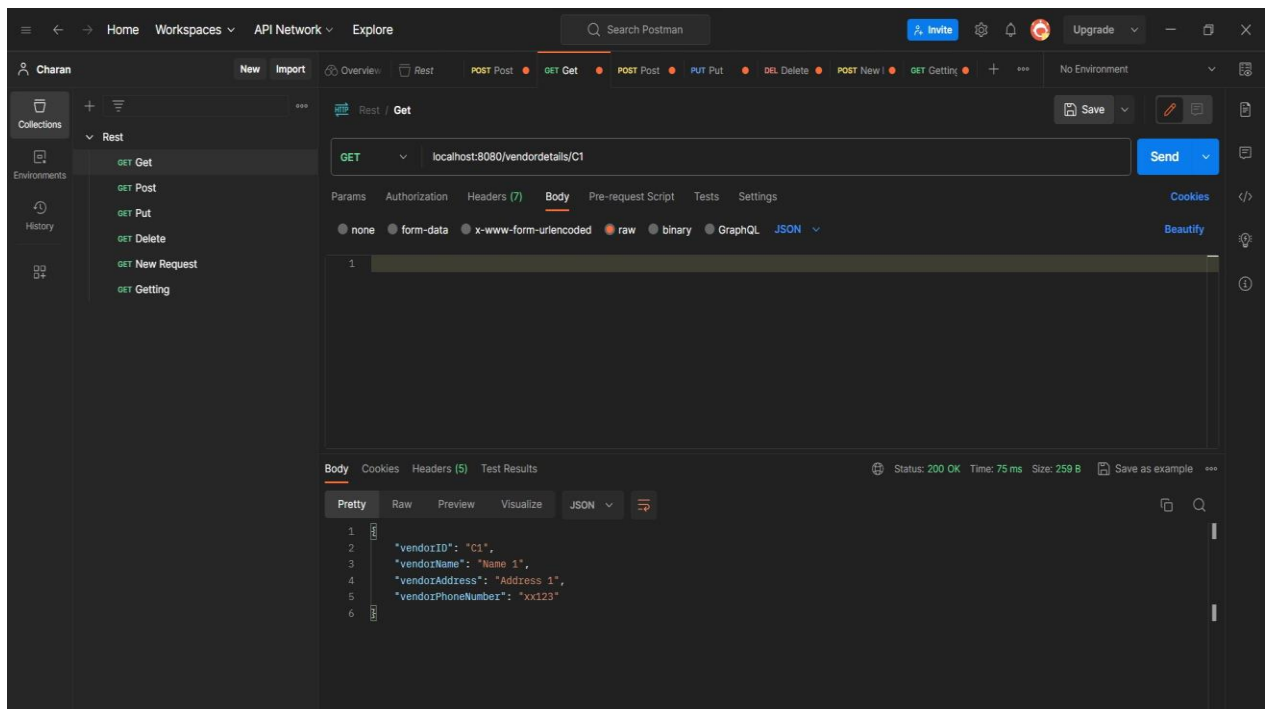
@RequestMapping("/details"): Specifies the base path for all endpoints defined within this controller.

Endpoints and Operations:

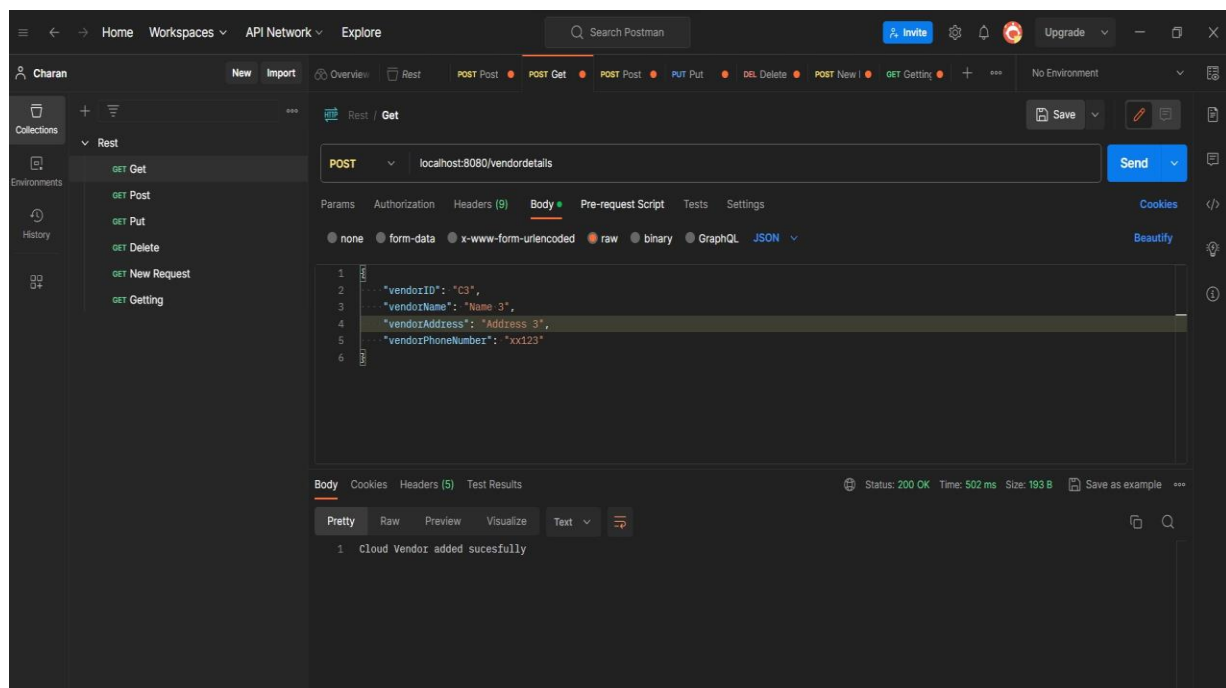**@GetMapping("{vendorId}"):** Handles HTTP GET requests for retrieving vendor details by vendorId.

Method: getCloudvendordetails(String vendorId)
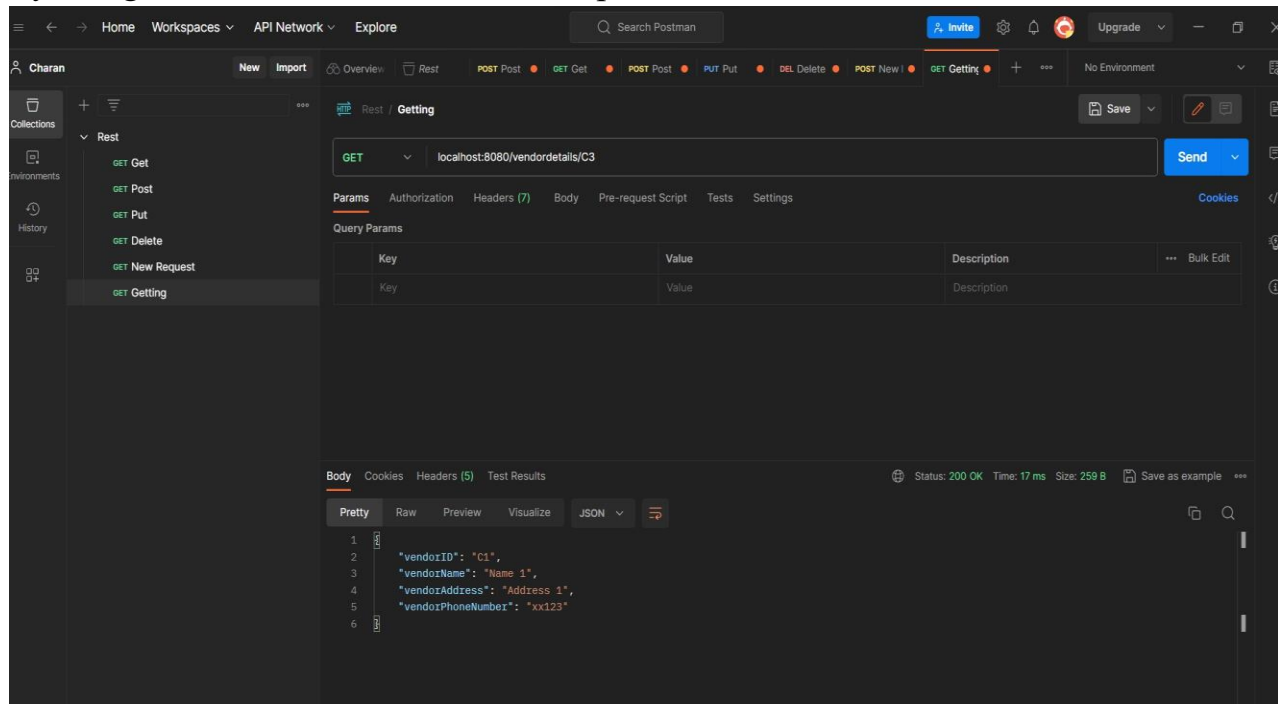Returns: A CloudVendor object corresponding to the provided vendorId.

@PutMapping: Handles HTTP PUT requests for updating vendor

details. Method: putVendorDetails(@RequestBody CloudVendor

cloudVendor)

Accepts a CloudVendor object in the request body and updates the controller's cloudVendor field.

By using GET method we can see the updated data.

@PostMapping: Handles HTTP POST requests for creating or updating vendor details.

Method: postCloudvendordetails(@RequestBody CloudVendor cloudVendor) Accepts a CloudVendor object in the request body and updates the controller's cloudVendor field.
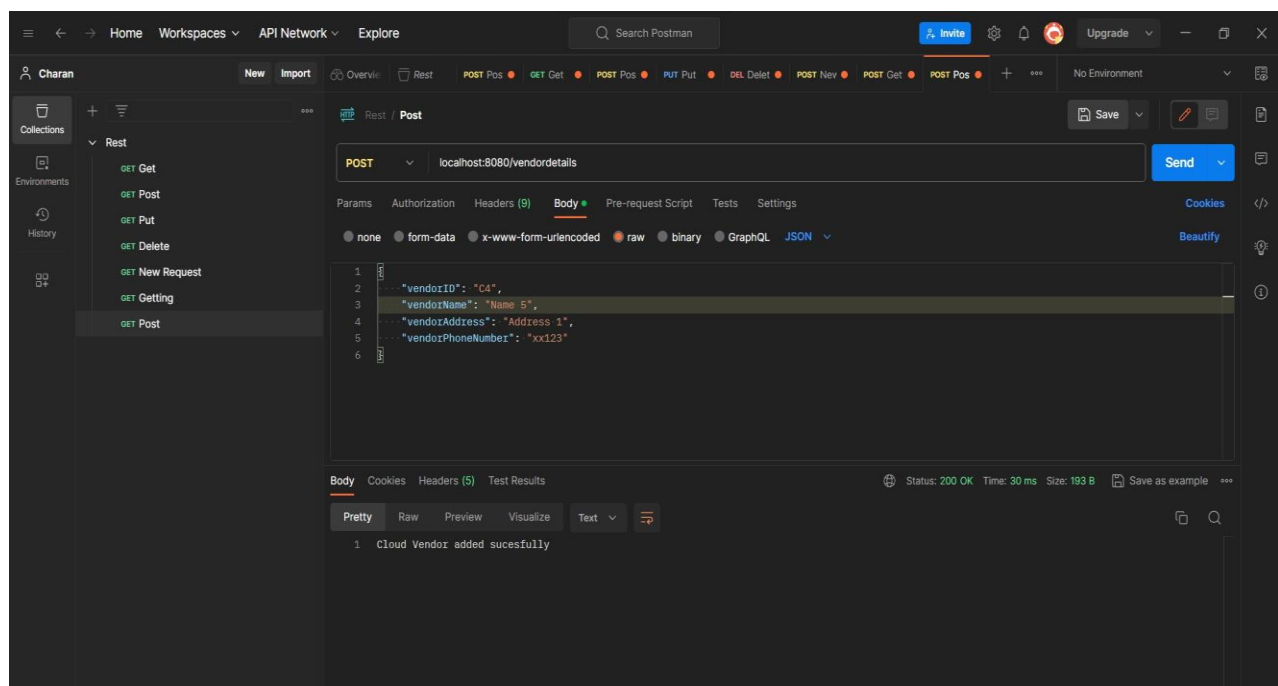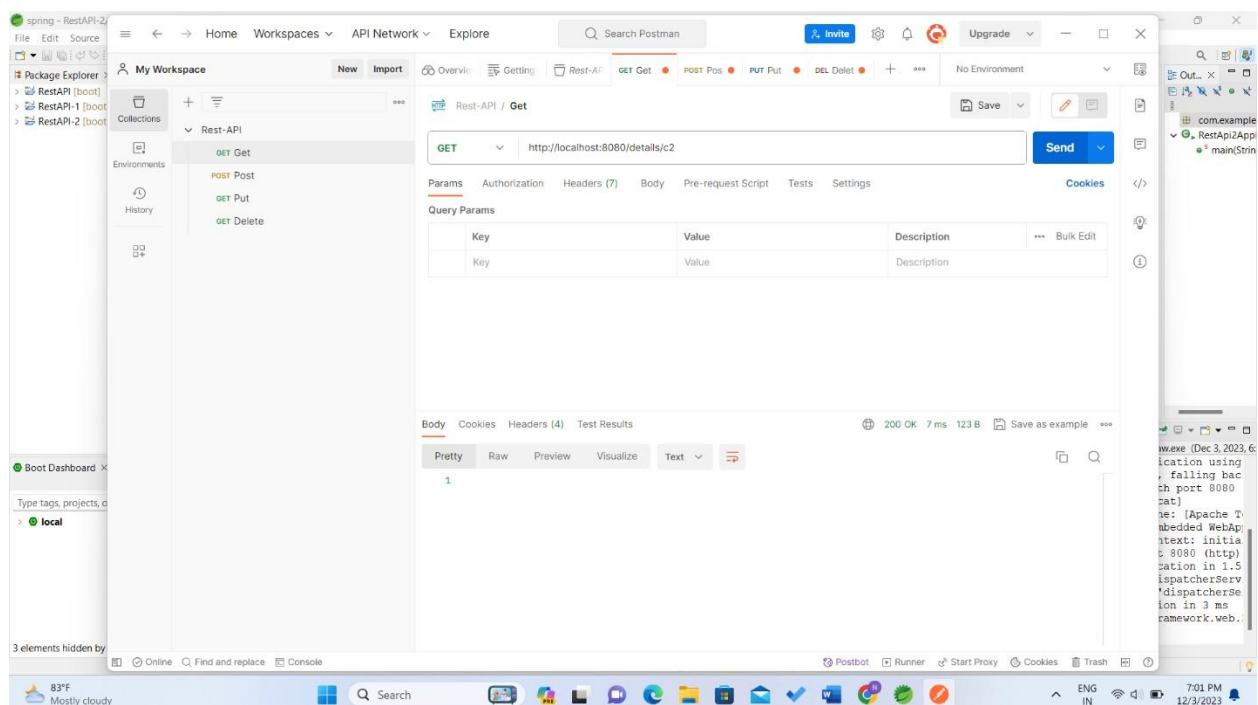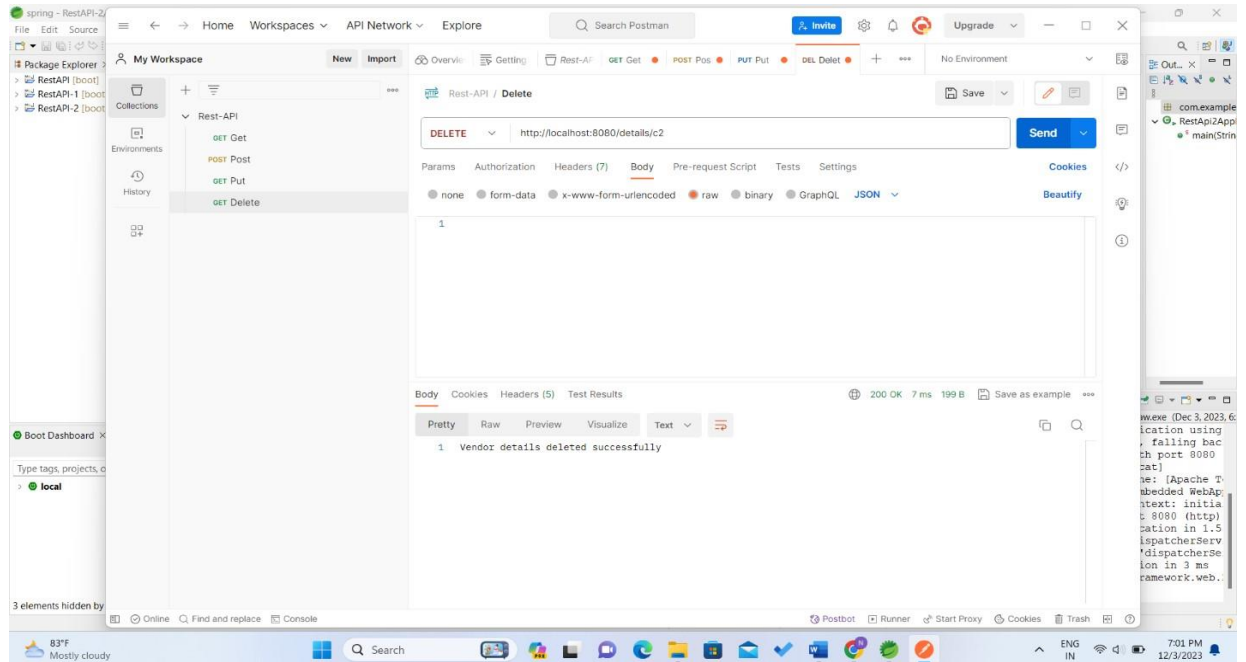
@DeleteMapping("{vendorId}"): Handles HTTP DELETE requests for deleting vendor details by vendorId.

Method: deleteVendorDetails(String vendorId)
Deletes the stored CloudVendor object by setting the controller's cloudVendor field to null.

Conclusion:

In summary, using Spring Boot with Postman made it easy to create, test, and manage vendor details through simple and efficient CRUD operations.