Project 1: Employee Management

System Date:

Description:

Create a Spring Boot application to manage employee records. The system should allow adding new employees, fetching details of existing employees, updating employee details, and deleting employees.

Entities Fields:

- Employee: id (Primary Key, auto-generated), name, email, department, salary.

Features:

- Create an Employee: POST request to /employees with employee details in the request body.
- Get All Employees: GET request to /employees.
- Get Employee by ID: GET request to /employees/{id}.
- Update an Employee: PUT request to /employees/{id} with the employee details to be updated in the request body.
- Delete an Employee: DELETE request to /employees/{id}.

Model:

```
package com.example.demo;
import jakarta.persistence.*;
@Entity
public class Model {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;
    private String name; private
    String email; private String
    department; private double
    salary;

    public Integer getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }
```

```
    public void setEmail(String email) {
        this.email = email;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }
}
```

Controller:

```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class Controller {
    @Autowired
    private Serv serv;

    @GetMapping("/employee")
    public List<Model> display(){
        return serv.get_all();
    }

    @GetMapping("/employee/{id}")
    public Model display(@PathVariable Integer id){
        return serv.get(id);
    }

    @PostMapping("/employee")
    public ResponseEntity<Model> create(@RequestBody Model m){
        return serv.create(m);
    }

    @PutMapping("/employee/{id}")
    public ResponseEntity<Model> update(@RequestBody Model m, @PathVariable Integer id){ return
        serv.update(m, id);
    }

    @DeleteMapping("/employee/{id}")
```

```
    public String delete(@PathVariable Integer id){
        return serv.delete(id);
    }
}
```

Repo:

```
package com.example.demo;
import org.springframework.data.jpa.repository.JpaRepository;

public interface Repo extends JpaRepository<Model, Integer> {

}
```

Service

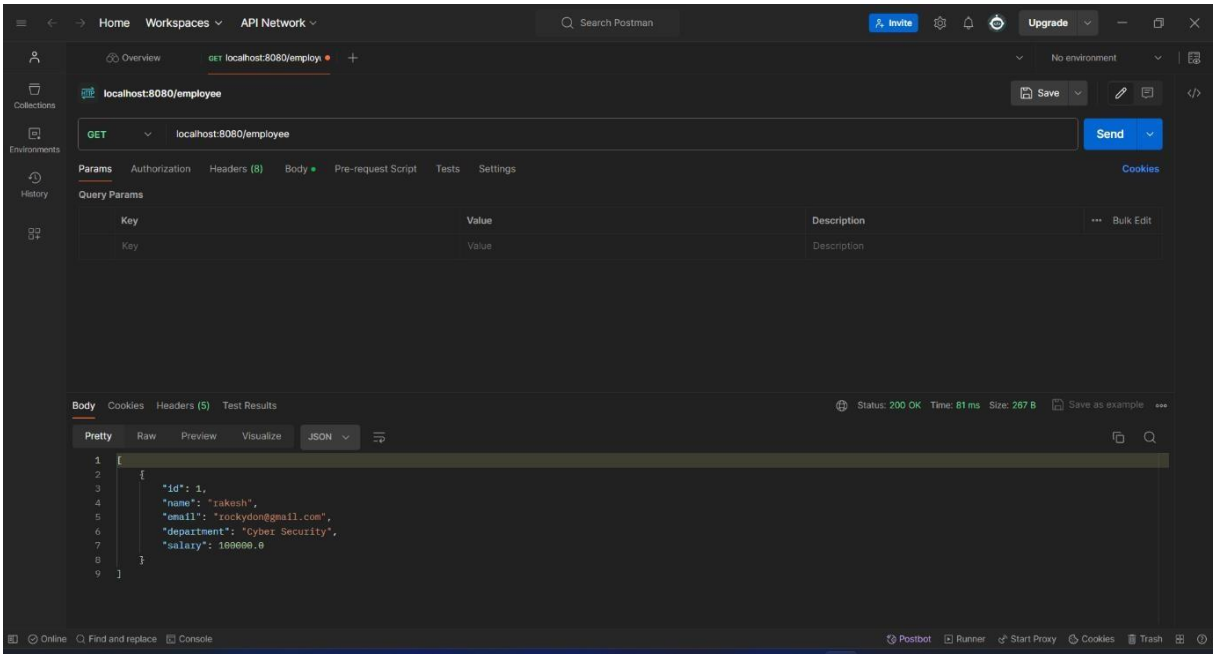
```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class Serv {
    @Autowired private
    Repo repo;

    public List<Model> get_all() {
        return repo.findAll();
    }

    public ResponseEntity<Model> create(Model m) {
        repo.save(m);
        return ResponseEntity.ok(m);
    }

    public Model get(Integer id) {
        return repo.findById(id).orElse(null);
    }

    public String delete(Integer id) {
        repo.deleteById(id);
        return "deleted successfully";
    }

    public ResponseEntity<Model> update(Model m, Integer id) {
        m.setId(id);
        repo.save(m);
        return ResponseEntity.ok(m);
    }
}
```

Application Property:

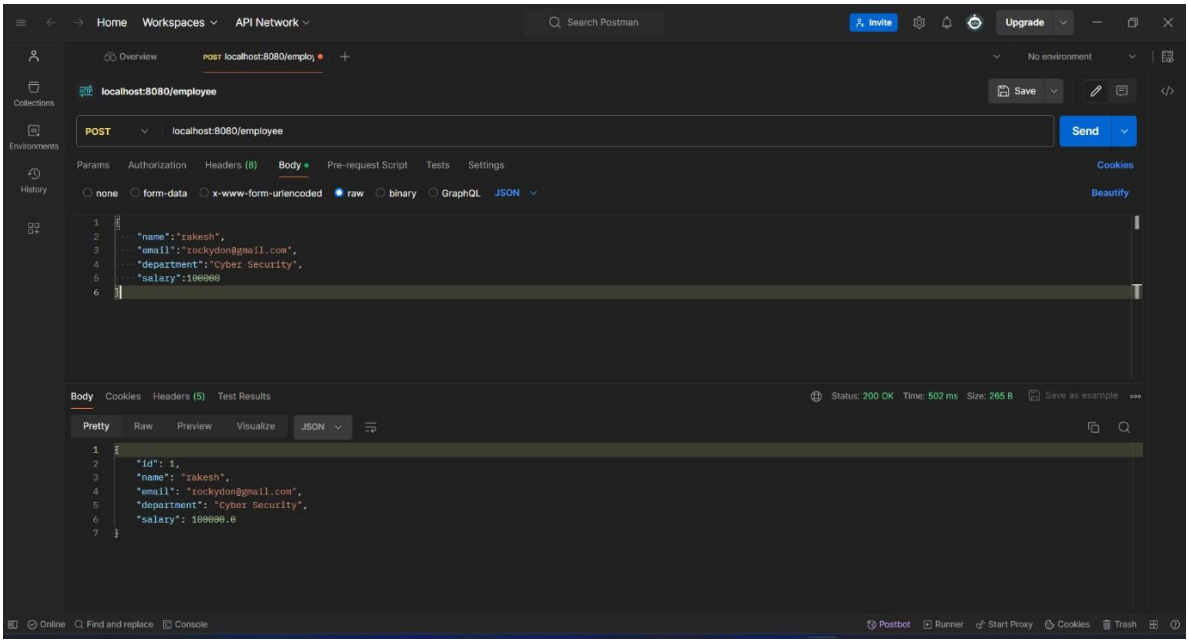
```
spring.application.name=ex-1
spring.datasource.url=jdbc:mysql://localhost:3306/ex1
spring.datasource.username=root
```

```
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database=mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

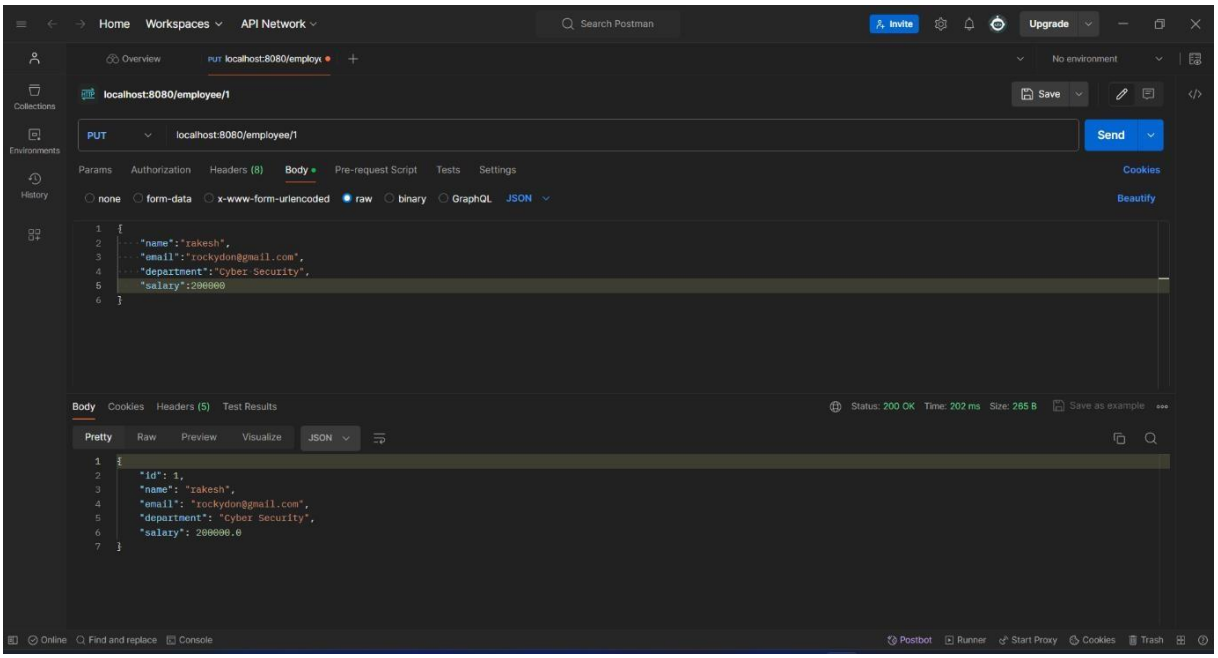
GET:



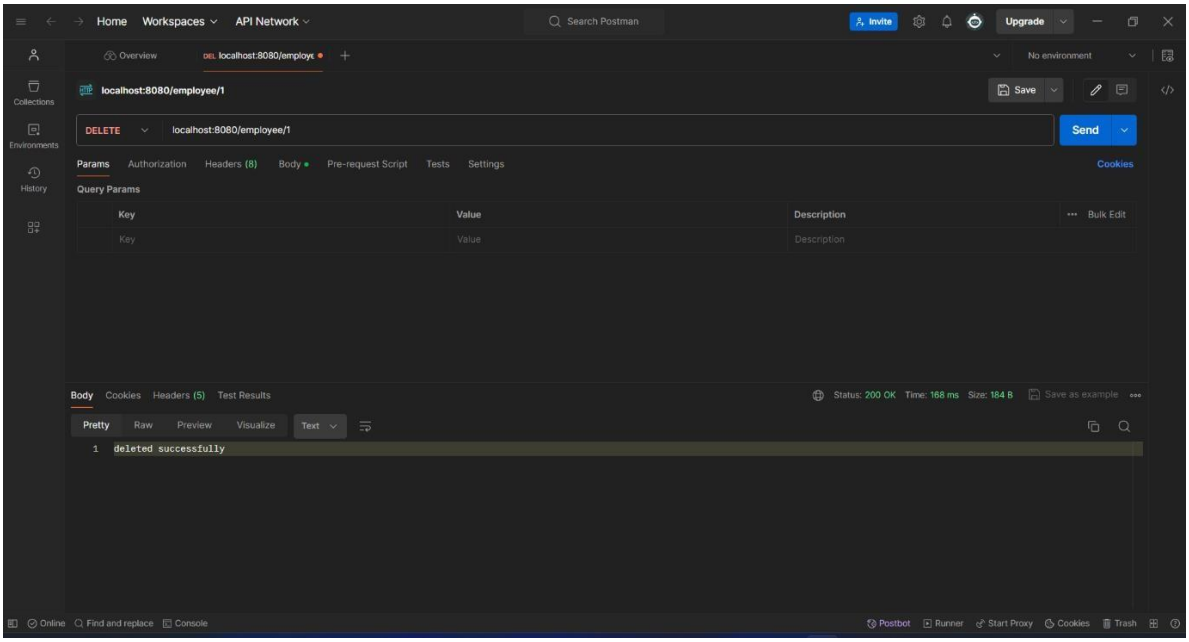
POST:



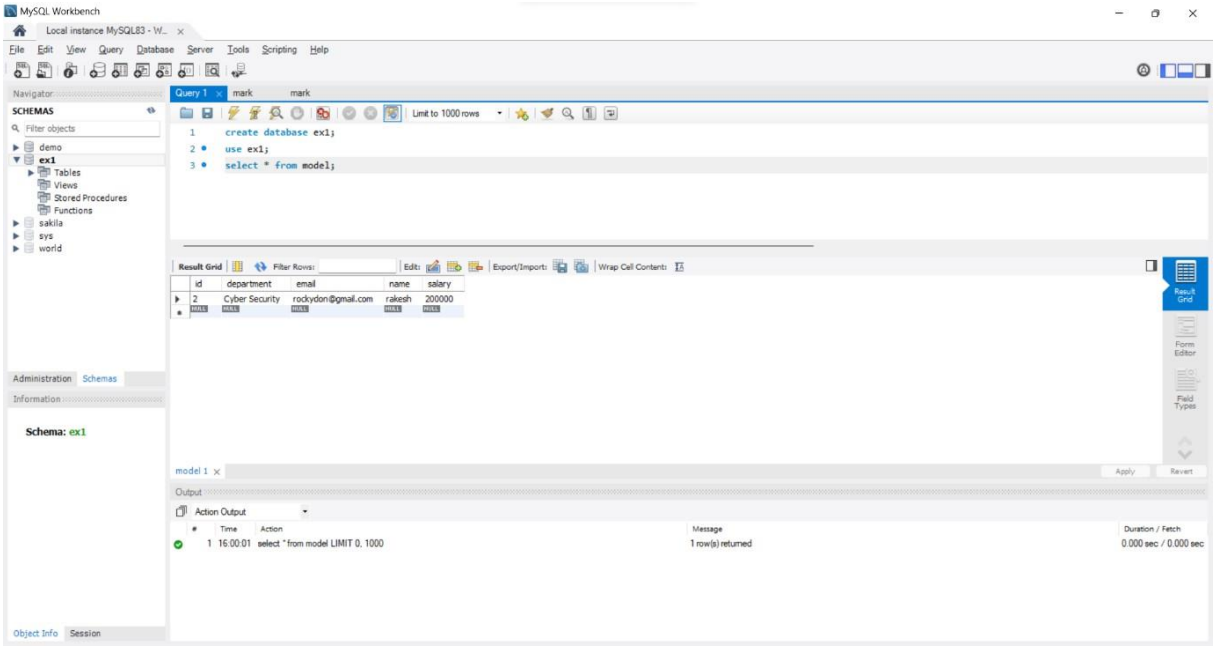
PUT:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Overview
PUT localhost:8080/employe
No environment
localhost:8080/employee/1
Save
PUT
localhost:8080/employee/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"rakesh",
"email":"rockydon@gmail.com",
"department":"Cyber Security",
"salary":200000
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 202 ms
Size: 265 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"name": "rakesh",
"email": "rockydon@gmail.com",
"department": "Cyber Security",
"salary": 200000.0
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash

DELETE:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Overview
DEL localhost:8080/employe
No environment
localhost:8080/employee/1
Save
DELETE
localhost:8080/employee/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 168 ms
Size: 184 B
Save as example
Pretty
Raw
Preview
Visualize
Text
deleted successfully
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash

Database:



Result:

Thus, the application is implemented in Spring boot.

Project 2: Book Library

Management System Date:

Description:

Develop a Spring Boot application to manage a library's book inventory. This system should allow the library to add new books, fetch details about available books, update details on books, and remove books from the inventory.

Entities Fields:

- Book: id (Primary Key, auto-generated), title, author, isbn, publishedYear, genre.

Features:

- Add a Book: POST request to /books with book details in the request body.
- Get All Books: GET request to /books.
- Get Book by ID: GET request to /books/{id}.
- Update a Book: PUT request to /books/{id} with the book details to be updated in the request body.
- Delete a Book: DELETE request to /books/{id}.

Model:

```
package com.example.demo;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="New")
public class Modelclass {
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
private int id;
private String title;
private String author;
private int isbn;
private int publishedYear;
private String genre; public
int getId() {
	return id;
}
public void setId(int id) {
	this.id = id;
}
public String getTitle() {
	return title;
}
public void setTitle(String title) {
	this.title = title;
```

```
}
public String getAuthor() {
	return author;
}
public void setAuthor(String author) {
	this.author = author;
}
 public int getIsbn() {
	 return isbn;
}
public void setIsbn(int isbn) {
	this.isbn = isbn;
}
public int getPublishedYear() {
	return publishedYear;
}
public void setPublishedYear(int publishedYear) {
	this.publishedYear = publishedYear;
}
public String getGenre() {
	return genre;
}
public void setGenre(String genre) {
	this.genre = genre;
}
}
```

Controller:

```
package com.example.demo;

import java.util.List;

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


@RestController
public class Controller {
	@Autowired
	service s;
	@GetMapping("/get")
	public List<Modelclass> getdata(){
		return s.getall();
	}
	@PostMapping("/save")
	public String save(@RequestBody Modelclass m) {
		s.add(m);
		return "added";
	}
	@PutMapping("/update/{id}")
```

```
    public String update(@PathVariable int id, @RequestBody Modelclass model) {
        s.update(id, model);
        return "Updated";
    }

    @DeleteMapping("/delete/{id}")
    public String delete(@PathVariable int id) {
        s.delete(id);
        return "Deleted";
    }
}
```

Repo:

```
package com.example.demo;


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

interface Repo extends JpaRepository<Modelclass,Integer> {

}
```

Service:

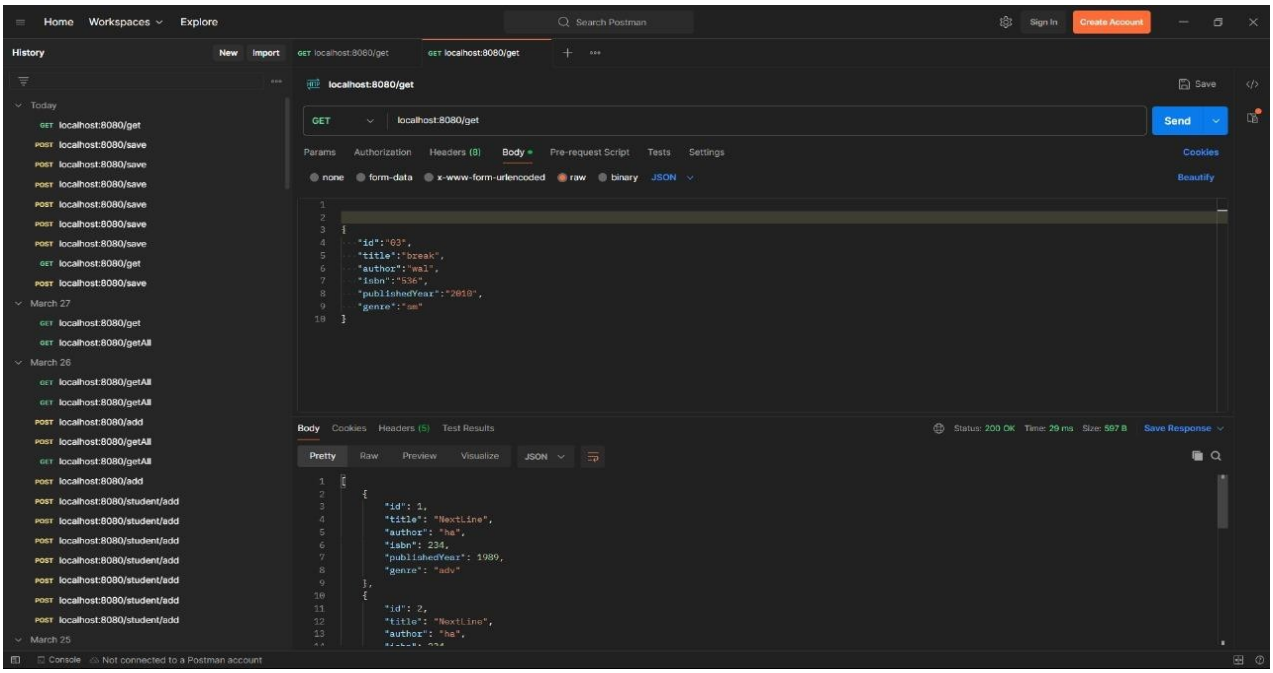
```
package com.example.demo;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

@Service

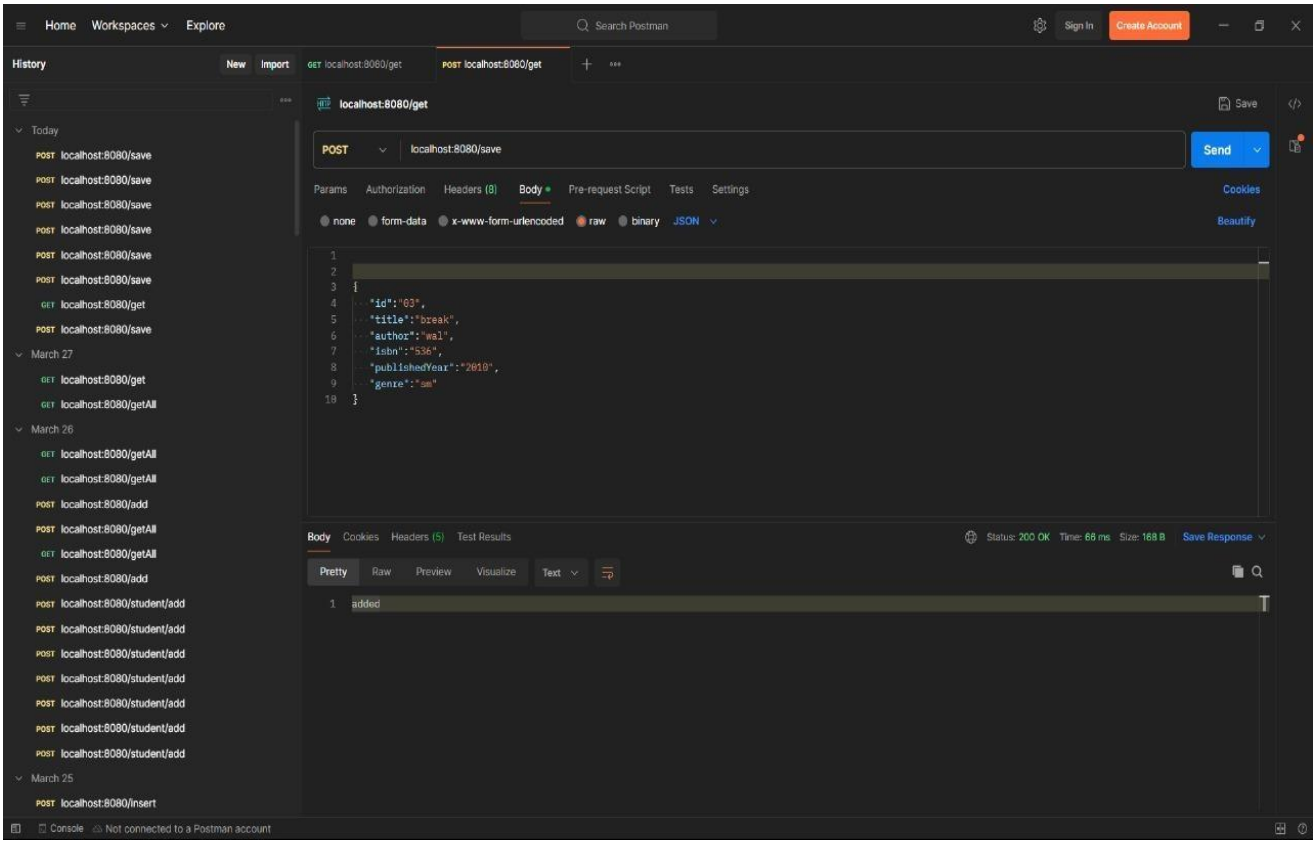
```
public class service {
        @Autowired
        Repo r;
        public String add(Modelclass m) {
                r.save(m);
                return "Added";
        }
        public List<Modelclass> getall(){
                return r.findAll();
        }

        public void update(int id, Modelclass model) { if
      (r.existsById(id)) {
         model.setId(id);
         r.save(model);
      }
   }

   public void delete(int id) {
      r.deleteById(id);
   }
}
```

Application Property: spring.application.name=demo-2
spring.datasource.url= jdbc:mysql://localhost:3306/lab
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database= mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto= update

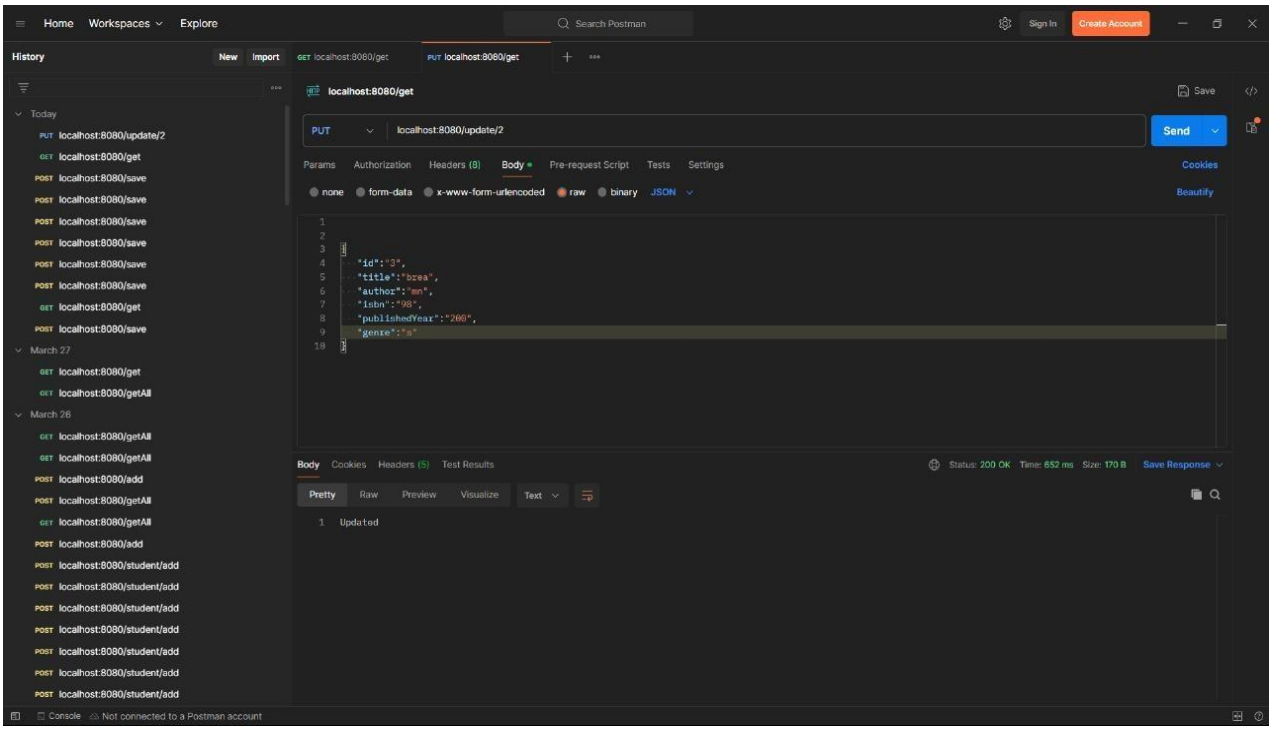
GET :



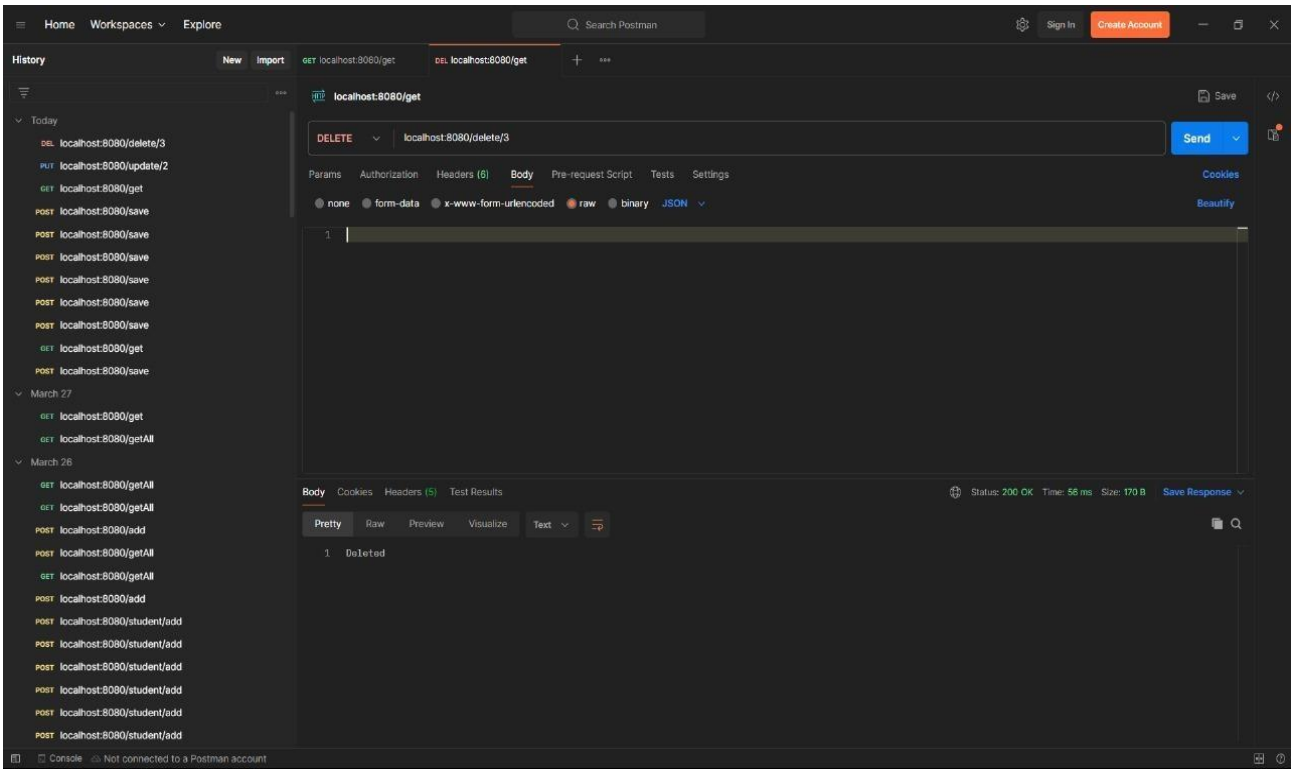
POST:


Home Workspaces Explore
Search Postman
Sign In
Create Account
History
New
Import
GET localhost:8080/get
POST localhost:8080/get
localhost:8080/get
Save
POST
localhost:8080/save
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"id":"03",
"title":"break",
"author":"wal",
"isbn":"536",
"publishedYear":"2010",
"genre":"sm"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 68 ms
Size: 168 B
Save Response
Pretty
Raw
Preview
Visualize
Text
added
Today
POST localhost:8080/save
POST localhost:8080/save
POST localhost:8080/save
POST localhost:8080/save
POST localhost:8080/save
POST localhost:8080/save
GET localhost:8080/get
POST localhost:8080/save
March 27
GET localhost:8080/get
GET localhost:8080/getAll
March 26
GET localhost:8080/getAll
GET localhost:8080/getAll
POST localhost:8080/add
POST localhost:8080/getAll
GET localhost:8080/getAll
POST localhost:8080/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
March 25
POST localhost:8080/insert
Console
Not connected to a Postman account

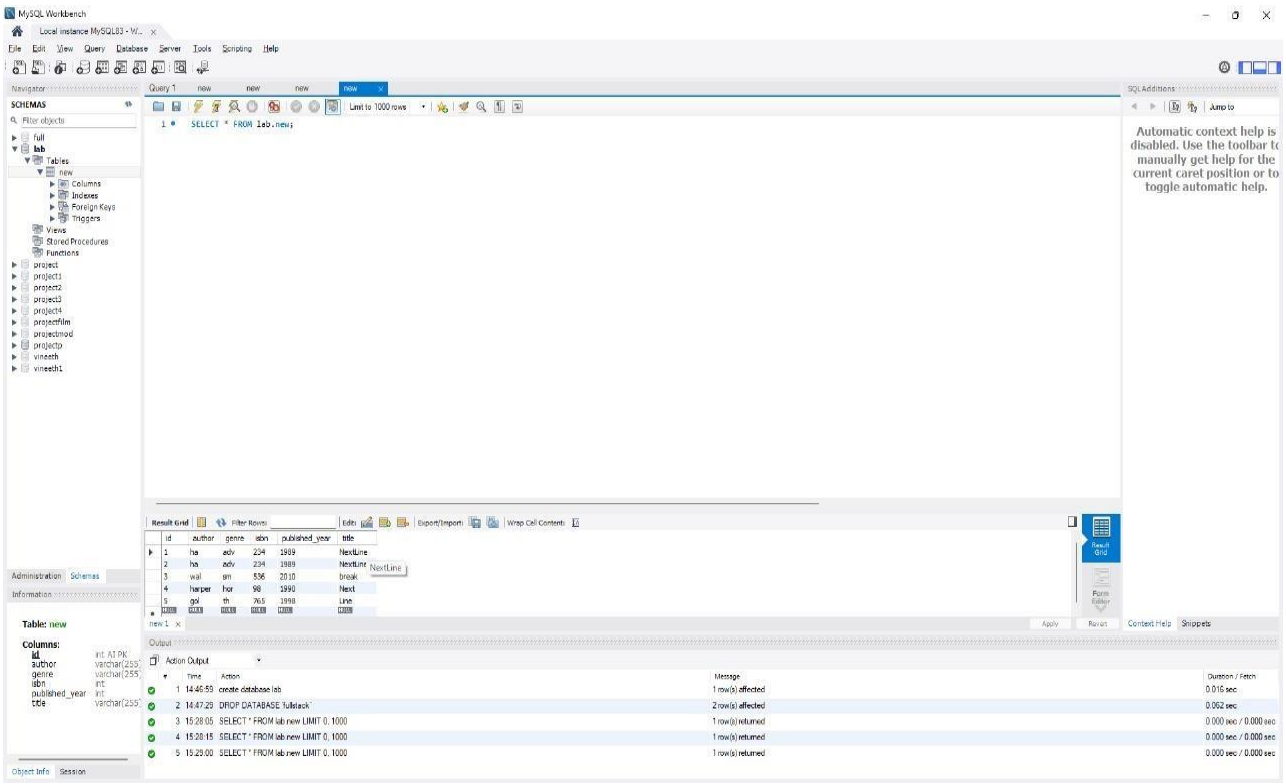
PUT:


Home Workspaces Explore
Search Postman
Sign In
Create Account
History
New
Import
GET localhost:8080/get
PUT localhost:8080/get
localhost:8080/get
Save
PUT
localhost:8080/update/2
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"id":"3",
"title":"brea",
"author":"en",
"isbn":"98",
"publishedYear":"200",
"genre":"s"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 652 ms
Size: 170 B
Save Response
Pretty
Raw
Preview
Visualize
Text
Updated
Today
PUT localhost:8080/update/2
GET localhost:8080/get
POST localhost:8080/save
POST localhost:8080/save
POST localhost:8080/save
POST localhost:8080/save
POST localhost:8080/save
POST localhost:8080/save
GET localhost:8080/get
POST localhost:8080/save
March 27
GET localhost:8080/get
GET localhost:8080/getAll
March 26
GET localhost:8080/getAll
GET localhost:8080/getAll
POST localhost:8080/add
POST localhost:8080/getAll
GET localhost:8080/getAll
POST localhost:8080/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
POST localhost:8080/student/add
Console
Not connected to a Postman account

DELETE:



DATABASE:



Result:

Thus, the application is implemented in Spring boot.

Project 3: Staff Administration

Platform Date:

Description:

Create an interactive web platform designed for staff management, enabling users to perform key actions such as creating, viewing, modifying, and removing staff profiles. Each profile should contain details such as Employee ID, full name, division, and earnings.

Key Features:

- A RESTful API that facilitates CRUD (Create, Read, Update, Delete) functionalities for staff profiles.
- Utilize JPA (Java Persistence API) repositories for seamless interaction with a MySQL database.
- Implement basic authentication mechanisms to safeguard the access to API endpoints.

Controller_Staff:

```
package com.gowtham.OneToOneMany;

import java.util.List;

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



public class Controller_Staff {
	@Autowired
	Serve_Staff s;

	@PostMapping("/create")
	public String create(@RequestBody Model_Staff m) {
		s.add(m);
		return "created sucessfully";
	}



	@GetMapping("/get")
	public List<Model_Staff> display(){
		return s.get_all();
	}

	@PutMapping("/update/{id}")
	public String update_user(@PathVariable int id, @RequestBody Model_Staff m){
		m.setEmp_id(id);
```

```
		return s.update_user(m);
	}


	@DeleteMapping("/delete/{id}")
	public String delete(@PathVariable int id) {
		s.delete(id);
		return "deleted Sucessfully";
	}
}
```

Model_Staff:

```
package com.gowtham.OneToOneMany;

import jakarta.persistence.Entity;
import	jakarta.persistence.GeneratedValue;
import	jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="Staff_detils")
public class Model_Staff {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int emp_id;
	private String name; private
	String department; private
	Long Salary; public int
	getEmp_id() {
		return emp_id;
	}
	public void setEmp_id(int emp_id) {
		this.emp_id = emp_id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getDepartment() {
		return department;
	}
	public void setDepartment(String department) {
		this.department = department;
	}
	public Long getSalary() {
		return Salary;
	}
	public void setSalary(Long salary) {
		Salary = salary;
	}

}
```

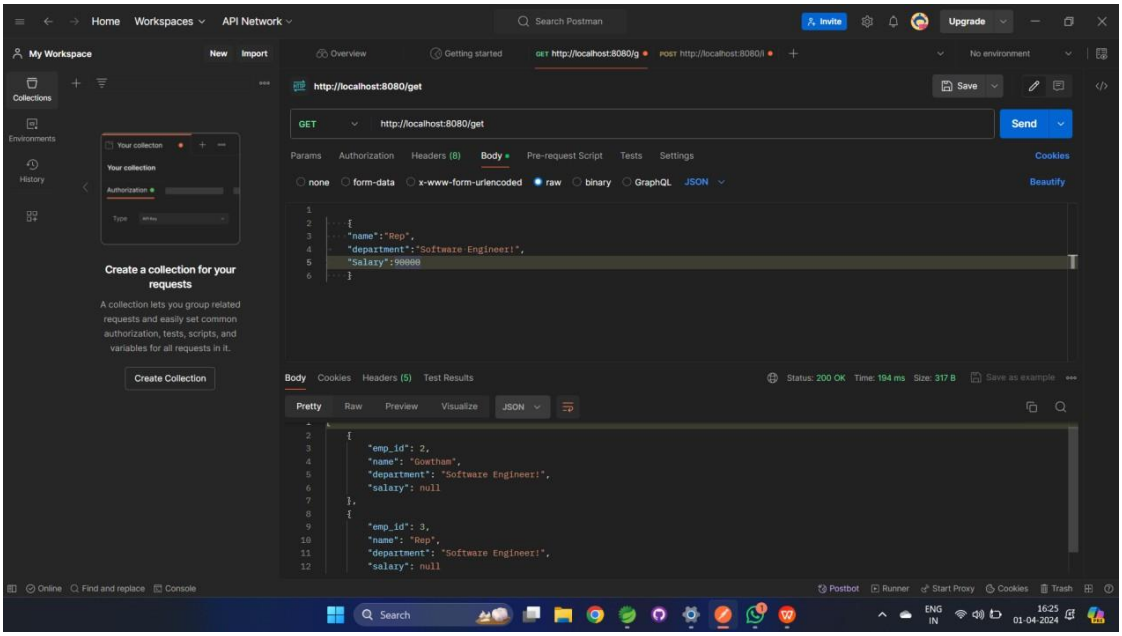
Repo_Admin:

```
package com.gowtham.OneToOneMany;

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

public interface Repo_Admin extends JpaRepository<Model_Staff,Integer>{

}
```

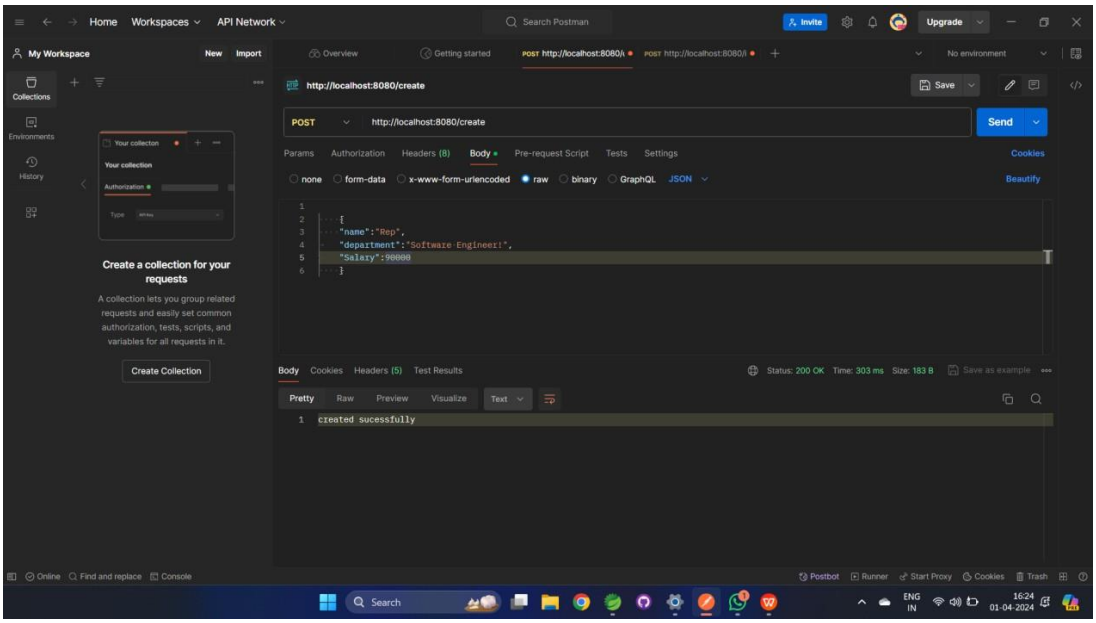
Serve_Staff:

```
package com.gowtham.OneToOneMany;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
@Service
public class Serve_Staff {
	@Autowired
	Repo_Admin r;

	public void add(Model_Staff m)
	{
		r.save(m);
	}
	public List<Model_Staff>get_all(){
		return r.findAll();
		}

	  public void delete(int id) {
		 r.deleteById(id);
	}

	public String update_user(Model_Staff m){
		r.save(m);
		return "Sucess";
		}
}
```

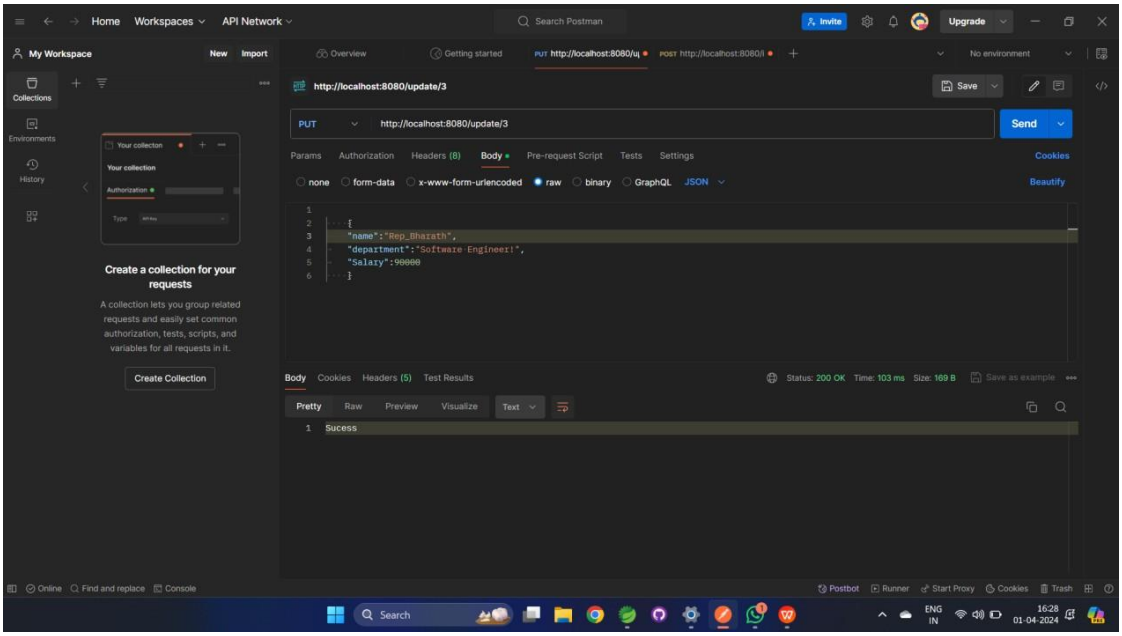
GET:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Getting started
GET http://localhost:8080/g
POST http://localhost:8080/l
No environment
Collections
Environments
History
http://localhost:8080/get
Save
GET
http://localhost:8080/get
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"Rep",
"department":"Software Engineer!",
"Salary":90000
}
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 194 ms
Size: 317 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"emp_id": 2,
"name": "Gowtham",
"department": "Software Engineer!",
"salary": null
},
{
"emp_id": 3,
"name": "Rep",
"department": "Software Engineer!",
"salary": null
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
ENG
IN
16:25
01-04-2024

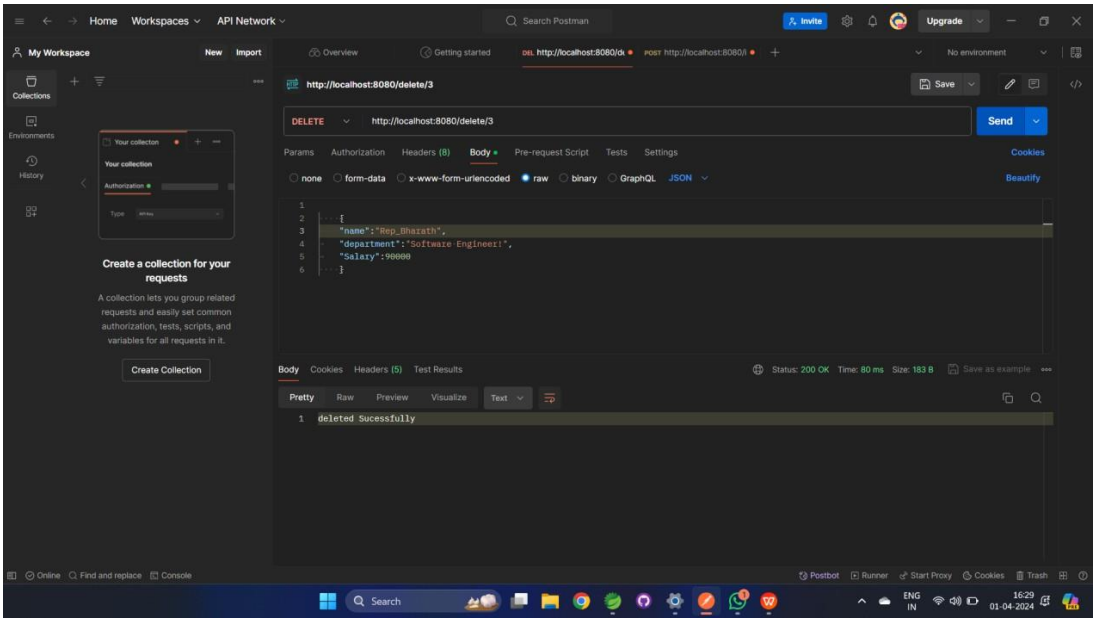
POST:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Getting started
POST http://localhost:8080/l
POST http://localhost:8080/l
No environment
Collections
Environments
History
http://localhost:8080/create
Save
POST
http://localhost:8080/create
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"Rep",
"department":"Software Engineer!",
"Salary":90000
}
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 303 ms
Size: 183 B
Save as example
Pretty
Raw
Preview
Visualize
Text
created sucessfully
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
ENG
IN
16:24
01-04-2024

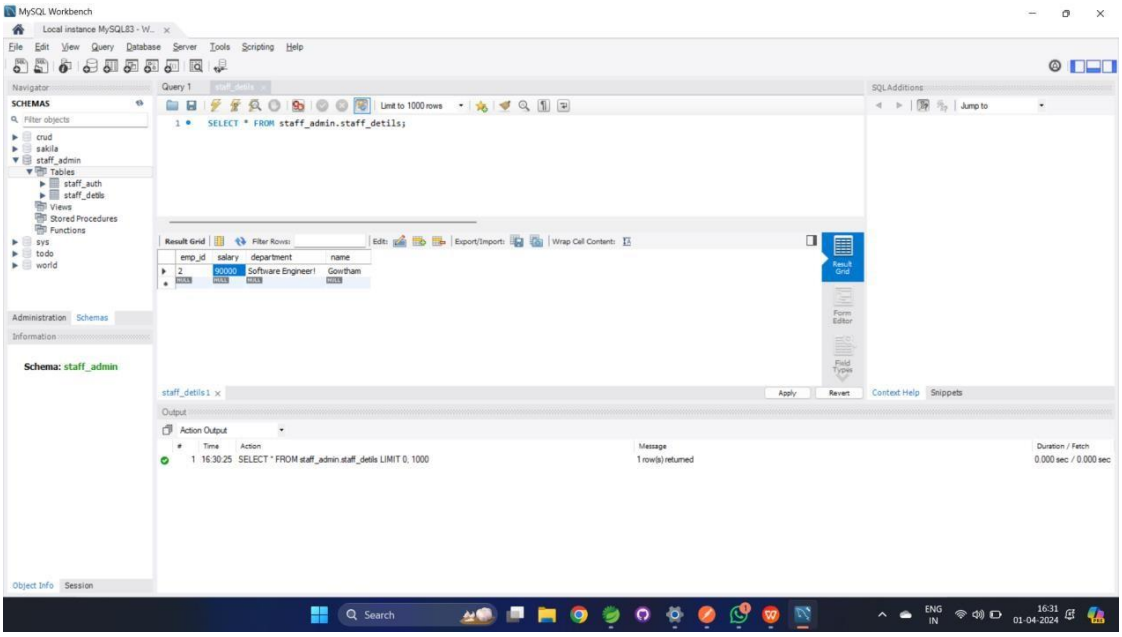
UPDATE:



DELETE:



DB:



Project staff Authentication:

Controller_Auth:

```
package com.gowtham.Staff_Auth;

import org.springframework.beans.factory.annotation.Autowired;
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.RequestBody; import
org.springframework.web.bind.annotation.RestController;

import jakarta.transaction.Transactional;


@RestController
public class Controller_Auth {
	@Autowired
	Serve_Auth s;

	@PostMapping("/register")
	public String create(@RequestBody Model_Auth m) {
		s.add(m);
		return "created sucessfully";
	}
	@Transactional
	@GetMapping("/get/{user_name}/{password}")
	public String get_By(@PathVariable String user_name,@PathVariable String password ) { return
		s.get_By(user_name,password);
```

```
	}
}
```

Model_Auth:

```
package com.gowtham.Staff_Auth;
import jakarta.persistence.Entity;
import	jakarta.persistence.GeneratedValue;
import	jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="Staff_Auth")
public class Model_Auth {
	@Id
	@GeneratedValue(strategy	=	GenerationType.IDENTITY)
	private int id;
	private	String	user;
	private	String	pass;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getUser() {
		return user;
	}
	public void setUser(String user) {
		this.user = user;
	}
	public String getPass() {
		return pass;
	}
	public void setPass(String pass) {
		this.pass= pass;
	}
}
```

Repo_Auth:

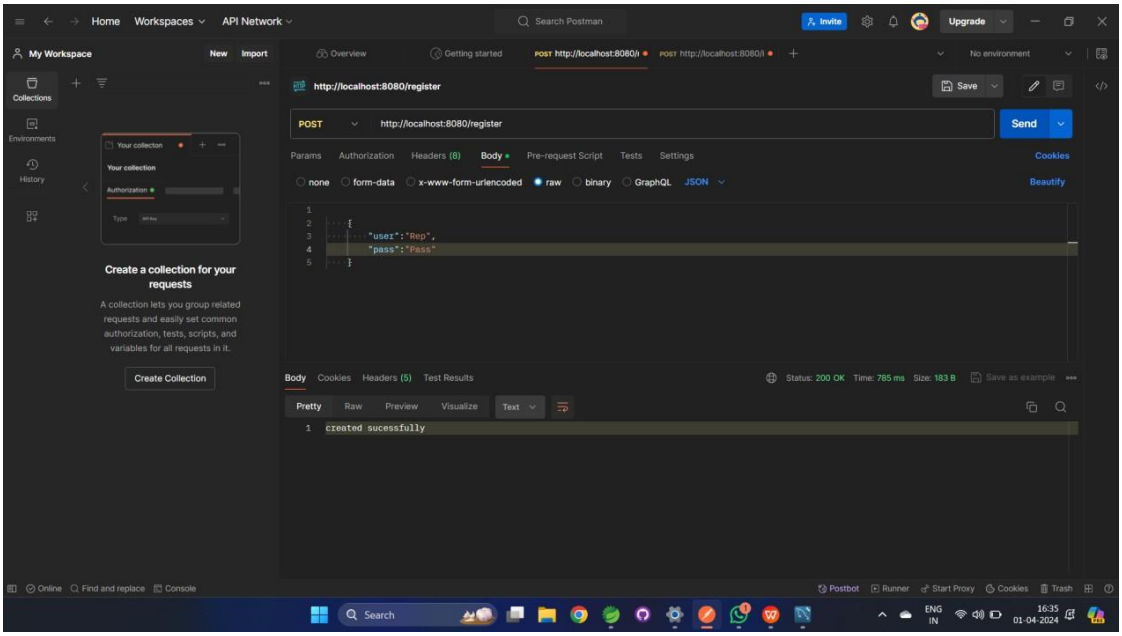
```
package com.gowtham.Staff_Auth;

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

public interface Repo_Auth extends JpaRepository<Model_Auth,Integer>{
```

```
	@Query("Select m from Model_Auth m Where m.user = :user_name")
	public Model_Auth get_By(String user_name);
}
```

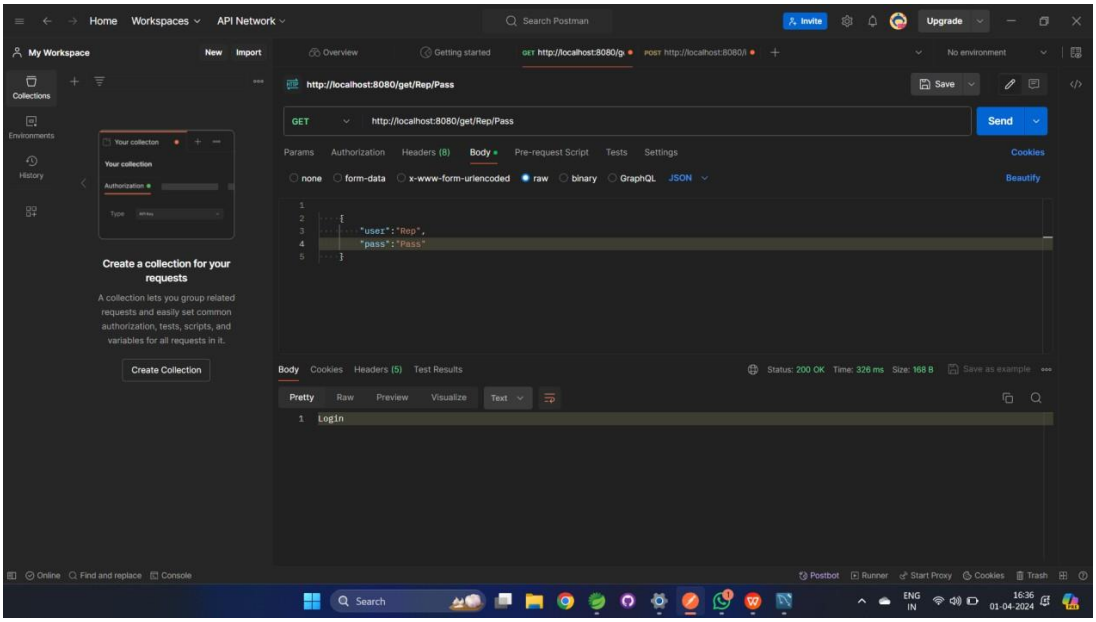
Serve_Auth:

```
package com.gowtham.Staff_Auth;

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

@Service
public class Serve_Auth {
	@Autowired
	Repo_Auth r;

	public void add(Model_Auth m)
	{
		r.save(m);
	}
	public String get_By(String user_name,String password) {
		Model_Auth a=r.get_By(user_name);
		if(a==null) {
			return "user_name does not match";
		}
		else if(a.getPass().equals(password)) {
			return "Login";
		}
		else {
								return "Password is wrong";
		}
			}


}
```

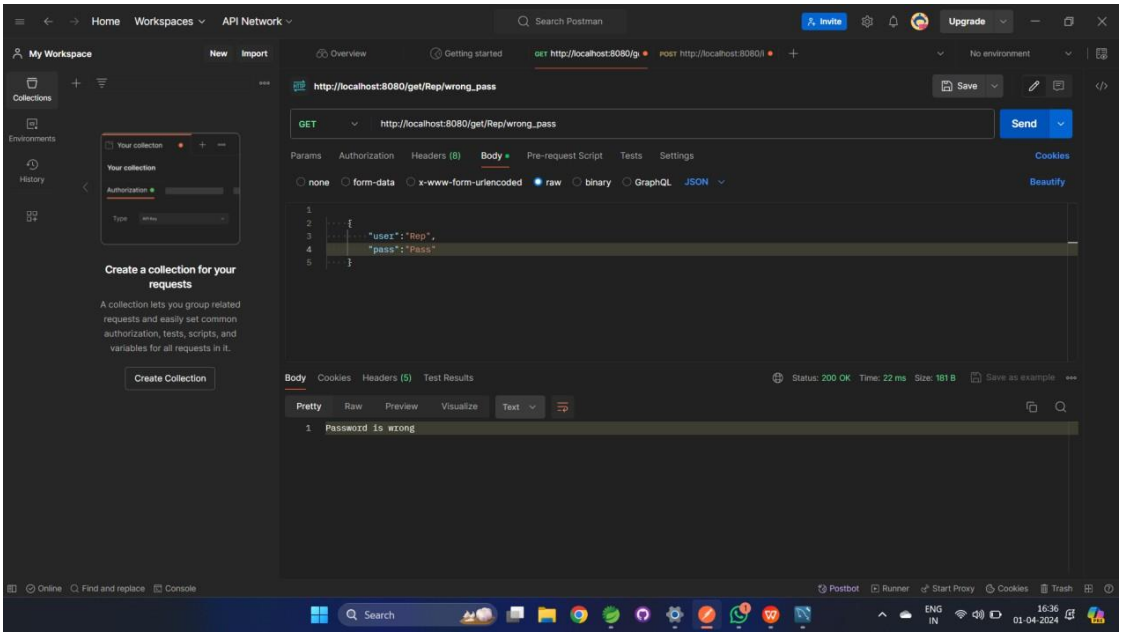
POST:



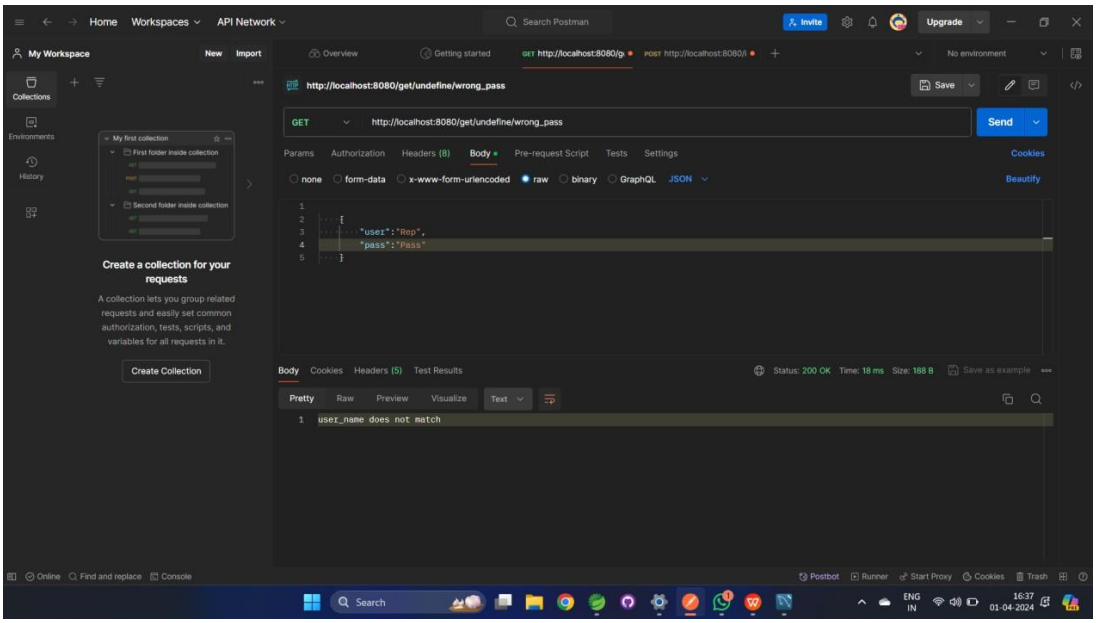
Get (if user name and password is correct):



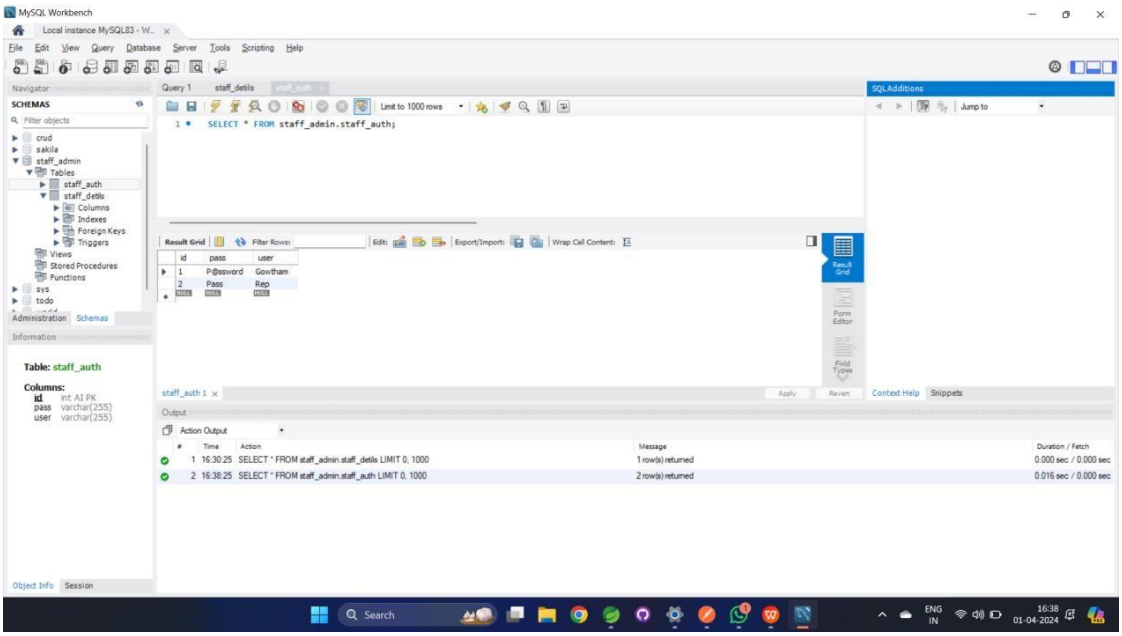
Get (if user name is correct and password is wrong):



Get (if user name and password also wrong):



DB :



Result:

Thus, the application is implemented in Spring boot.

Project 4: Product

Catalog System Date:

Description:

Create a simple product catalog system where users can add new products, view a list of products, update product details, and delete products. Each product should have attributes like ID, name, category, and price.

Key Features:

- CRUD operations for product management.
- Use of JPA to query the database effectively.

Model:

```
package com.example.demo;
import jakarta.persistence.Entity;
import	jakarta.persistence.GeneratedValue;
import	jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="products")
public class Product {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   private String name;
   private String category;
   private double price;
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getCategory() {
		return category;
	}
	public void setCategory(String category) {
		this.category = category;
	}
	public double getPrice() {
		return price;
```

```
	}
	public void setPrice(double price) {
		this.price = price;
	}

}
```

Controller:

```
package com.example.demo;

import java.util.List;

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

@RestController
@RequestMapping("/products")
public class ProductController {
    @Autowired
    private ProductService productService;

    @GetMapping
    public List<Product> getAllProducts() {
        return productService.getAllProducts();
    }

    @GetMapping("/{id}")
    public Product getProductById(@PathVariable Long id) {
        return productService.getProductById(id);
    }

    @PostMapping("/insert")
    public Product addProduct(@RequestBody Product product) {
        return productService.addProduct(product);
    }

    @PutMapping("/{id}")
    public Product updateProduct(@PathVariable Long id, @RequestBody Product updatedProduct) { return
        productService.updateProduct(id, updatedProduct);
    }

    @DeleteMapping("/{id}")
    public void deleteProduct(@PathVariable Long id) {
        productService.deleteProduct(id);
    }
}
```

Repo:

```
package com.example.demo;

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

@Repository
public interface ProductRepository extends JpaRepository<Product, Long> {
}
```

Service

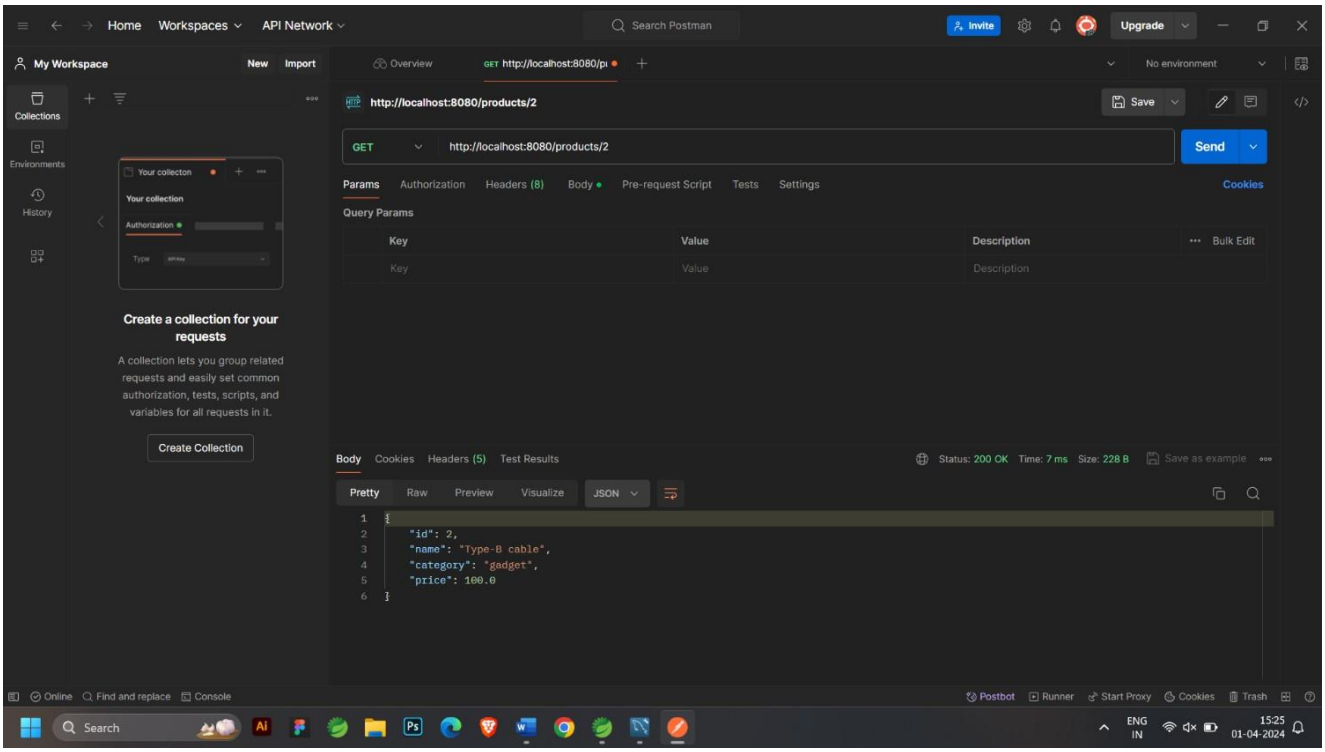
```
package com.example.demo;

import java.util.List;

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

@Service
public class ProductService {
    @Autowired
    private ProductRepository productRepository;

    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }

    public Product getProductById(Long id) {
        return productRepository.findById(id).orElse(null);
    }

    public Product addProduct(Product product) {
        return productRepository.save(product);
    }

    public Product updateProduct(Long id, Product updatedProduct) { if
        (productRepository.existsById(id)) {
            updatedProduct.setId(id);
            return productRepository.save(updatedProduct);
        }
        return null;
    }

    public void deleteProduct(Long id) {
        productRepository.deleteById(id);
    }
}
```

Application Property:

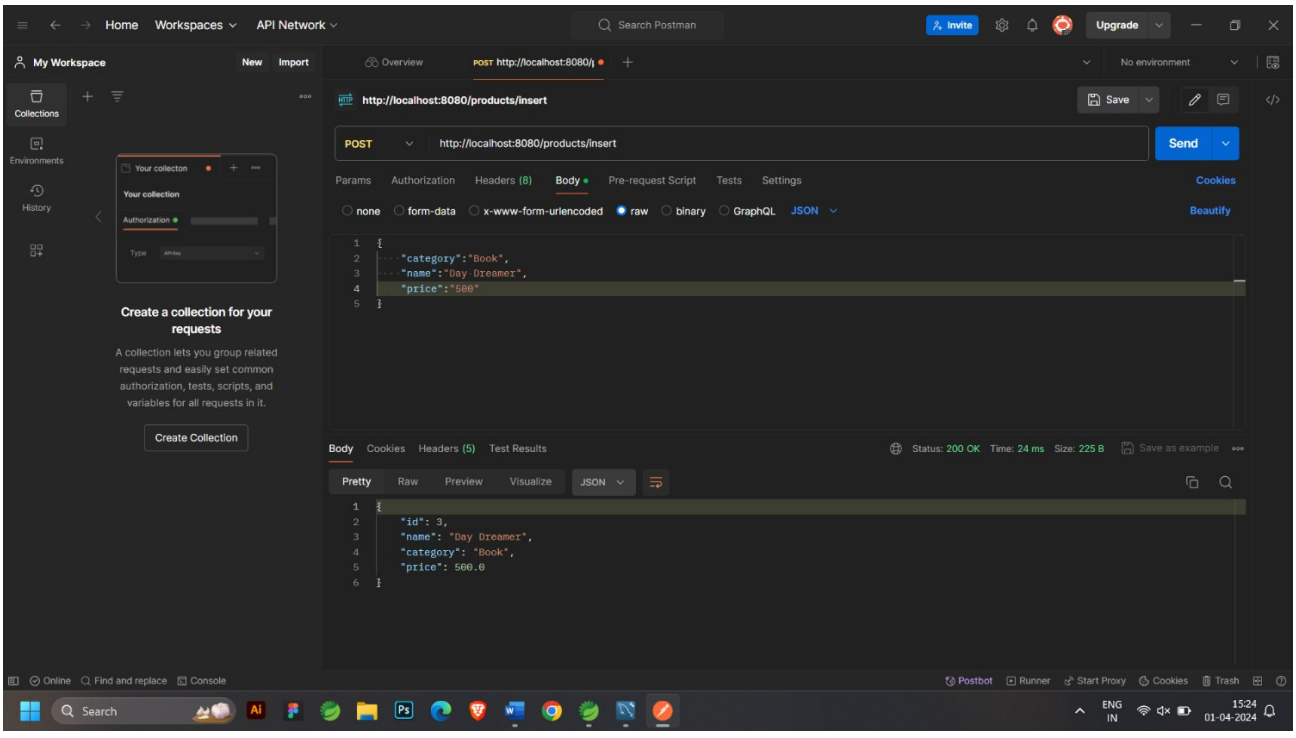
```
spring.application.name=ProductCatalogSystem
spring.datasource.url=jdbc:mysql://localhost:3306/example9
spring.datasource.username=root spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database=mysql
```

spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update

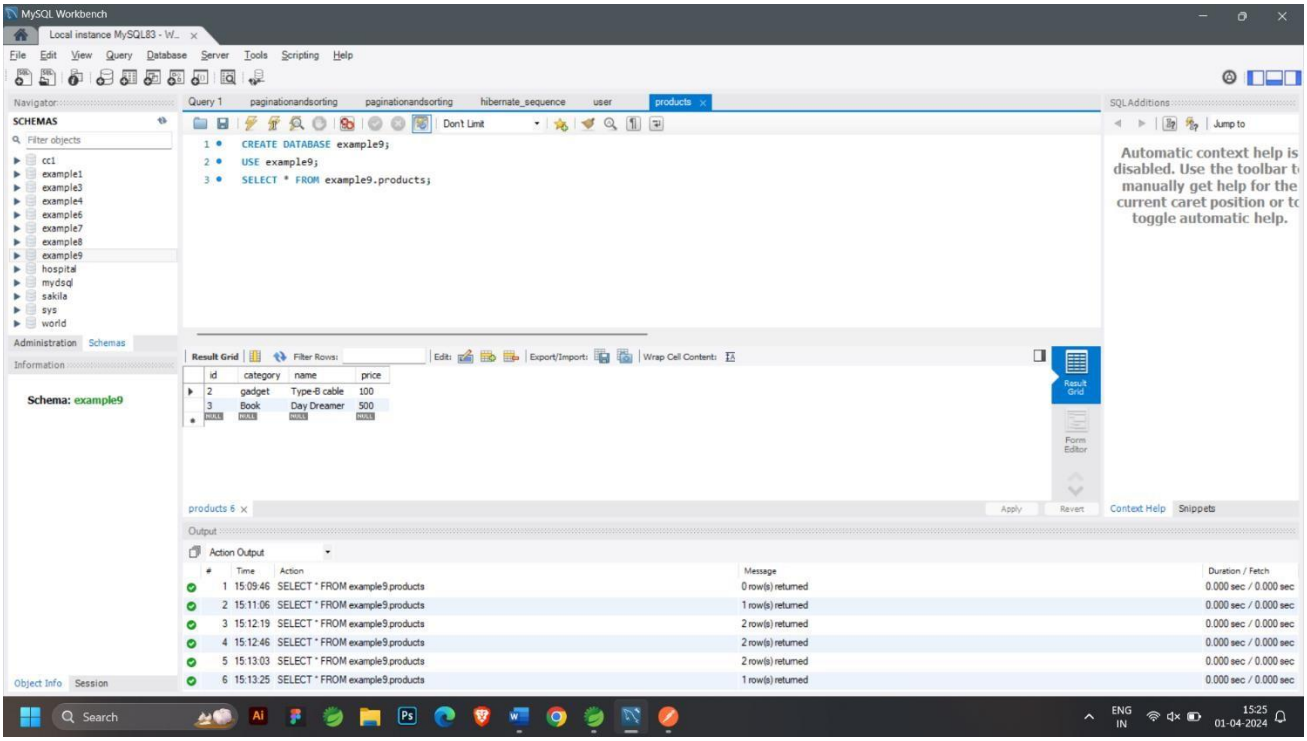
GET :



POST:

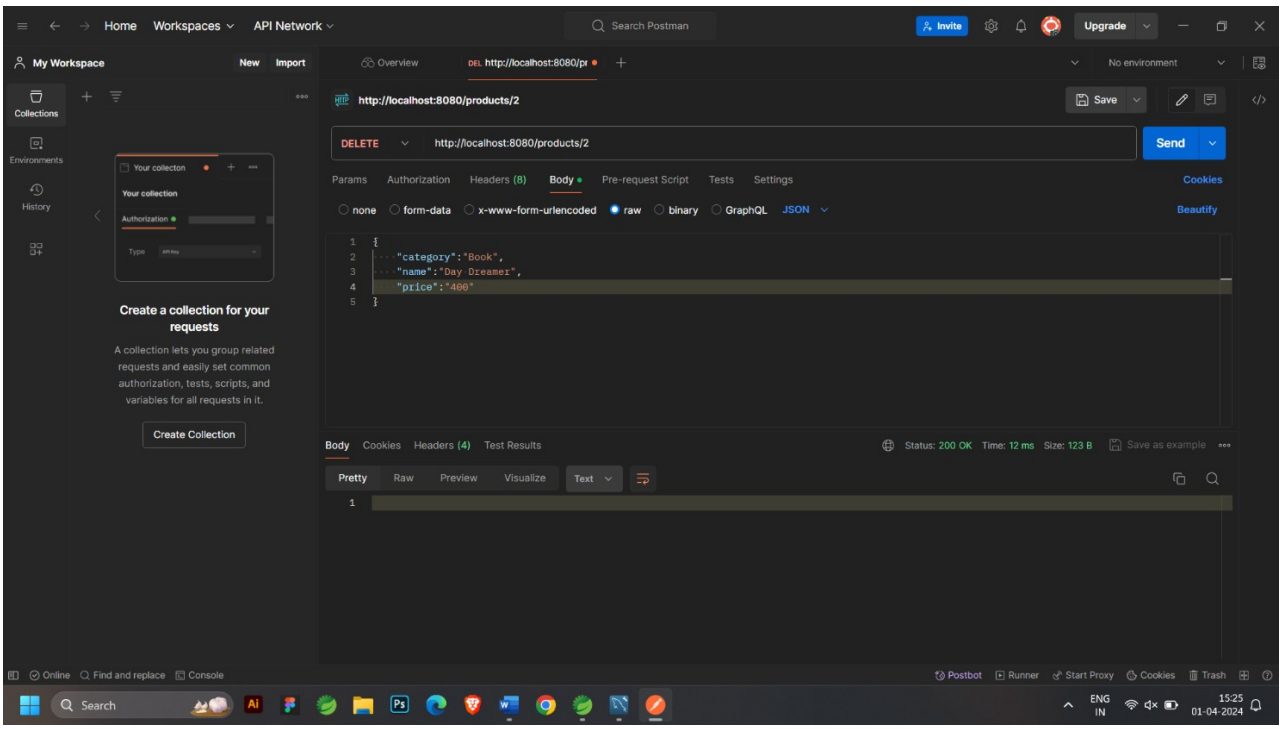


PUT:



DELETE:

DATABASE:



Result:

Thus, the application is implemented in Spring boot.

Project 5: Online Recipe

Repository Date:

Objective: Develop a Spring Boot application to manage a digital collection of recipes. The application should enable users to execute basic CRUD (Create, Read, Update, Delete) operations on recipes, such as adding new recipes, updating existing recipe details, deleting recipes, and retrieving a list of all recipes or a single recipe by its ID.

Entities Fields:

- Recipe (id, name, chef, yearOfCreation, cuisineType)

Endpoints:

- POST /recipes: Add a new recipe
- GET /recipes: Retrieve all recipes
- GET /recipes/{id}: Fetch a recipe by ID
- PUT /recipes/{id}: Update details of a recipe
- DELETE /recipes/{id}: Remove a recipe by ID

Recipe.java: (Entity)

```
package com.example.demo; import
jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import java.util.Date;

@Entity
public class Recipe {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String chef;
    private Date yearOfCreation;
    private String cuisineType;

    public Recipe() {
    }

    public Recipe(String name, String chef, Date yearOfCreation, String cuisineType) { this.name
        = name;
        this.chef = chef;
        this.yearOfCreation = yearOfCreation;
        this.cuisineType = cuisineType;
    }

    public Long getId() {
        return id;
    }
```

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getChef() {
        return chef;
    }

    public void setChef(String chef) {
        this.chef = chef;
    }

    public Date getYearOfCreation() {
        return yearOfCreation;
    }

    public void setYearOfCreation(Date yearOfCreation) {
        this.yearOfCreation = yearOfCreation;
    }

    public String getCuisineType() {
        return cuisineType;
    }

    public void setCuisineType(String cuisineType) {
        this.cuisineType = cuisineType;
    }
}
```

RecipeController.java :

```
package com.example.demo;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/recipes")
public class RecipeController {
    @Autowired
    private RecipeRepository recipeRepository;

    @PostMapping("/post")
    public Recipe addRecipe(@RequestBody Recipe recipe) {
        return recipeRepository.save(recipe);
```

```
    }


    @GetMapping("/{id}")
    public Optional<Recipe> getRecipeById(@PathVariable Long id) {
        return recipeRepository.findById(id);
    }

    @PutMapping("/{id}")
    public Recipe updateRecipe(@PathVariable Long id, @RequestBody Recipe updatedRecipe) {
        updatedRecipe.setId(id); // Ensure ID consistency
        return recipeRepository.save(updatedRecipe);
    }

    @DeleteMapping("/{id}")
    public String deleteRecipe(@PathVariable Long id) {
        recipeRepository.deleteById(id);
        return "Value Deleted" ;
    }
}
```

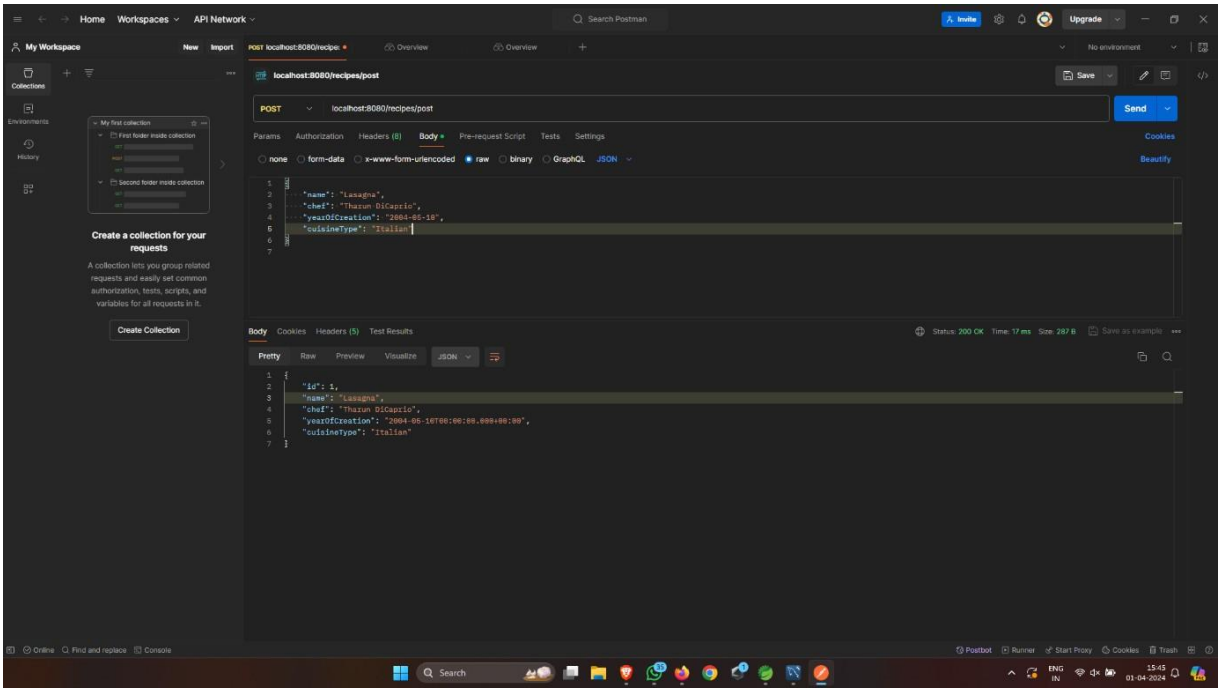
RecipeRepository.java:

```
package com.example.demo;

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

public interface RecipeRepository extends JpaRepository<Recipe, Long> {
}
```

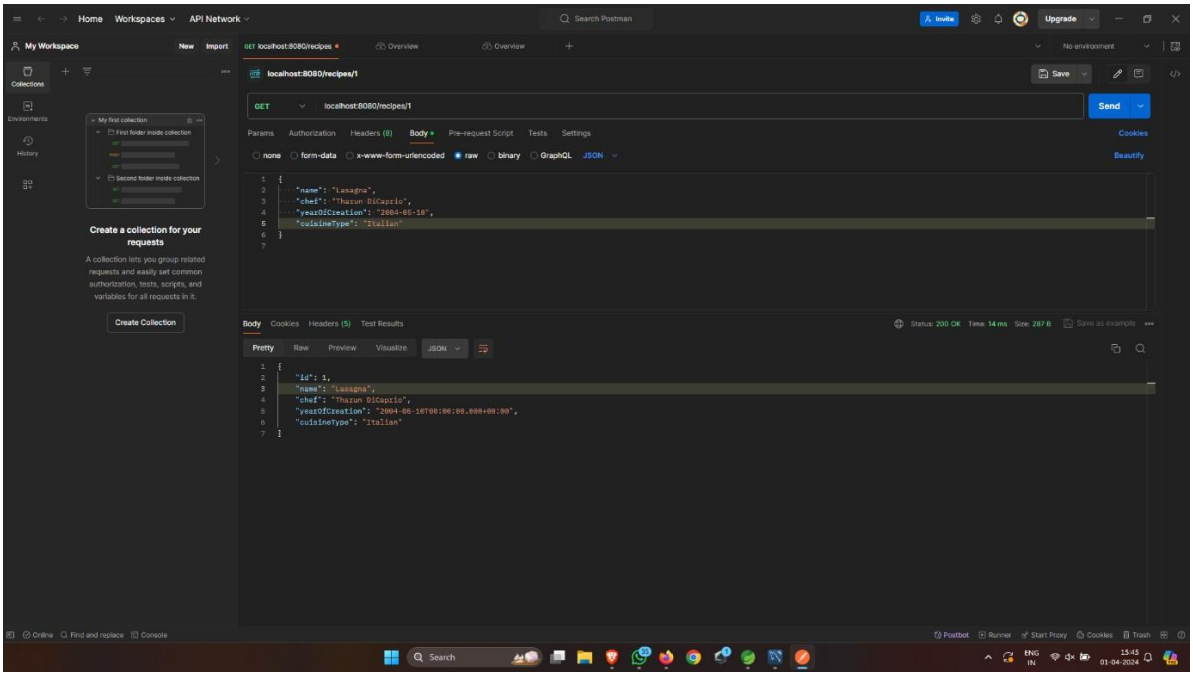
Application.properties:

```
spring.application.name=Project5
spring.datasource.url=jdbc:mysql://localhost:3306/Project5
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.database=mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

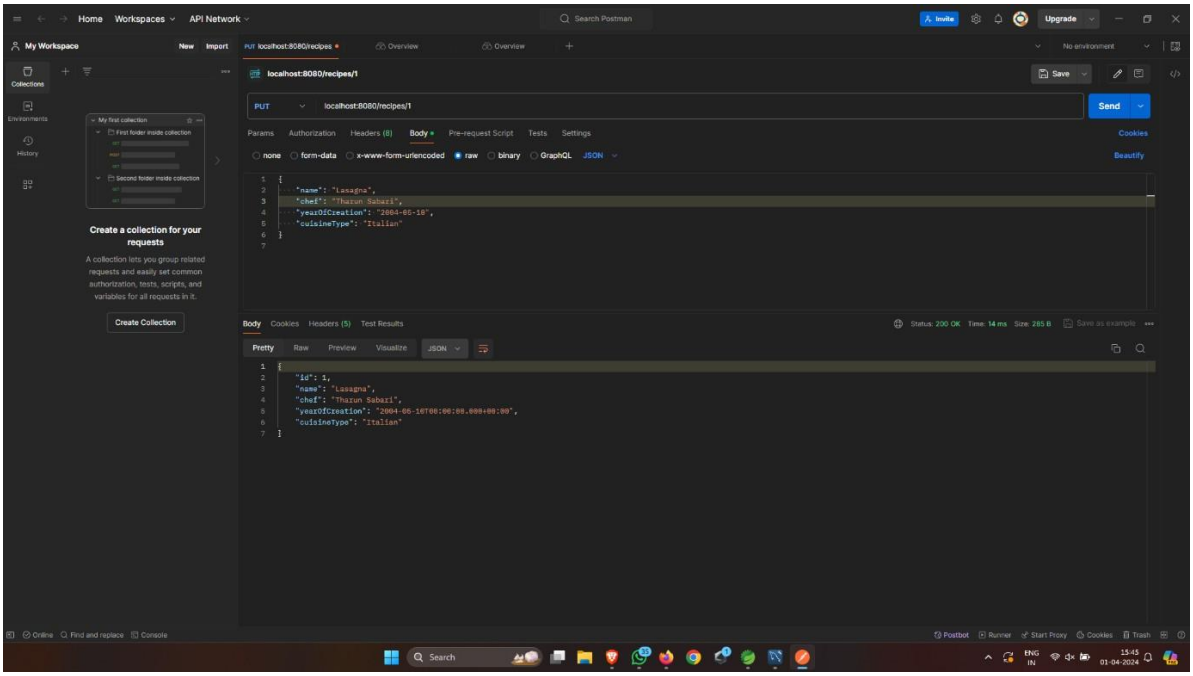
POST:


Home
Workspaces
API Network
Search Postman
Upgrade
My Workspace
New
Import
localhost:8080/recipes/post
POST
localhost:8080/recipes/post
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "Lasagna",
"chef": "Tharun DiCaprio",
"yearOfCreation": "2004-05-18",
"cuisineType": "Italian"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 17 ms
Size: 287 B
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"name": "Lasagna",
"chef": "Tharun DiCaprio",
"yearOfCreation": "2004-05-18T00:00:00.000+00:00",
"cuisineType": "Italian"
}
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection

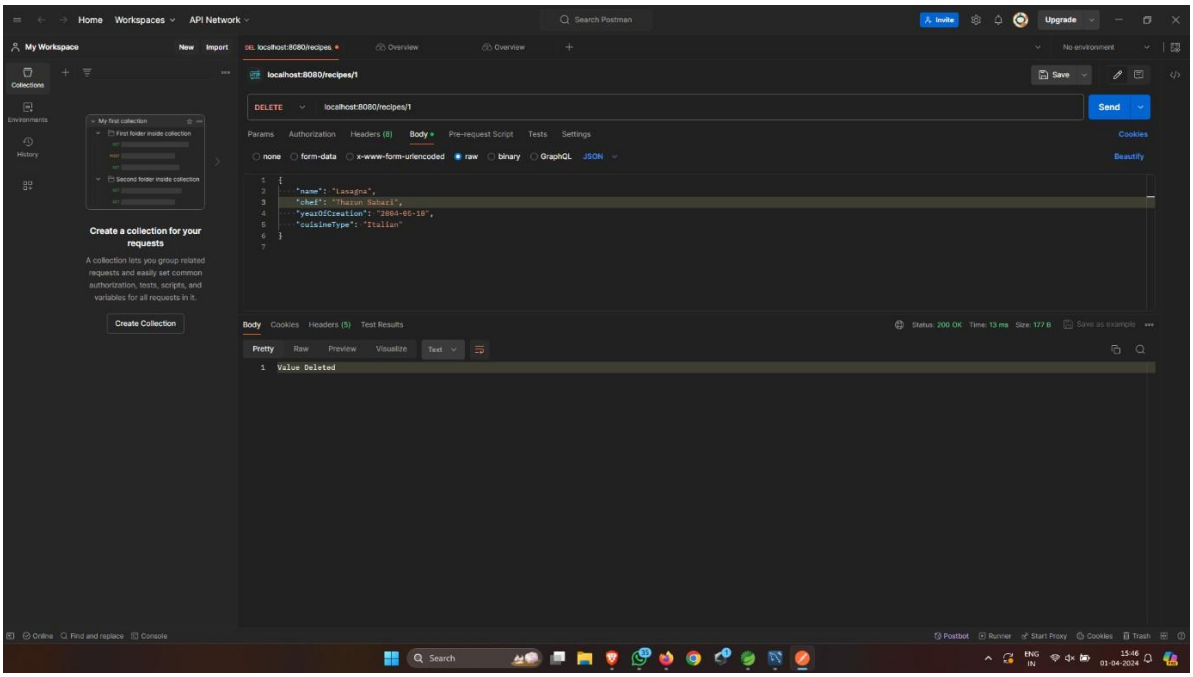
GET:


Home
Workspaces
API Network
Search Postman
Upgrade
My Workspace
New
Import
localhost:8080/recipes/1
GET
localhost:8080/recipes/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "Lasagna",
"chef": "Tharun DiCaprio",
"yearOfCreation": "2004-05-18",
"cuisineType": "Italian"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 14 ms
Size: 287 B
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"name": "Lasagna",
"chef": "Tharun DiCaprio",
"yearOfCreation": "2004-05-18T00:00:00.000+00:00",
"cuisineType": "Italian"
}
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection

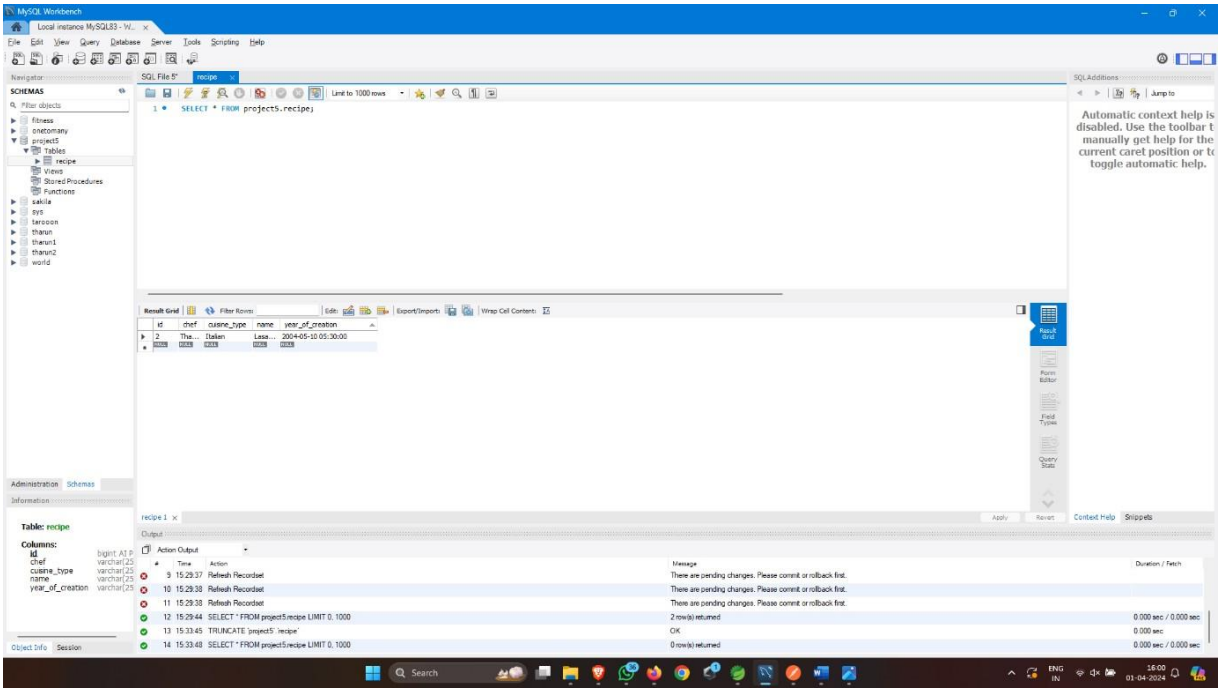
PUT:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
PUT localhost:8080/recipes
Overview
No environment
localhost:8080/recipes/1
Save
Collections
Environments
History
PUT
localhost:8080/recipes/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "Lasagna",
"chef": "Tharun Sabari",
"yearOfCreation": "2004-05-18",
"cuisineType": "Italian"
}
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 14 ms
Size: 285 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"name": "Lasagna",
"chef": "Tharun Sabari",
"yearOfCreation": "2004-05-18T00:00:00.000+00:00",
"cuisineType": "Italian"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
ENG
IN
15:45
01-04-2024

DELETE:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
DEL localhost:8080/recipes
Overview
No environment
localhost:8080/recipes/1
Save
Collections
Environments
History
DELETE
localhost:8080/recipes/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "Lasagna",
"chef": "Tharun Sabari",
"yearOfCreation": "2004-05-18",
"cuisineType": "Italian"
}
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 13 ms
Size: 177 B
Save as example
Pretty
Raw
Preview
Visualize
Text
Value Deleted
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
ENG
IN
15:46
01-04-2024

DATABASE:



Result:

Thus, the application is implemented in Spring boot.

Project 6: Customer Relationship

Management Tool Date:

Objective: Develop a Spring Boot application designed to streamline customer management processes for a growing business. This application should offer capabilities to register new customers, update customer profiles, delete customer records, and fetch detailed information about a single customer or an overview of all customers.

Entities Fields:

Customer (id, firstName, lastName, email, loyaltyPoints,address)

Key Features:

- POST /customers: Register a new customer
- GET /customers: Retrieve a list of all customers
- GET /customers/{id}: Access details of a specific customer by ID
- PUT /customers/{id}: Update a customer's profile
- DELETE /customers/{id}: Delete a customer record

Additional Features to Consider:

- Implement pagination and sorting for displaying customers
- Ensure robust validation for all data inputs

Model Class:

```
package com.example.Exercise6;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
@Entity
@Table(name="Customer")

public class Model {
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private int id;
	private String firstname;
	private String lastname;
	private String email;
	private int loyaltyPoints;
	private String address;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
```

```
	}
	public String getFirstname() {
		return firstname;
	}
	public void setFirstname(String firstname) {
		this.firstname = firstname;
	}
	public String getLastname() {
		return lastname;
	}
	public void setLastname(String lastname) { this.lastname
		= lastname;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public int getLoyaltyPoints() {
		return loyaltyPoints;
	}
	public void setLoyaltyPoints(int loyaltyPoints) {
		this.loyaltyPoints = loyaltyPoints;
	}
	public String getAddress() {
		return address;
	}
	public void setAddress(String address) {
		this.address = address;
	}


}
```

Controller class:

```
package com.example.Exercise6;

import java.util.List;

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;

@RestController
@RequestMapping("/customers")
```

```
public class Controller {
    @Autowired
    private serv customerService;

    @PostMapping("/create")
    public ResponseEntity<Model> createCustomer(@RequestBody Model customer) {
        Model createdCustomer = customerService.createCustomer(customer);
        return new ResponseEntity<>(createdCustomer, HttpStatus.CREATED);
    }

    @GetMapping("/get")
    public ResponseEntity<List<Model>> getAllCustomers() {
        List<Model> customers = customerService.getAllCustomers();
        return new ResponseEntity<>(customers, HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Model> getCustomerById(@PathVariable int id) { Model
        customer = customerService.getCustomerById(id);
        if (customer != null) {
            return new ResponseEntity<>(customer, HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    @PutMapping("/{id}")
    public ResponseEntity<Model> updateCustomer(@PathVariable int id, @RequestBody Model updatedCustomer) {
        Model customer = customerService.updateCustomer(id, updatedCustomer); if
        (customer != null) {
            return new ResponseEntity<>(customer, HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteCustomer(@PathVariable int id) {
        customerService.deleteCustomer(id);
        return new ResponseEntity<>(HttpStatus.NO_CONTENT);
    }
}
```

Service Class:

```
package com.example.Exercise6;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
@Service
public class serv {
        @Autowired
        Repo customerRepo;
        public Model createCustomer(Model customer) { return
                customerRepo.save(customer);
        }
        public List<Model> getAllCustomers(){
                return customerRepo.findAll();
        }

        public Model getCustomerById(int id) { return
    customerRepo.findById(id).orElse(null);
  }
   public Model updateCustomer(int id, Model updatedCustomer) {
        Model existingCustomer = customerRepo.findById(id).orElseGet(null); if(existingCustomer
        != null) {
                return customerRepo.save(existingCustomer);
        }
        return null;
   }
   public void deleteCustomer(int id) {
        customerRepo.deleteById(id);
   }
}
```

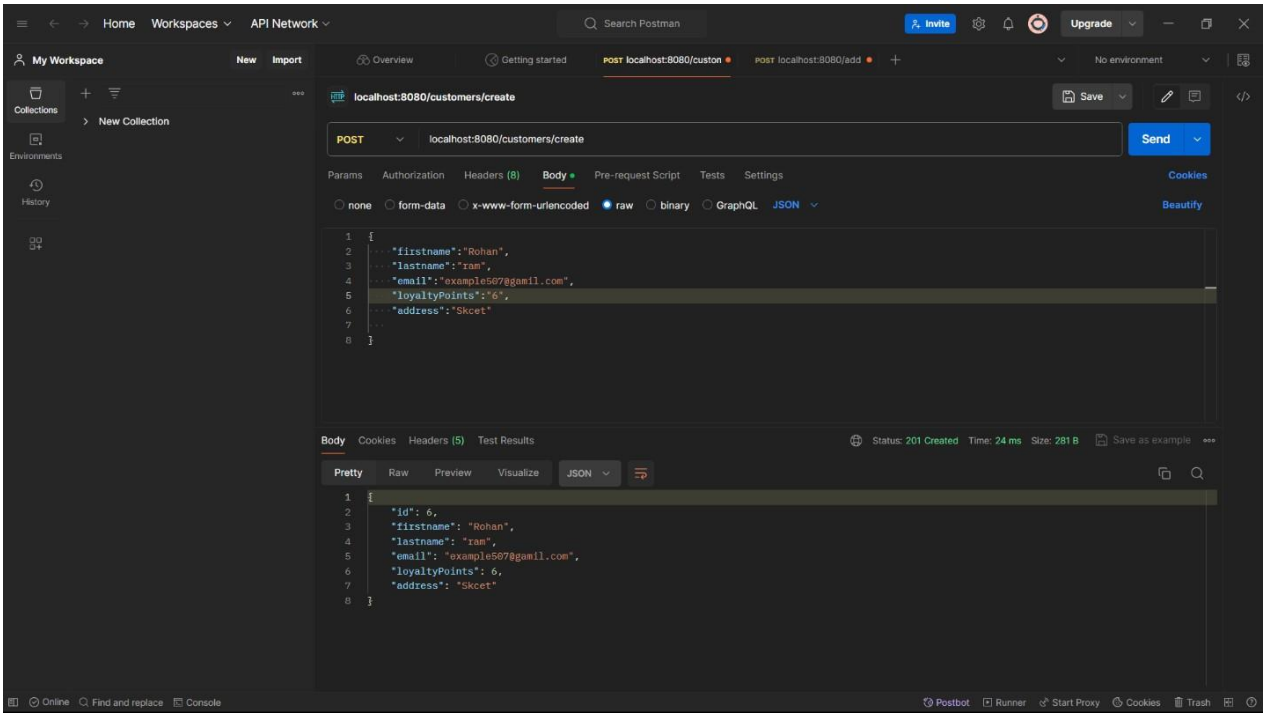
Repo Interface:

```
package com.example.Exercise6;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository; @Repository
public interface Repo extends JpaRepository<Model,Integer>{

}
```

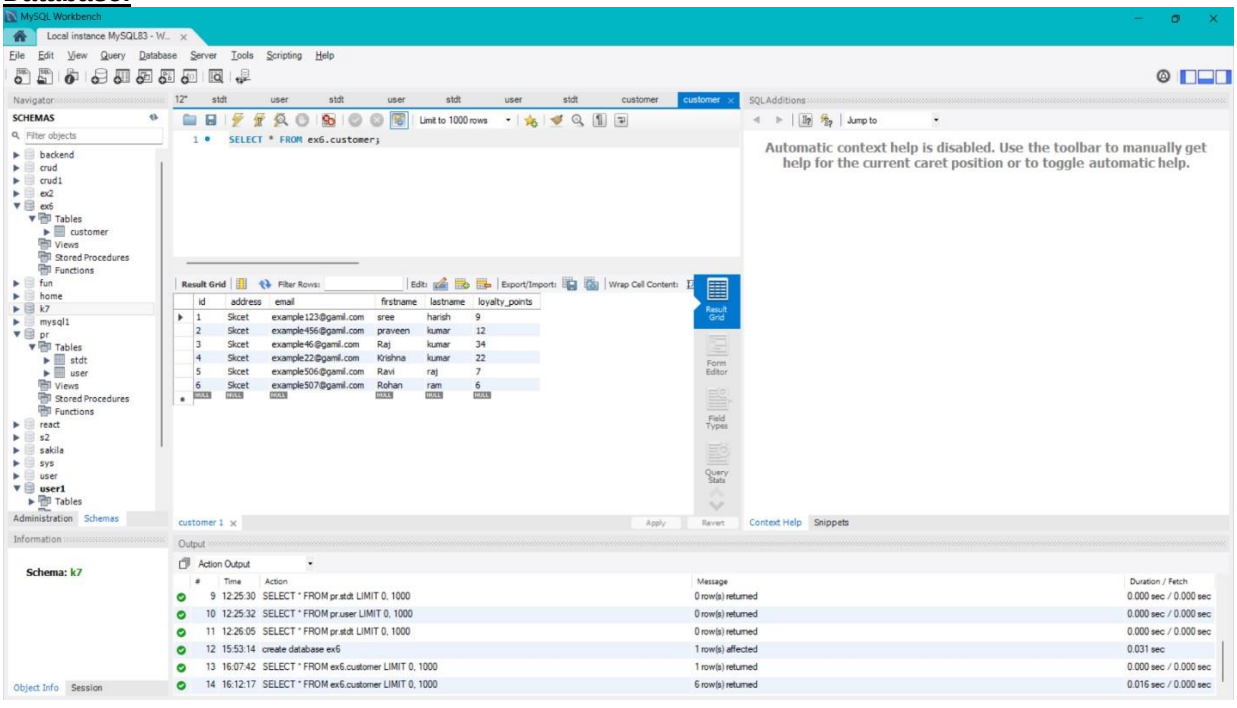
application.properties:

```
spring.application.name=Exercise6 spring.datasource.url=
jdbc:mysql://localhost:3306/ex6
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database= mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto= update
```

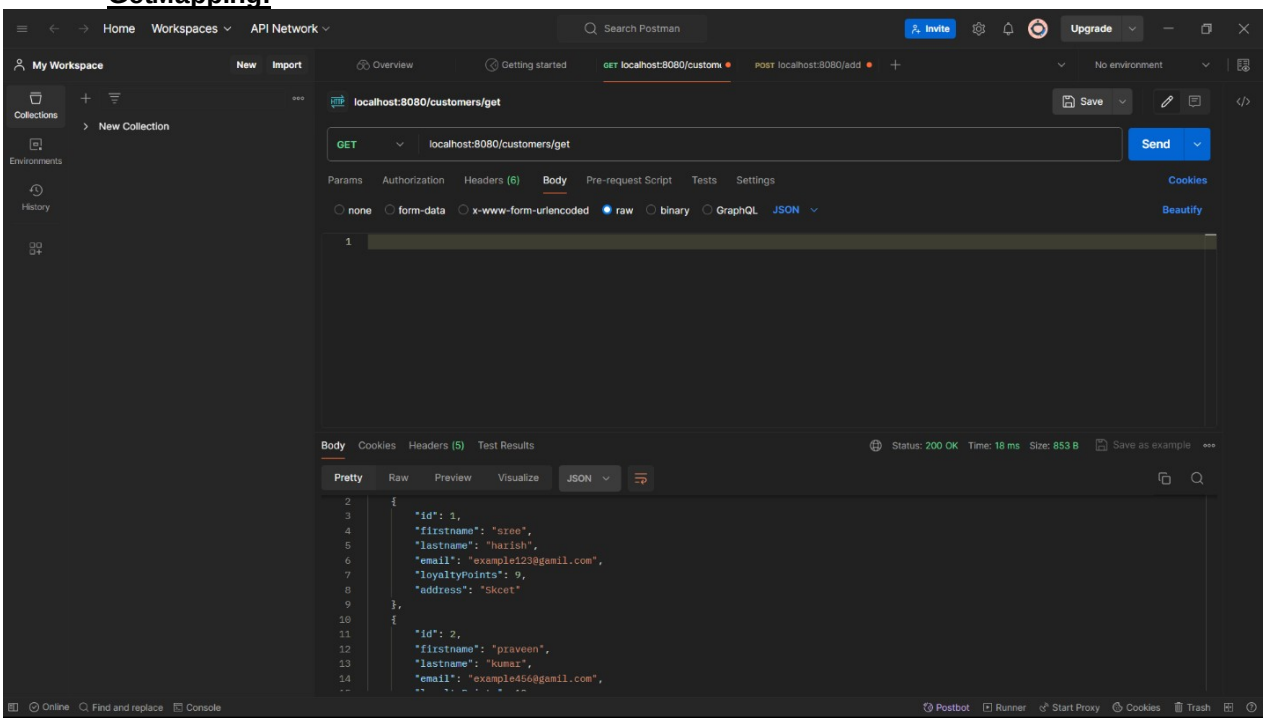
PostMapping:



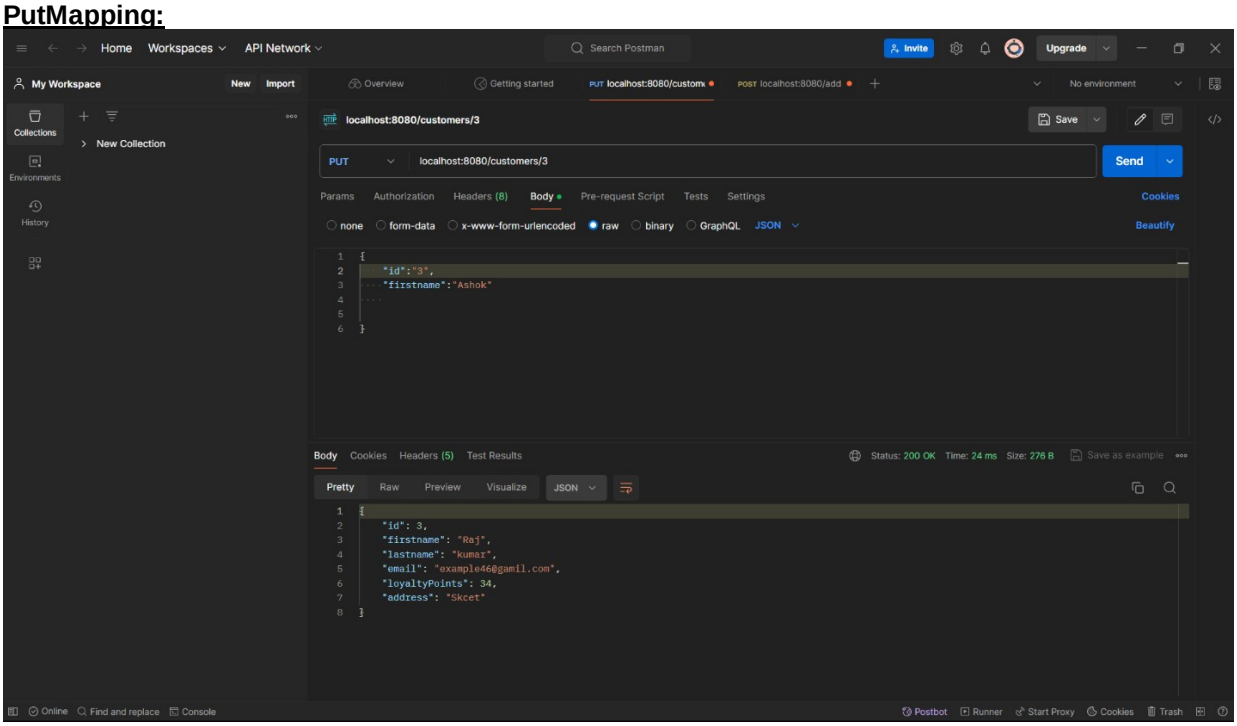
Database:



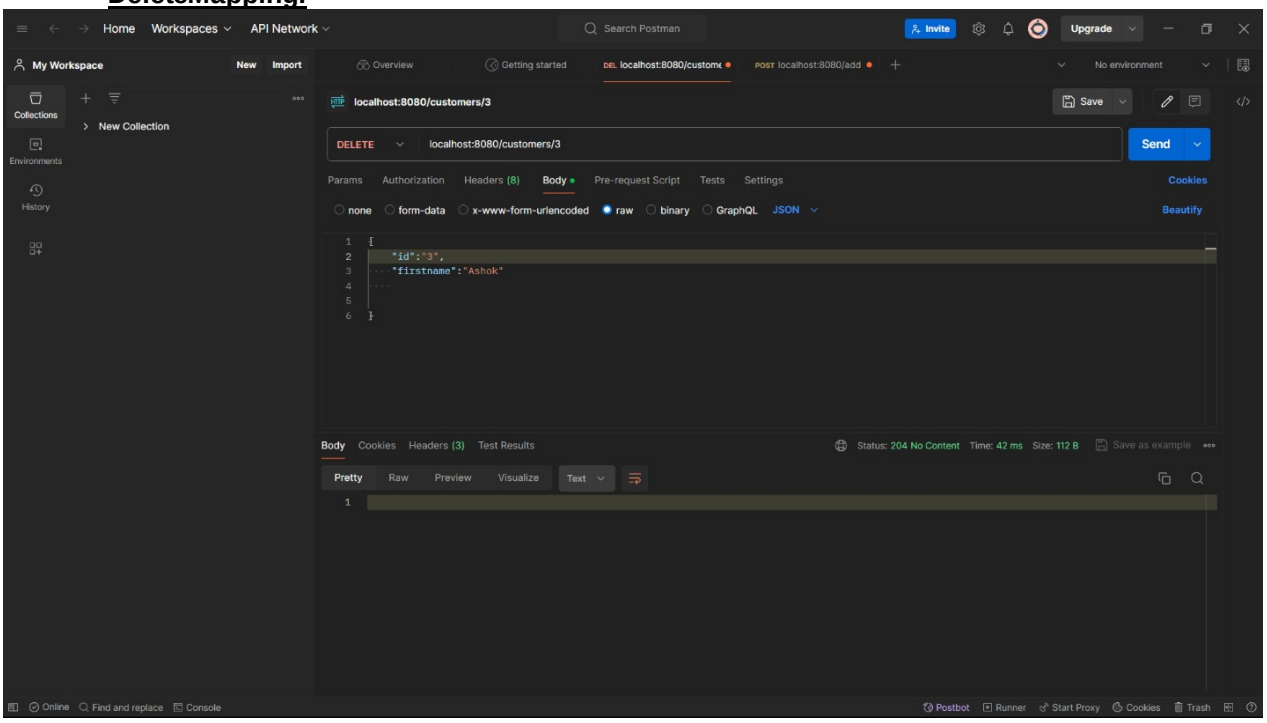
GetMapping:



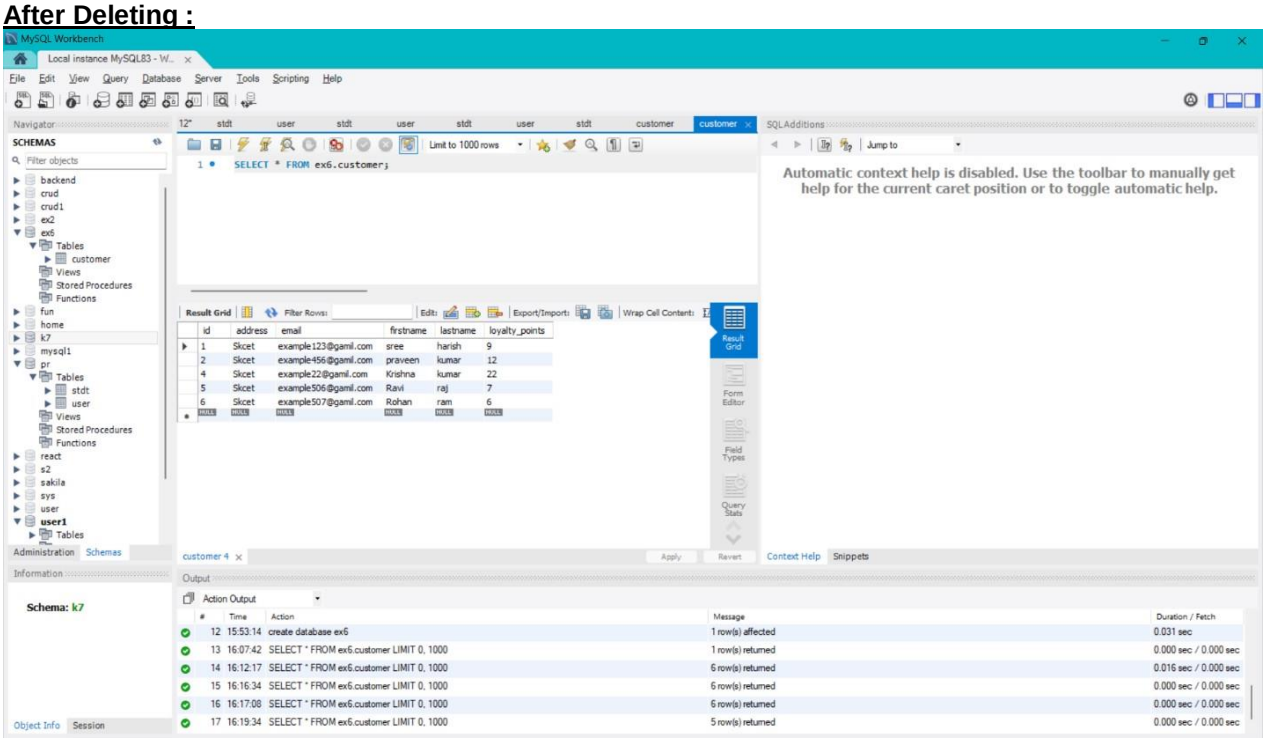
PutMapping:



DeleteMapping:



After Deleting :



Result:

Thus, the application is implemented in Spring boot.

Project 7: Campus Club

Management System Date:

Objective: Develop a web-based application to manage member information within various clubs at a college campus.

Entities and Relationships:

Member - Represents the members in various clubs on campus.

Attributes: id (Primary Key), name, email, club (Many-to-One relationship with Club entity)

Club - Represents the different clubs available on campus.

Attributes: id (Primary Key), name, members (One-to-Many relationship with Member entity)

Relational Mapping:

The Member entity has a Many-to-One relationship with the Club entity, indicating that multiple members can be part of one club.

Basic Operations:

- Create, Read, Update, and Delete members.
- Create, Read, Update, and Delete clubs.
- Enroll members into a club.

Member.java:

```
package com.example.pipe; import
jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
@Entity
public class Member {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String name;
   private String email;
   @ManyToOne
   @JoinColumn(name = "club_id")
   private Club club;
        public Club getClub() {
        return club;
        }
        public Long getId() {
                return id;
        }
        public void setId(Long id) {
```

```
                this.id = id;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public String getEmail() {
                return email;
        }
        public void setEmail(String email) {
                this.email = email;
        }
        public void setClub(Club club) {
                this.club = club;
        }
    // Constructors, getters, and setters
}
```

Club.Java:

```
package com.example.pipe;

import java.util.ArrayList;
import java.util.List;

import     jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import     jakarta.persistence.GeneratedValue;
import     jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.OneToMany;

@Entity
public class Club {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    @OneToMany(mappedBy = "club", cascade = CascadeType.ALL)
    private List<Member> members = new ArrayList<>();
        public Long getId() {
                return id;
        }
        public void setId(Long id) {
                this.id = id;
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public List<Member> getMembers() {
                return members;
        }
```

```
        public void setMembers(List<Member> members) {
                this.members = members;
        }}
```

MemberService.java:

```
package com.example.pipe;

import java.util.List;

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

@Service
public class MemberService {
    @Autowired
    private MemberRepository memberRepository;

    public List<Member> getAllMembers() {
        return memberRepository.findAll();
    }

    // Other CRUD operations for members
}
```

ClubService.java:

```
package com.example.pipe;

import java.util.List;

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

@Service
public class ClubService {
    @Autowired
    private ClubRepository clubRepository;

    public List<Club> getAllClubs() {
        return clubRepository.findAll();
    }

    // Other CRUD operations for clubs

    public void enrollMember(Long clubId, Member member) {
        Club club = clubRepository.findById(clubId).orElseThrow(() -> new RuntimeException("Club not found"));
        member.setClub(club);
        club.getMembers().add(member);
        clubRepository.save(club);
    }
}
```

MemberController.java:

```
package com.example.pipe;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.RequestMapping; import
org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/api/members")
public class MemberController {
    @Autowired
    private MemberService memberService;
    @GetMapping
    public List<Member> getAllMembers() {
        return memberService.getAllMembers();
    }
    // Other CRUD endpoints for members
}
```

ClubController.java:

```
package com.example.pipe;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired; 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.RequestBody; import
org.springframework.web.bind.annotation.RequestMapping; import
org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/api/clubs")
public class ClubController {
    @Autowired
    private ClubService clubService;
    @GetMapping
    public List<Club> getAllClubs() {
        return clubService.getAllClubs();
    }
    @PostMapping("/{clubId}/enroll")
    public void enrollMember(@PathVariable Long clubId, @RequestBody Member member) {
        clubService.enrollMember(clubId, member);
    }
    // Other CRUD endpoints for clubs
}
```

MemberRepository.java:

```
package com.example.pipe;
import org.springframework.data.jpa.repository.JpaRepository;
public interface MemberRepository extends JpaRepository<Member, Long> {
}
```

ClubRepository.java:

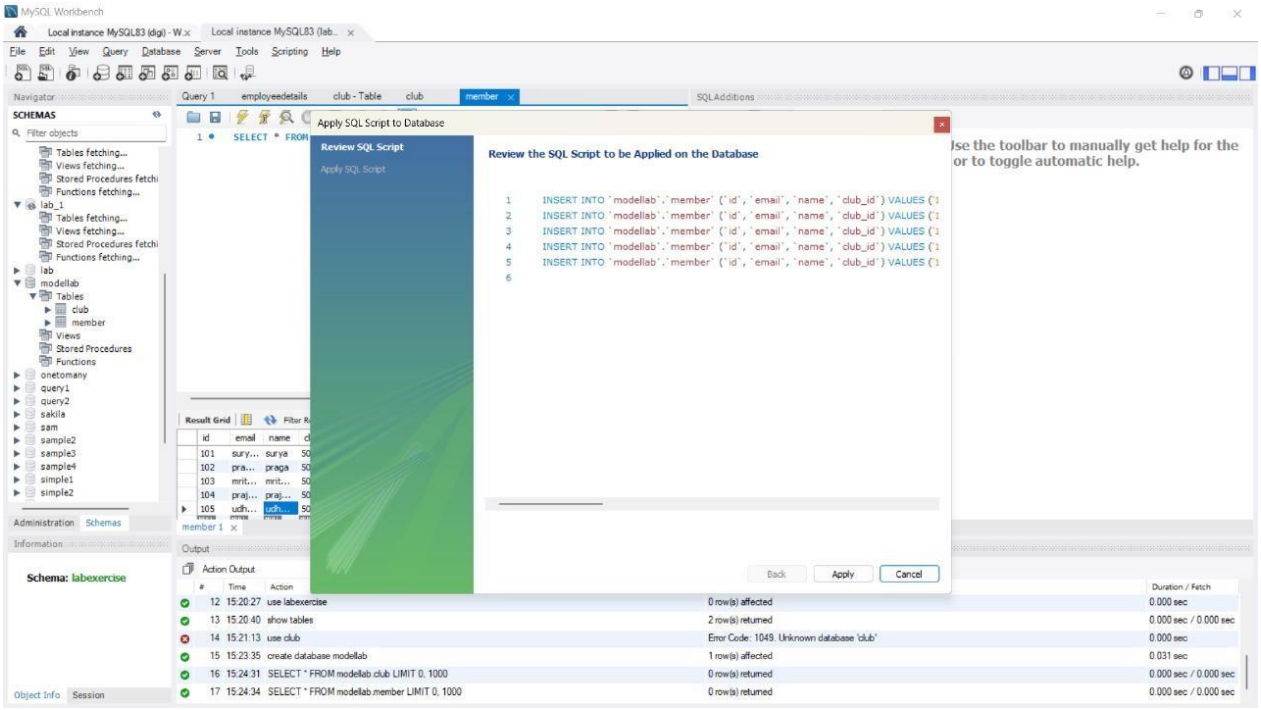
```
package com.example.pipe;

import org.springframework.data.jpa.repository.JpaRepository;
public interface ClubRepository extends JpaRepository<Club, Long> {
}
```

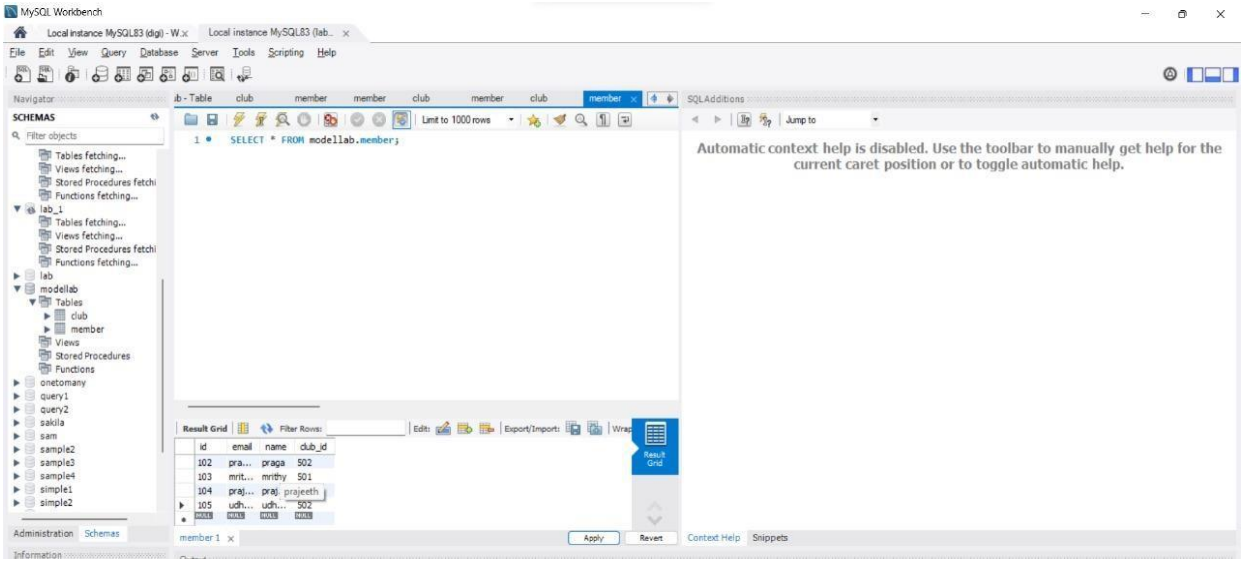
Application.properties:

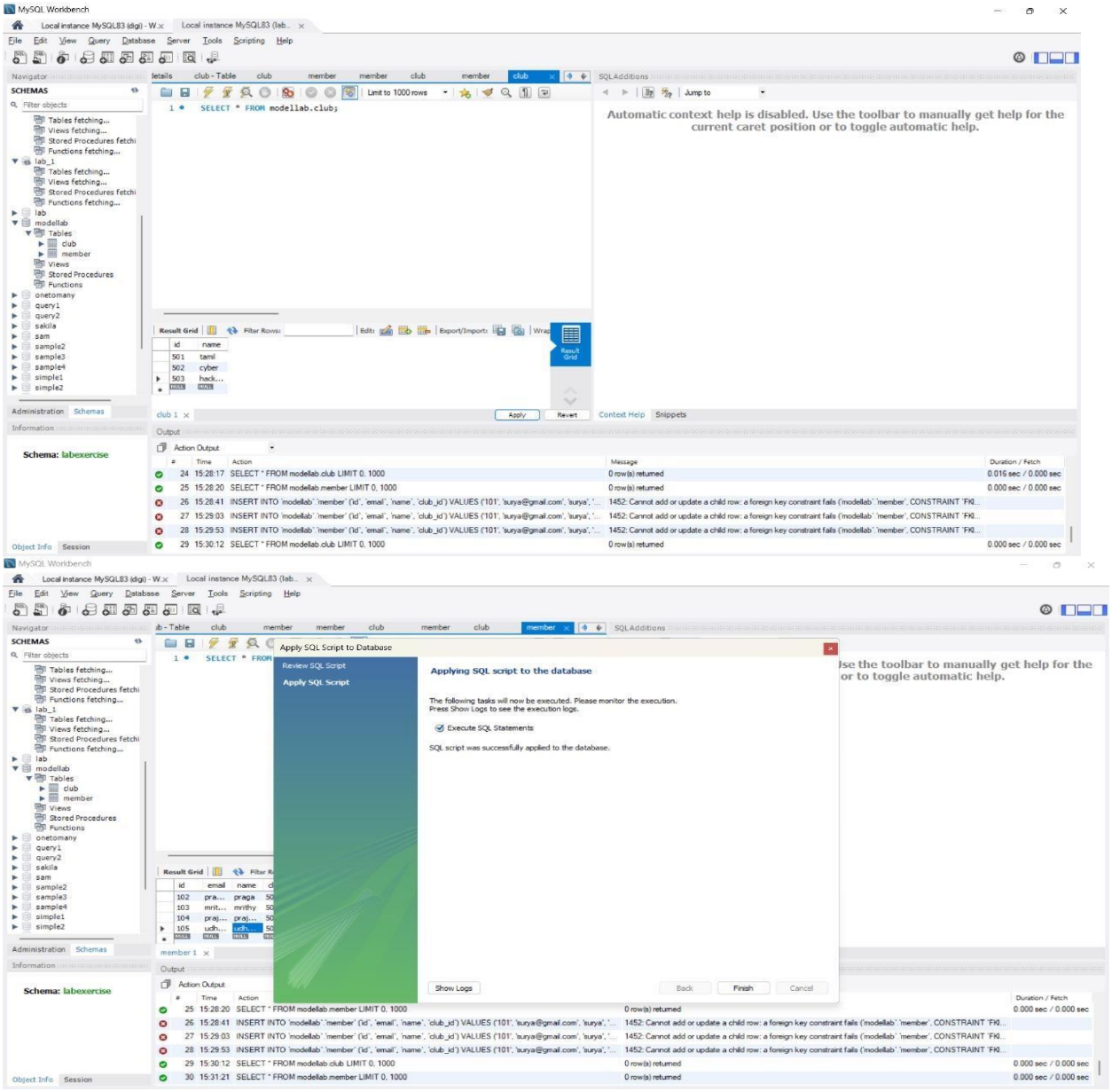
```
spring.application.name=lab
spring.datasource.url= jdbc:mysql://localhost:3306/modellab
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

spring.jpa.database= mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto= update

OUTPUT:







Result:

Thus, the application is implemented in Spring boot.

Project 8: Patient and Appointment

Management System Date:

Objective: Develop a comprehensive system to manage patients and their appointments in a medical facility.

Entities and Relationships:

Patient - Represents patients visiting the medical facility.

Attributes: patientId (Primary Key), name, Date_of_appointment, contactNumber, doctor (Many-to-One relationship with Doctor entity),issue

Doctor - Represents doctors providing medical services.

Attributes: doctorId (Primary Key), name, specialty, appointments (One-to-Many relationship with Patient entity through appointments)

Appointment - Represents appointments between patients and doctors.

Attributes: appointmentId (Primary Key), date, time, patient (Many-to-One relationship with Patient entity), doctor (Many-to-One relationship with Doctor entity)

Relational Mapping:

The Appointment entity has a Many-to-One relationship with both the Patient and Doctor entities, indicating that a patient can have multiple appointments with possibly different doctors, and a doctor can have appointments with multiple patients.

Basic Operations:

- Create, Read, Update, and Delete patients.
- Create, Read, Update, and Delete doctors.
- Schedule, reschedule, and cancel appointments.

MODEL CLASS:

1) Patient.java

```
package com.example.PatientManagementSystem;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Patient {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long patientId;
```

```
    private String name; private
    String dateOfBirth;
    private String contactNumber;
    private String issue;
        public Long getPatientId() {
            return patientId;
        }
        public void setPatientId(Long patientId) {
            this.patientId = patientId;
        }
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getDateOfBirth() {
            return dateOfBirth;
        }
        public void setDateOfBirth(String dateOfBirth) {
            this.dateOfBirth = dateOfBirth;
        }
        public String getContactNumber() {
            return contactNumber;
        }
        public void setContactNumber(String contactNumber) {
            this.contactNumber = contactNumber;
        }
        public String getIssue() {
            return issue;
        }
        public void setIssue(String issue) {
            this.issue = issue;
        }


}
```

2) Doctor.java

```
package com.example.PatientManagementSystem;

import jakarta.persistence.Entity;
import    jakarta.persistence.GeneratedValue;
import    jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Doctor {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long doctorId;
    private String name;
    private String specialty;
    // Getters and setters
        public Long getDoctorId() {
            return doctorId;
```

```
	}
	public void setDoctorId(Long doctorId) {
		this.doctorId = doctorId;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getSpecialty() {
		return specialty;
	}
	public void setSpecialty(String specialty) {
		this.specialty = specialty;
	}

}
```

3) Appointment.java

```
package com.example.PatientManagementSystem;

import jakarta.persistence.Entity;
import	jakarta.persistence.GeneratedValue;
import	jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import	jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;

@Entity
public class Appointment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long appointmentId;
    private String date;
    private String time;

    @ManyToOne @JoinColumn(name =
    "patient_id") private Patient patient;

    @ManyToOne @JoinColumn(name =
    "doctor_id") private Doctor doctor;

	public Long getAppointmentId() {
		return appointmentId;
	}

	public void setAppointmentId(Long appointmentId) {
		this.appointmentId = appointmentId;
	}

	public String getDate() {
		return date;
```

```
	}

	public void setDate(String date) {
		this.date = date;
	}

	public String getTime() {
		return time;
	}

	public void setTime(String time) {
		this.time = time;
	}

	public Patient getPatient() {
		return patient;
	}

	public void setPatient(Patient patient) {
		this.patient = patient;
	}

	public Doctor getDoctor() {
		return doctor;
	}

	public void setDoctor(Doctor doctor) {
		this.doctor = doctor;
	}

}
```

SERVICE

1) PatientService.java

```
package com.example.PatientManagementSystem;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
@Service
public class PatientService {
    @Autowired
    private PatientRepository patientRepository;

    public Patient createPatient(Patient patient) {
        return patientRepository.save(patient);
    }

    public List<Patient> getAllPatients() {
        return patientRepository.findAll();
    }

    public Patient getPatientById(Long id) {
```

```
        return patientRepository.findById(id).orElse(null);
    }

    public Patient updatePatient(Long id, Patient newPatientData) { Patient
        existingPatient = patientRepository.findById(id).orElse(null); if
        (existingPatient != null) {
            existingPatient.setName(newPatientData.getName()); existingPatient.setDateOfBirth(newPatientData.getDateOfBirth());
            existingPatient.setContactNumber(newPatientData.getContactNumber());
            existingPatient.setIssue(newPatientData.getIssue());
            return patientRepository.save(existingPatient);
        }
        return null;
    }

    public void deletePatient(Long id) {
        patientRepository.deleteById(id);
    }
}
```

2) DoctorService.java

```
package com.example.PatientManagementSystem;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service

public class DoctorService {
    @Autowired
    private DoctorRepository doctorRepository;

    public Doctor createDoctor(Doctor doctor) {
        return doctorRepository.save(doctor);
    }

    public List<Doctor> getAllDoctors() {
        return doctorRepository.findAll();
    }

    public Doctor getDoctorById(Long id) {
        return doctorRepository.findById(id).orElse(null);
    }

    public Doctor updateDoctor(Long id, Doctor newDoctorData) { Doctor
        existingDoctor = doctorRepository.findById(id).orElse(null); if
        (existingDoctor != null) {
            existingDoctor.setName(newDoctorData.getName());
            existingDoctor.setSpecialty(newDoctorData.getSpecialty());
            return doctorRepository.save(existingDoctor);
        }
        return null;
    }

    public void deleteDoctor(Long id) {
        doctorRepository.deleteById(id);
```

```
    }
}
```

3) AppointmentService.java

```
package com.example.PatientManagementSystem;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class AppointmentService {
    @Autowired
    private AppointmentRepository appointmentRepository;

    public Appointment scheduleAppointment(Appointment appointment) {
        return appointmentRepository.save(appointment);
    }

    public List<Appointment> getAllAppointments() {
        return appointmentRepository.findAll();
    }

    public Appointment getAppointmentById(Long id) {
        Optional<Appointment> appointmentOptional = appointmentRepository.findById(id);
        return appointmentOptional.orElse(null);
    }

    public Appointment rescheduleAppointment(Long id, String newDate, String newTime) {
        Optional<Appointment> appointmentOptional = appointmentRepository.findById(id); if
        (appointmentOptional.isPresent()) {
            Appointment existingAppointment = appointmentOptional.get();
            existingAppointment.setDate(newDate);
            existingAppointment.setTime(newTime);
            return appointmentRepository.save(existingAppointment);
        }
        return null; // Or throw an exception indicating appointment not found
    }

    public void cancelAppointment(Long id) {
        appointmentRepository.deleteById(id);
    }
}
```

REPOSITORY

1) PateintRepository.java

```
package com.example.PatientManagementSystem;

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

@Repository
public interface PatientRepository extends JpaRepository<Patient, Long> {
}
```

2) DoctorRepository.java

```
package com.example.PatientManagementSystem;

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

@Repository
public interface DoctorRepository extends JpaRepository<Doctor, Long> {
}
```

3) AppointmentRepository.java

```
package com.example.PatientManagementSystem;

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

@Repository
public interface AppointmentRepository extends JpaRepository<Appointment, Long> {
}
```

CONTROLLER

1) PatientController.java

```
package com.example.PatientManagementSystem;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/patient")
public class PatientController {
    @Autowired
    private PatientService patientService;

    @PostMapping
    public Patient createPatient(@RequestBody Patient patient) {
        return patientService.createPatient(patient);
    }

    @GetMapping
    public List<Patient> getAllPatients() {
        return patientService.getAllPatients();
    }

    @GetMapping("/{id}")
    public Patient getPatientById(@PathVariable Long id) {
        return patientService.getPatientById(id);
    }

    @PutMapping("/{id}")
    public Patient updatePatient(@PathVariable Long id, @RequestBody Patient newPatientData) { return
        patientService.updatePatient(id, newPatientData);
    }

    @DeleteMapping("/{id}")
    public void deletePatient(@PathVariable Long id) {
```

```
        patientService.deletePatient(id);
    }
}
```

2) DoctorController.java

```
package com.example.PatientManagementSystem;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/doctors")
public class DoctorController {
    @Autowired
    private DoctorService doctorService;

    @PostMapping
    public Doctor createDoctor(@RequestBody Doctor doctor) {
        return doctorService.createDoctor(doctor);
    }

    @GetMapping
    public List<Doctor> getAllDoctors() {
        return doctorService.getAllDoctors();
    }

    @GetMapping("/{id}")
    public Doctor getDoctorById(@PathVariable Long id) {
        return doctorService.getDoctorById(id);
    }

    @PutMapping("/{id}")
    public Doctor updateDoctor(@PathVariable Long id, @RequestBody Doctor newDoctorData) { return
        doctorService.updateDoctor(id, newDoctorData);
    }

    @DeleteMapping("/{id}")
    public void deleteDoctor(@PathVariable Long id) {
        doctorService.deleteDoctor(id);
    }
}
```

3) AppointmentController.java

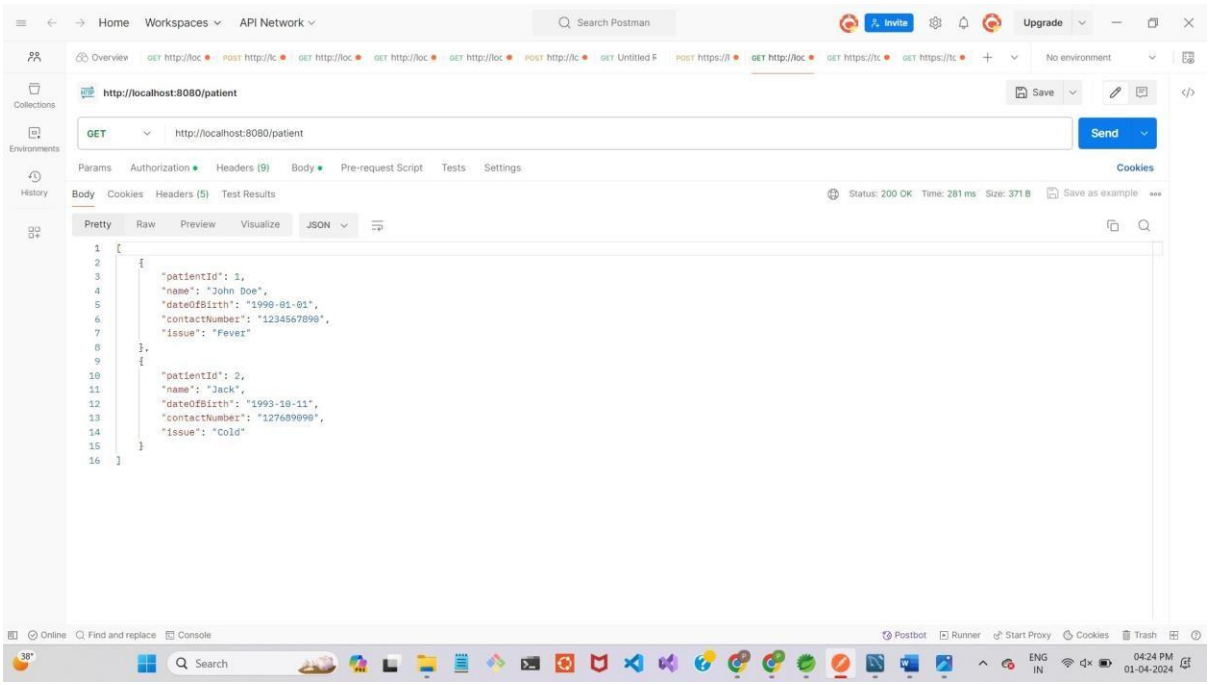
```
package com.example.PatientManagementSystem;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/appointments")
public class AppointmentController {
    @Autowired
    private AppointmentService appointmentService;
```

```
    @PostMapping
    public Appointment scheduleAppointment(@RequestBody Appointment appointment) { return
        appointmentService.scheduleAppointment(appointment);
    }

    @GetMapping
    public List<Appointment> getAllAppointments() {
        return appointmentService.getAllAppointments();
    }

    @GetMapping("/{id}")
    public Appointment getAppointmentById(@PathVariable Long id) {
        return appointmentService.getAppointmentById(id);
    }

    @PutMapping("/{id}/reschedule")
    public Appointment rescheduleAppointment(@PathVariable Long id, @RequestParam String newDate, @RequestParam
String newTime) {
        return appointmentService.rescheduleAppointment(id, newDate, newTime);
    }

    @DeleteMapping("/{id}")
    public void cancelAppointment(@PathVariable Long id) {
        appointmentService.cancelAppointment(id);
    }
}
```

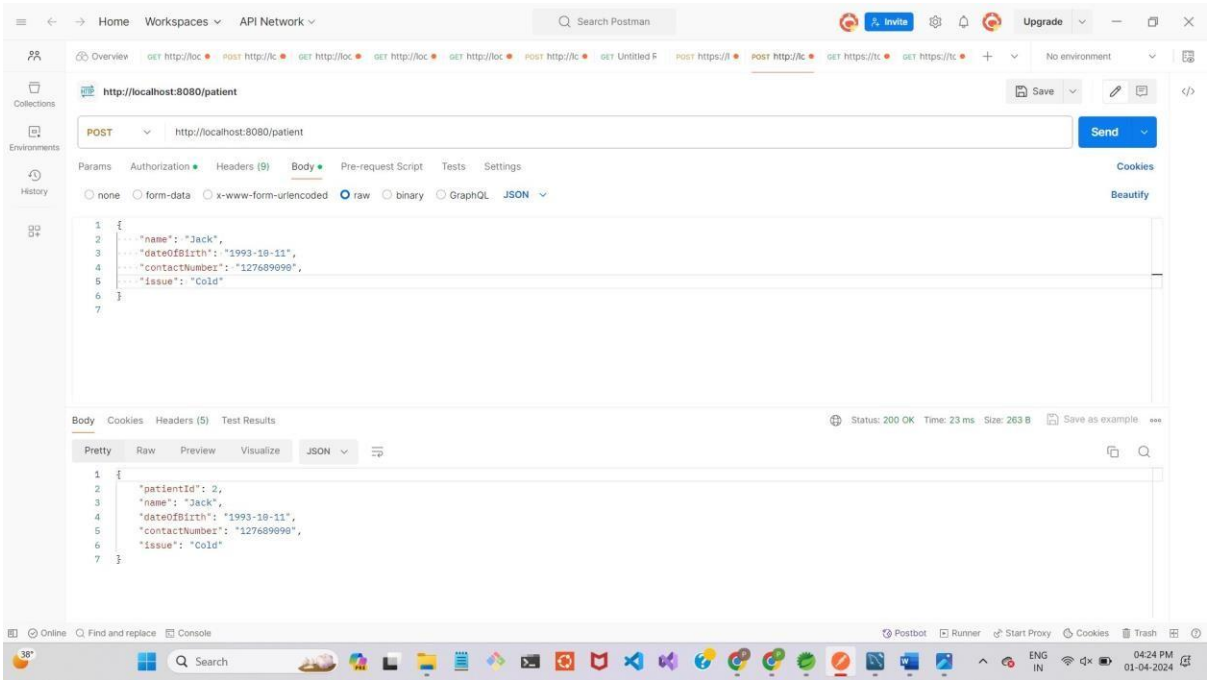
APPLICATION PROPERTY

```
spring.application.name=PatientManagementSystem

spring.datasource.url=jdbc:mysql://localhost:3306/sample9
spring.datasource.username=root spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database=mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

GET:

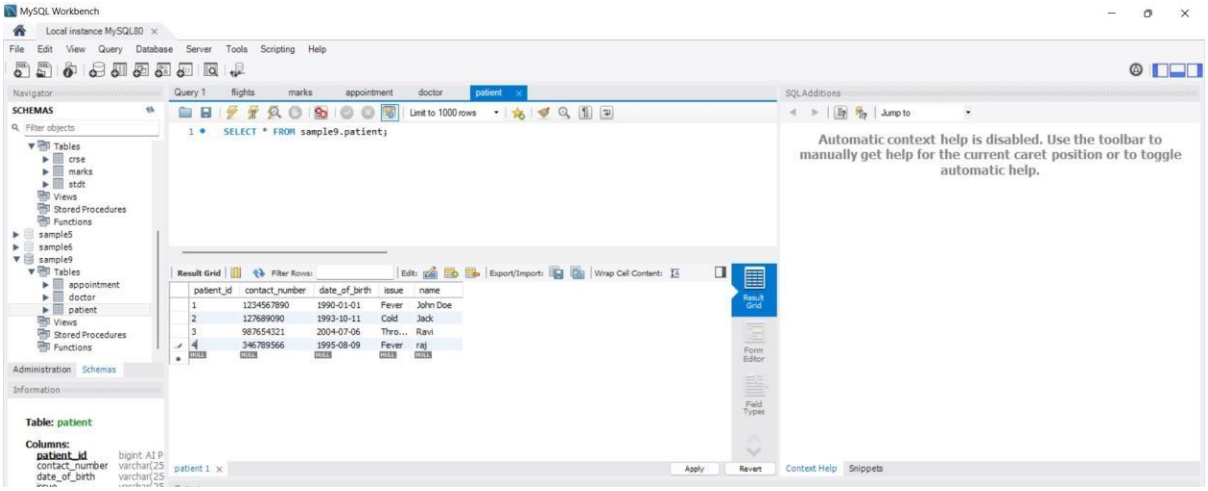

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
No environment
http://localhost:8080/patient
Save
GET
http://localhost:8080/patient
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 281 ms
Size: 371 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
[
{
"patientId": 1,
"name": "John Doe",
"dateOfBirth": "1990-01-01",
"contactNumber": "1234567890",
"issue": "Fever"
},
{
"patientId": 2,
"name": "Jack",
"dateOfBirth": "1993-10-11",
"contactNumber": "127689090",
"issue": "Cold"
}
]
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
ENG
IN
04:24 PM
01-04-2024

POST:

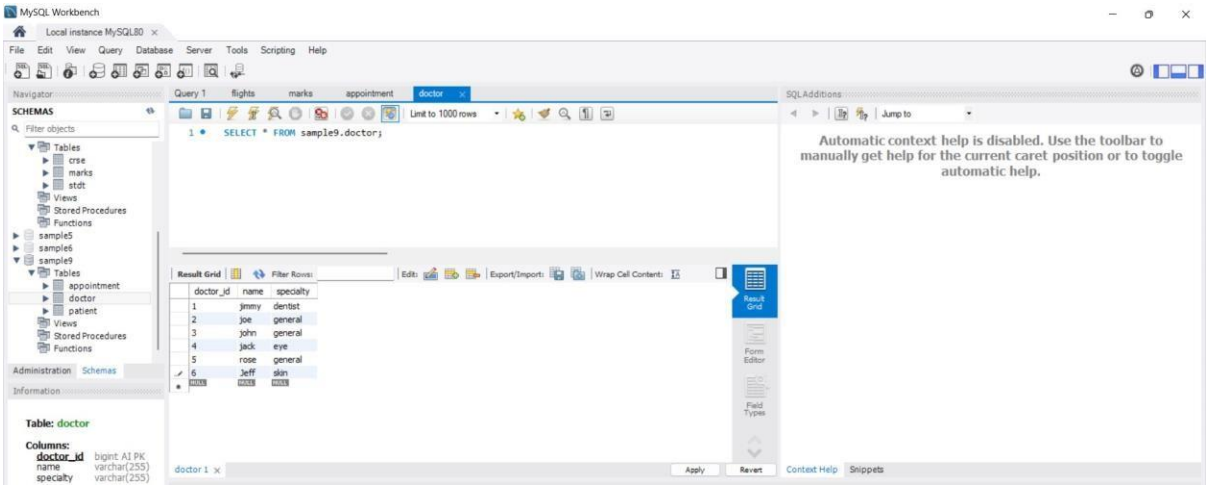

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
No environment
http://localhost:8080/patient
Save
POST
http://localhost:8080/patient
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name": "Jack",
"dateOfBirth": "1993-10-11",
"contactNumber": "127689090",
"issue": "Cold"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 23 ms
Size: 263 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"patientId": 2,
"name": "Jack",
"dateOfBirth": "1993-10-11",
"contactNumber": "127689090",
"issue": "Cold"
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
Search
ENG
IN
04:24 PM
01-04-2024

DATABA

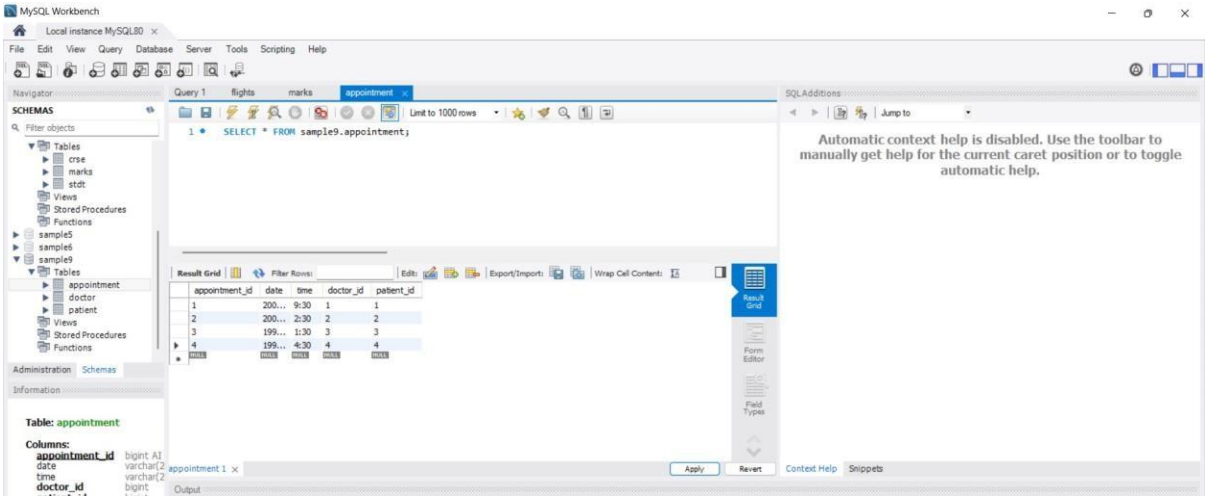
SE: PATIENT:



DOCTOR



APPOINTMENT



Result:

Thus, the application is implemented in Spring boot.

Project 9: Managing User Entities

with JPA Date:

Create a simple CRUD application using Spring framework, Maven, and MySQL as a backend database. Implement the CRUD operations for a 'User' entity using JPA. Your application should have the following functionalities:

- Create a new user with attributes such as name, email, and age.
- Retrieve a list of all users from the database.
- Retrieve a specific user by their ID.
- Update the details of an existing user.
- Delete a user by their ID.

User Entity Fields:

- **id:** A unique identifier for each user. Typically, this would be an auto-generated field.
- **name:** The user's full name. This is a string.
- **email:** The user's email address. This field should be unique to ensure that each user can be individually identified and contacted. This is a string.
- **age:** The user's age. This could be represented by an integer.
- **createdAt:** The date and time when the user account was created. This can be automatically set when a new user record is created.
- **updatedAt:** The date and time when the user account was last updated. This field can be automatically updated every time the user's information is modified.
- **status:** The user's account status. This can indicate whether the account is active, suspended, or deleted. Enumerations can be used here to define possible states.

Model:

```
package com.example.spring;


import jakarta.persistence.Entity;
import	jakarta.persistence.GeneratedValue;
import	jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name="Marks")
public class Model {
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private int id;
	private String name,email;
	private int age;
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
```

```
	public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
}
```

Controller:

```
package com.example.spring;
import java.util.List;

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

@RestController
public class Controller {
	@Autowired
	public service s;
	@GetMapping("/get")
	public List<Model> getdata(){
		return s.getall();
	}
	@GetMapping("/viewbyraw/{searchid}")
	public List<Model> viewbyraw(@PathVariable int searchid){
		return s.findid(searchid);
	}
	@PostMapping("/save")
	public String save(@RequestBody Model m) { s.add(m);
		return "added";
	}
	@PutMapping("/update/{id}")
	public ResponseEntity<Model> updateuser(@PathVariable int id,@RequestBody Model m){
		m.setId(id);
```

```
		return s.updateuser(m);
	}
	@DeleteMapping("/delete/{id}")
	public String deleteuser(@PathVariable int id){ return
		s.delete(id);

	}
}
```

Repo:

```
package com.example.spring;


import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository; import
org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

@Repository
public interface Repo extends JpaRepository<Model,Integer>{
	@Query("select e from Model e where e.id = :searchid")
	List<Model> find(int searchid);
}
```

service:

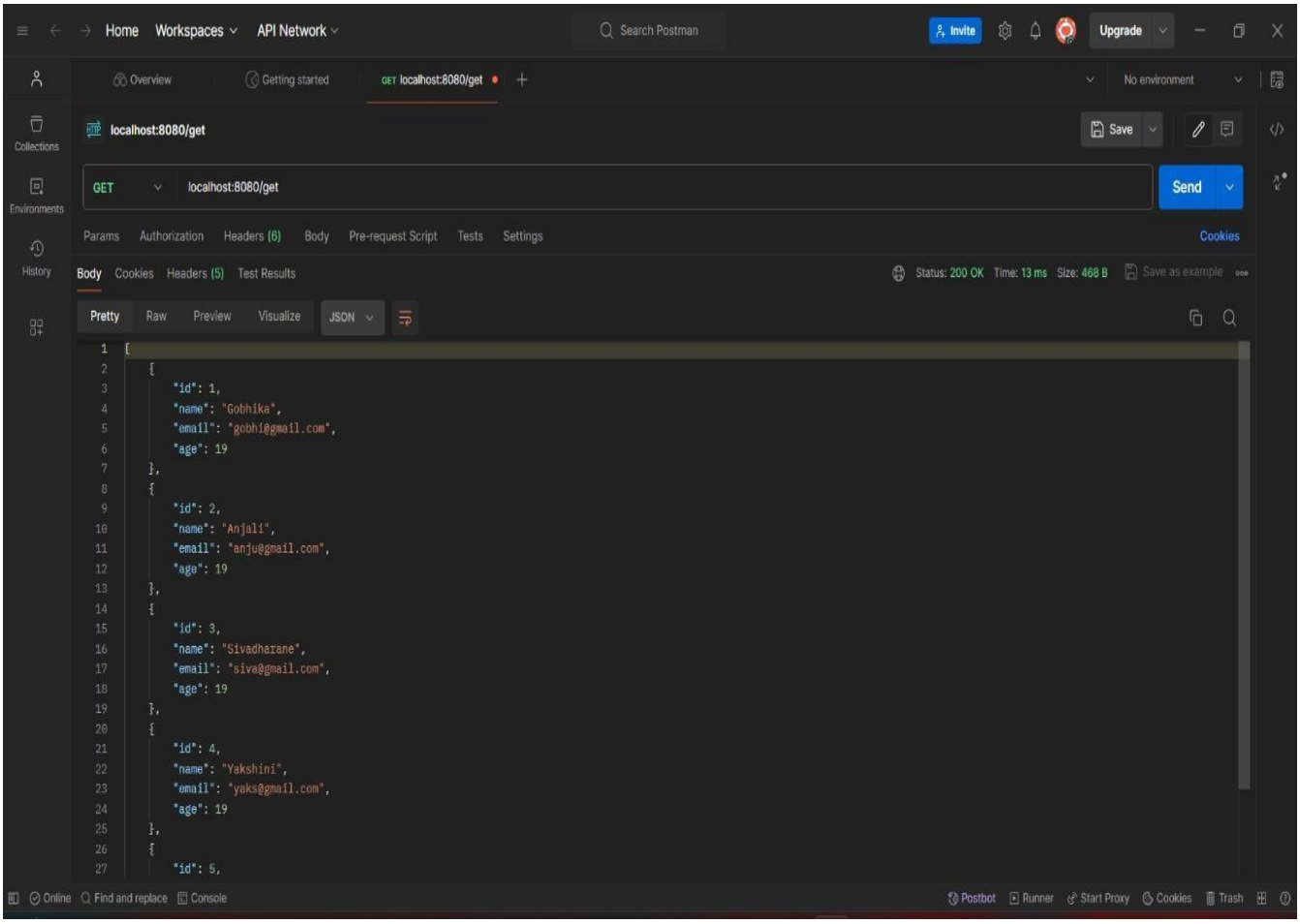
```
package com.example.spring;


import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;

@Service
public class service {
	@Autowired
	public Repo r;
	public List<Model> getall() {
		return r.findAll();
	}
	public String add(Model m) {
	r.save(m);
	return "Added";
	}
	public List<Model> findid(int searchid){
		return r.find(searchid);
	}
	public ResponseEntity<Model> updateuser( Model m) {
```

```
            r.save(m);
            return ResponseEntity.ok(m);
    }
   public String delete(int id){
            r.deleteById(id);
            return "Successfully deleted";
   }
}
```

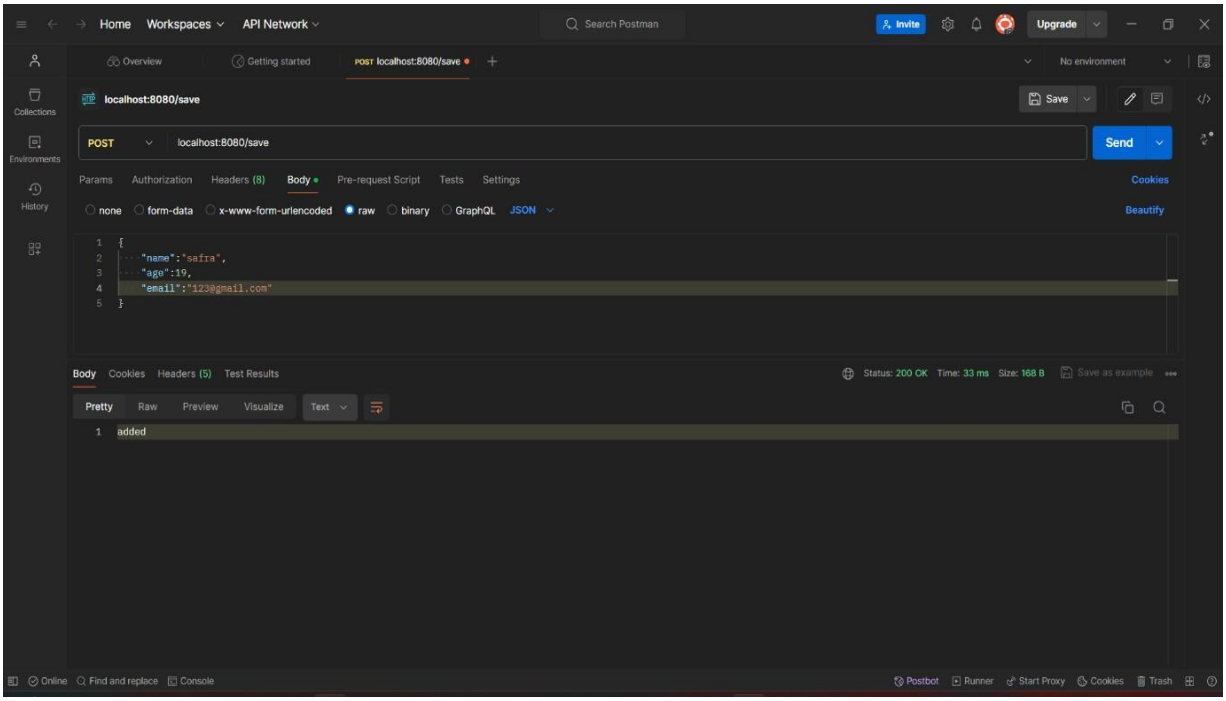
Application Property :

```
spring.application.name=UserEntity
spring.datasource.url=jdbc:mysql://localhost:3306/entity
spring.datasource.username=root
spring.datasource.password=mysql
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
#spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.database=mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

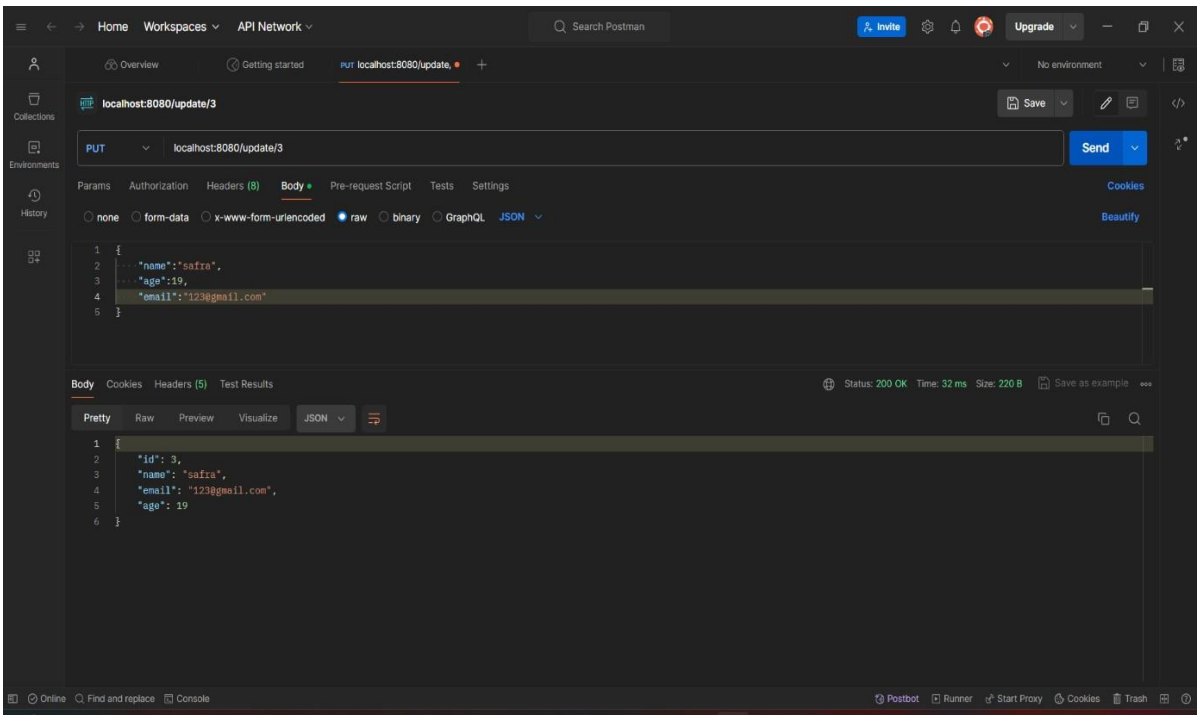
Get:



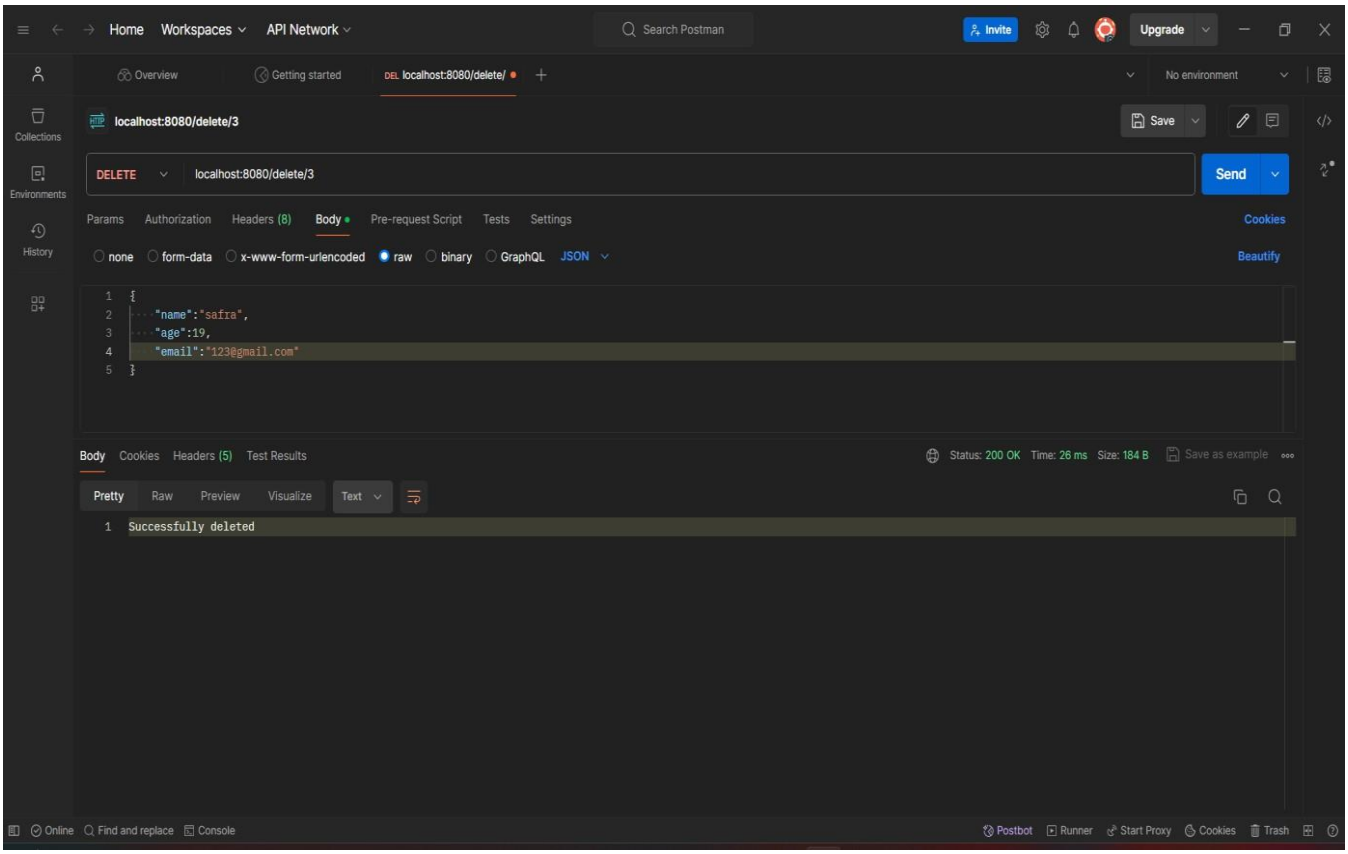
Post:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Overview
Getting started
POST localhost:8080/save
No environment
localhost:8080/save
Save
Collections
Environments
History
POST
localhost:8080/save
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"safra",
"age":19,
"email":"123@gmail.com"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 33 ms
Size: 168 B
Save as example
Pretty
Raw
Preview
Visualize
Text
added
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash

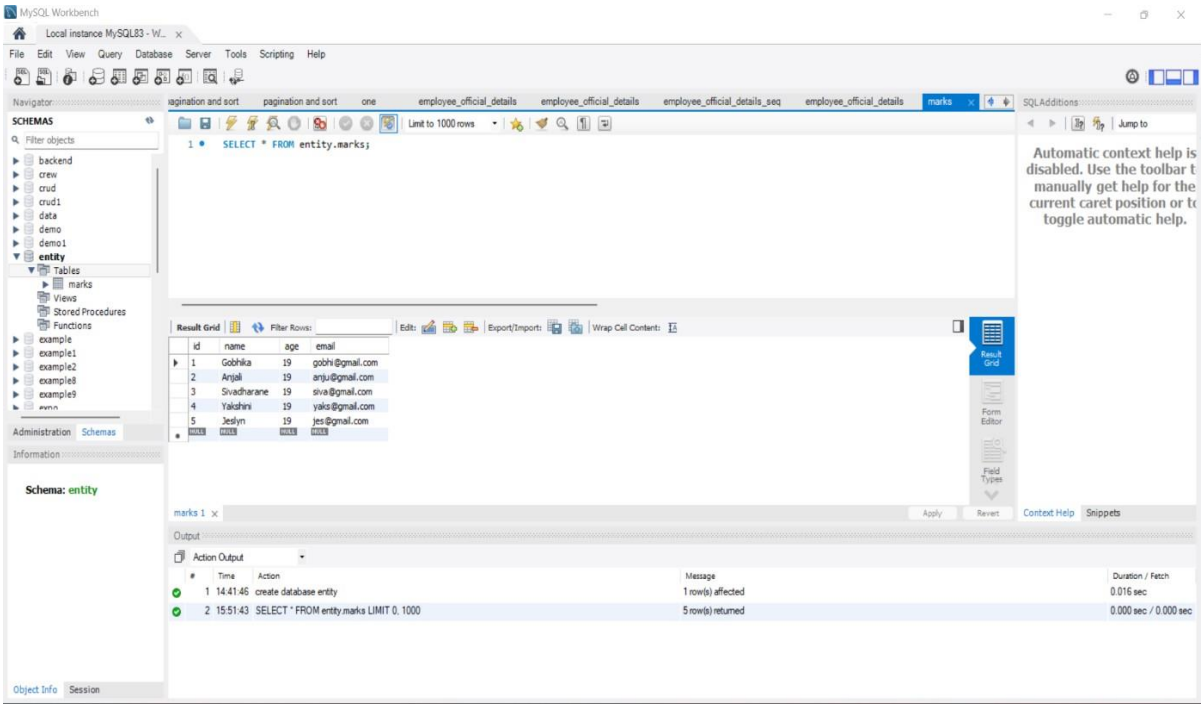
Put:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
Overview
Getting started
PUT localhost:8080/update,
No environment
localhost:8080/update/3
Save
Collections
Environments
History
PUT
localhost:8080/update/3
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"name":"safra",
"age":19,
"email":"123@gmail.com"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 32 ms
Size: 220 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 3,
"name": "safra",
"email": "123@gmail.com",
"age": 19
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash

Delete:



Database:



Result:

Thus, the application is implemented in Spring boot.

Project 10: Vehicle Dealership Inventory

Management System Date:

Design and implement a basic web application for a vehicle dealership using the Spring framework, Maven, and MySQL. Utilize JPA for performing CRUD operations on 'Vehicle' entities. The application should enable the following capabilities:

- Register a new vehicle with details like make, model, VIN (Vehicle Identification Number), and year of manufacture.
- Display a catalog of all vehicles in the dealership's inventory.
- Provide a feature to search for vehicles based on make or model.
- Allow for the modification of vehicle details.
- Facilitate the removal of a vehicle from the inventory.

Vehicle Entity Fields:

- **id:** A unique identifier for each vehicle in the inventory. This could be an auto-generated field .
- **make:** The manufacturer of the vehicle. This is a string representing the brand.
- **model:** The specific model of the vehicle. This is a string.
- **vin:** The Vehicle Identification Number, which is a unique code used to identify individual motor vehicles. This is a string.
- **yearOfManufacture:** The year the vehicle was manufactured. This could be represented by an integer.
- **color:** The color of the vehicle. This detail can be important for buyers and is a string.
- **price:** The sale price of the vehicle. This could be a decimal to accommodate cents, useful for pricing.

MODEL:

```
package com.example.Vehicle;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "vehicles")
public class Vehicle {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    private Long id;

    private String make;
    private String model;
    private String vin;
    private int yearOfManufacture;
    private String color;
    private double price; public
        Long getId() {
            return id;
        }
        public void setId(Long id) {
            this.id = id;
        }
        public String getMake() {
            return make;
        }
        public void setMake(String make) {
            this.make = make;
        }
        public String getModel() {
            return model;
        }
        public void setModel(String model) {
            this.model = model;
        }
        public String getVin() {
            return vin;
        }
        public void setVin(String vin) {
            this.vin = vin;
        }
        public int getYearOfManufacture() {
            return yearOfManufacture;
        }
        public void setYearOfManufacture(int yearOfManufacture) {
            this.yearOfManufacture = yearOfManufacture;
        }
        public String getColor() {
            return color;
        }
        public void setColor(String color) {
            this.color = color;
        }
        public double getPrice() {
            return price;
        }
        public void setPrice(double price) {
            this.price = price;
        }


}
```

CONTROLLER:

```
package com.example.Vehicle;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController @RequestMapping("/vehicles")
@CrossOrigin(origins = "http://localhost:3000")
public class VehicleController {

    @Autowired
    private VehicleService vehicleService;

    @GetMapping
    public List<Vehicle> getAllVehicles() {
        return vehicleService.getAllVehicles();
    }

    @GetMapping("/{id}")
    public Vehicle getVehicleById(@PathVariable Long id) {
        return vehicleService.getVehicleById(id);
    }

    @PostMapping
    public Vehicle saveVehicle(@RequestBody Vehicle vehicle) {
        return vehicleService.saveVehicle(vehicle);
    }

    @PutMapping("/{id}")
    public Vehicle updateVehicle(@PathVariable Long id, @RequestBody Vehicle updatedVehicle) { return
        vehicleService.updateVehicle(id, updatedVehicle);
    }

    @DeleteMapping("/{id}")
    public void deleteVehicleById(@PathVariable Long id) {
        vehicleService.deleteVehicleById(id);
    }
}
```

REPO:

```
package com.example.Vehicle;

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

public interface VehicleRepository extends JpaRepository<Vehicle, Long>{
}
```

SERVICE

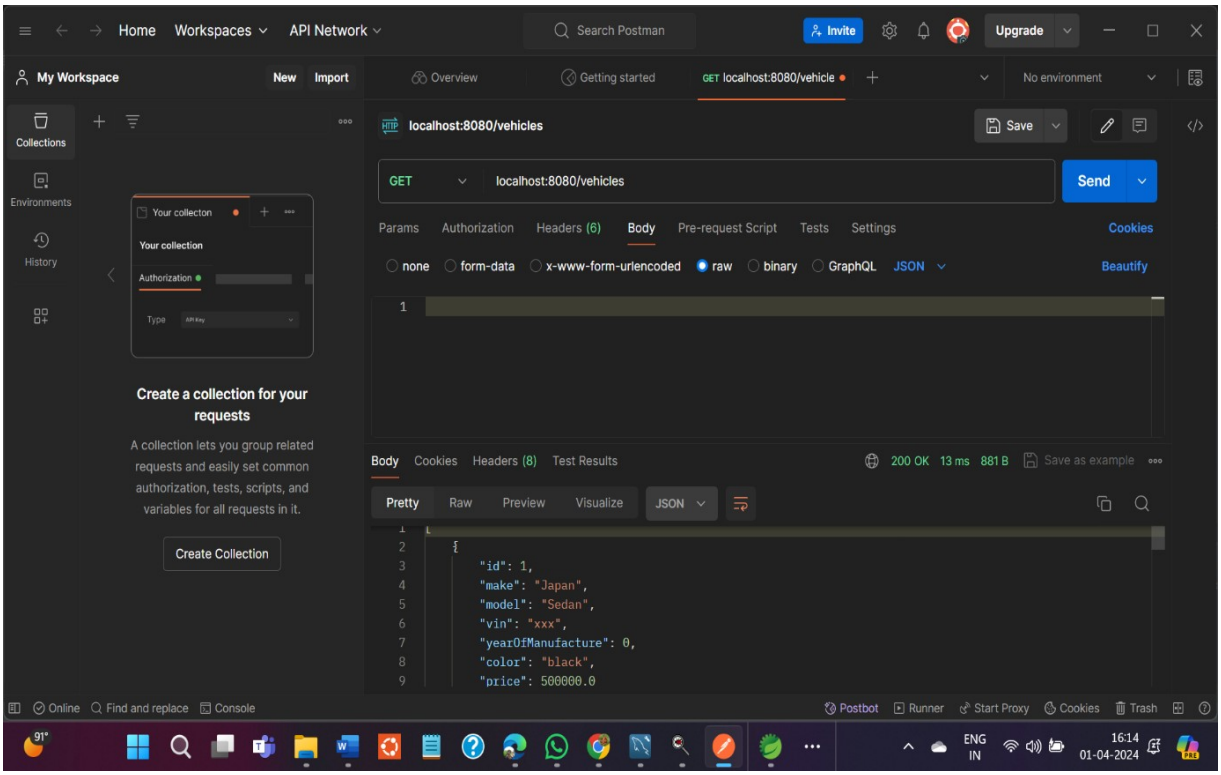
```
package com.example.Vehicle;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
```

```
import java.util.Optional;

@Service
public class VehicleService {
    @Autowired
    private VehicleRepository vehicleRepository;

    public List<Vehicle> getAllVehicles() {
        return vehicleRepository.findAll();
    }

    public Vehicle saveVehicle(Vehicle vehicle) {
        return vehicleRepository.save(vehicle);
    }

    public Vehicle getVehicleById(Long id) {
        return vehicleRepository.findById(id).orElse(null);
    }

    public void deleteVehicleById(Long id) {
        vehicleRepository.deleteById(id);
    }
    public Vehicle updateVehicle(Long id, Vehicle updatedVehicle) {
        Optional<Vehicle> optionalVehicle = vehicleRepository.findById(id); if
        (optionalVehicle.isPresent()) {
            Vehicle existingVehicle = optionalVehicle.get();
            existingVehicle.setMake(updatedVehicle.getMake());
            existingVehicle.setModel(updatedVehicle.getModel());
            existingVehicle.setColor(updatedVehicle.getColor());
            existingVehicle.setVin(updatedVehicle.getVin());
            existingVehicle.setYearOfManufacture(updatedVehicle.getYearOfManufacture()); return
            vehicleRepository.save(existingVehicle);
        } else {
            return null; // Or throw an exception, depending on your error handling strategy
        }
    }
}
```

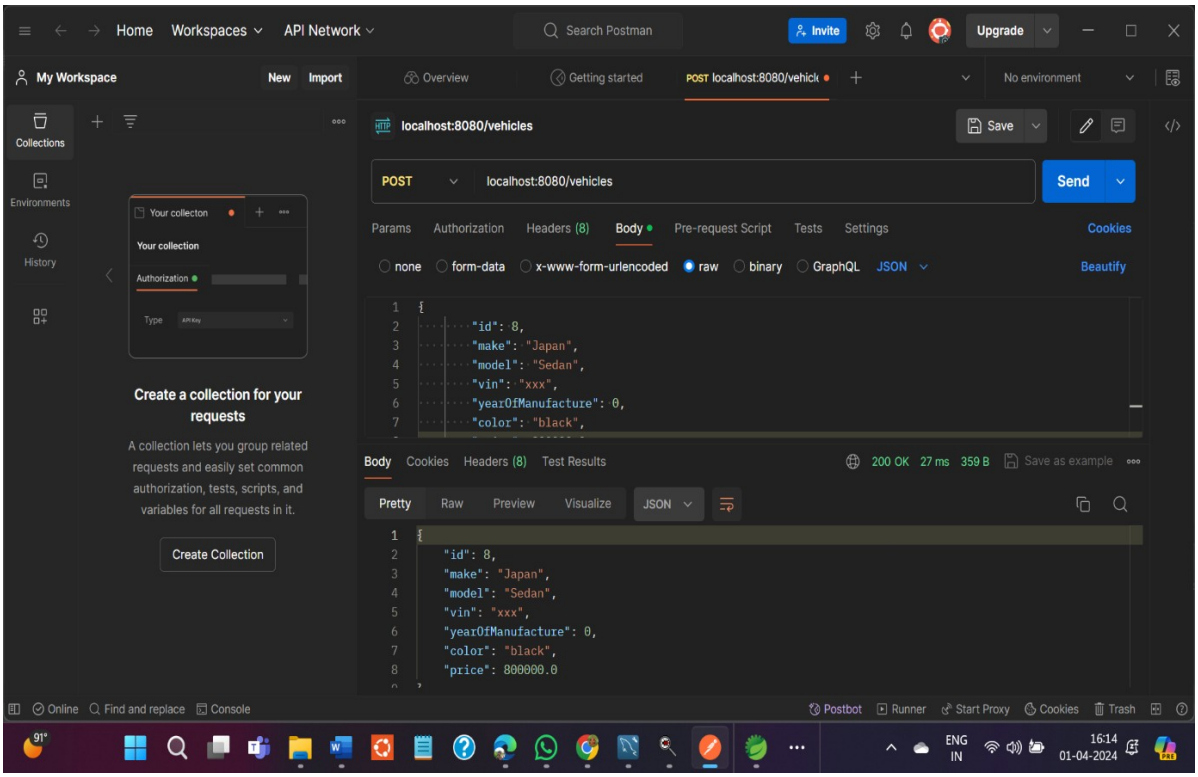
APPLICATION-PROPERTIES:

```
spring.application.name=Vehicle spring.datasource.url=
jdbc:mysql://localhost:3306/new
spring.datasource.username=root
spring.datasource.password=kasthuri@1102004
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database= mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto= update
```

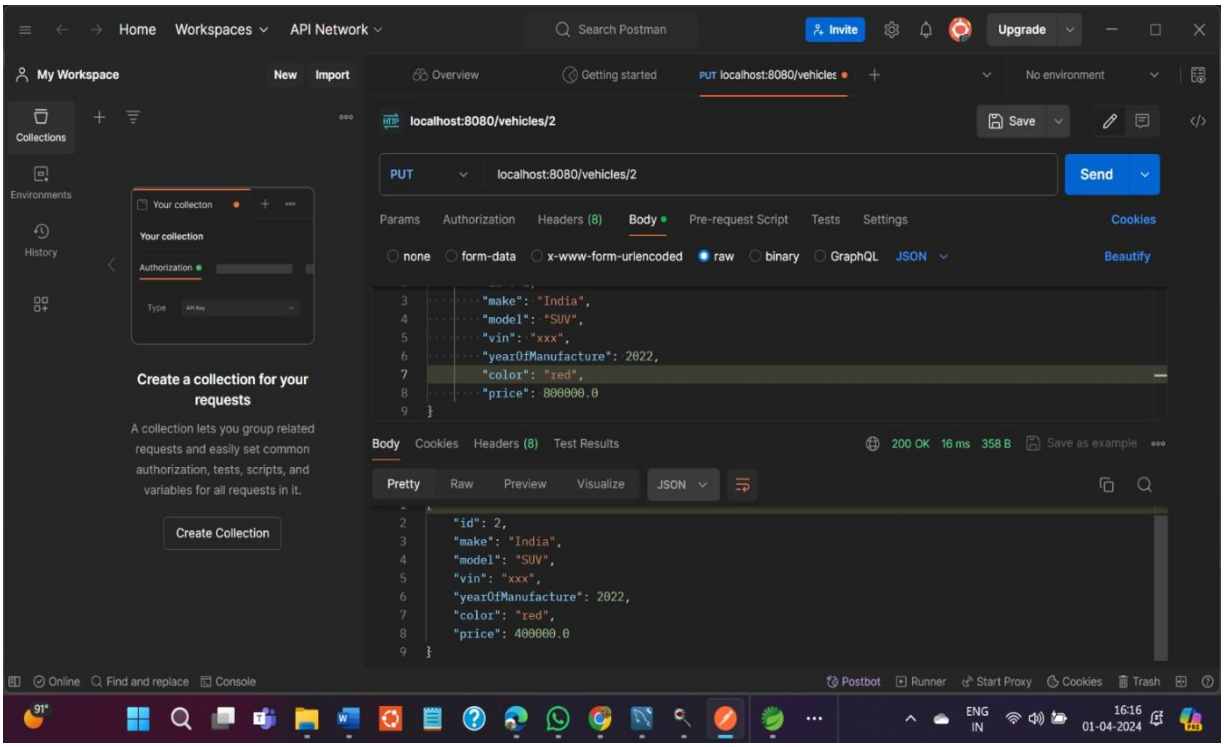
GET:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Getting started
GET localhost:8080/vehicle
No environment
Collections
Environments
History
localhost:8080/vehicles
Save
GET
localhost:8080/vehicles
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (8)
Test Results
200 OK
13 ms
881 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"make": "Japan",
"model": "Sedan",
"vin": "xxx",
"yearOfManufacture": 0,
"color": "black",
"price": 500000.0
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
ENG
IN
16:14
01-04-2024

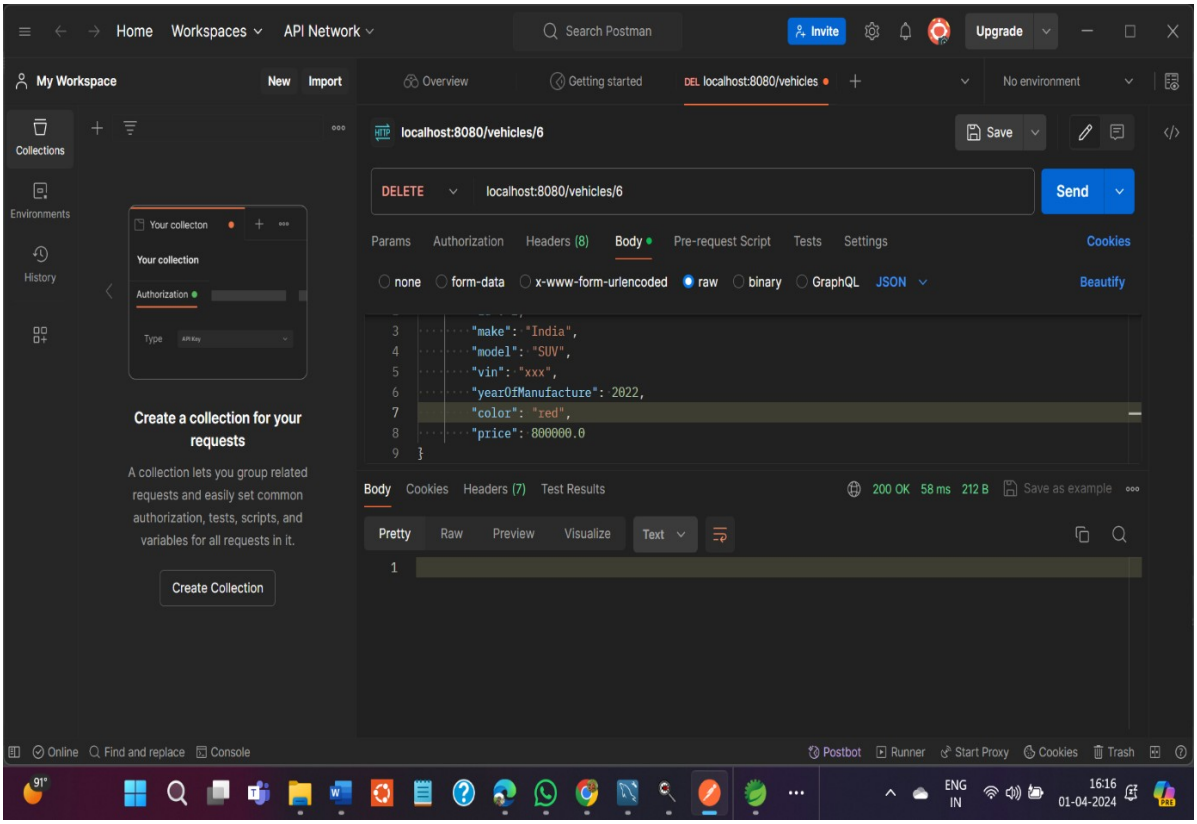
POST:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Getting started
POST localhost:8080/vehicle
No environment
Collections
Environments
History
localhost:8080/vehicles
Save
POST
localhost:8080/vehicles
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Body
Cookies
Headers (8)
Test Results
200 OK
27 ms
359 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
{
"id": 8,
"make": "Japan",
"model": "Sedan",
"vin": "xxx",
"yearOfManufacture": 0,
"color": "black",
"price": 800000.0
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
ENG
IN
16:14
01-04-2024

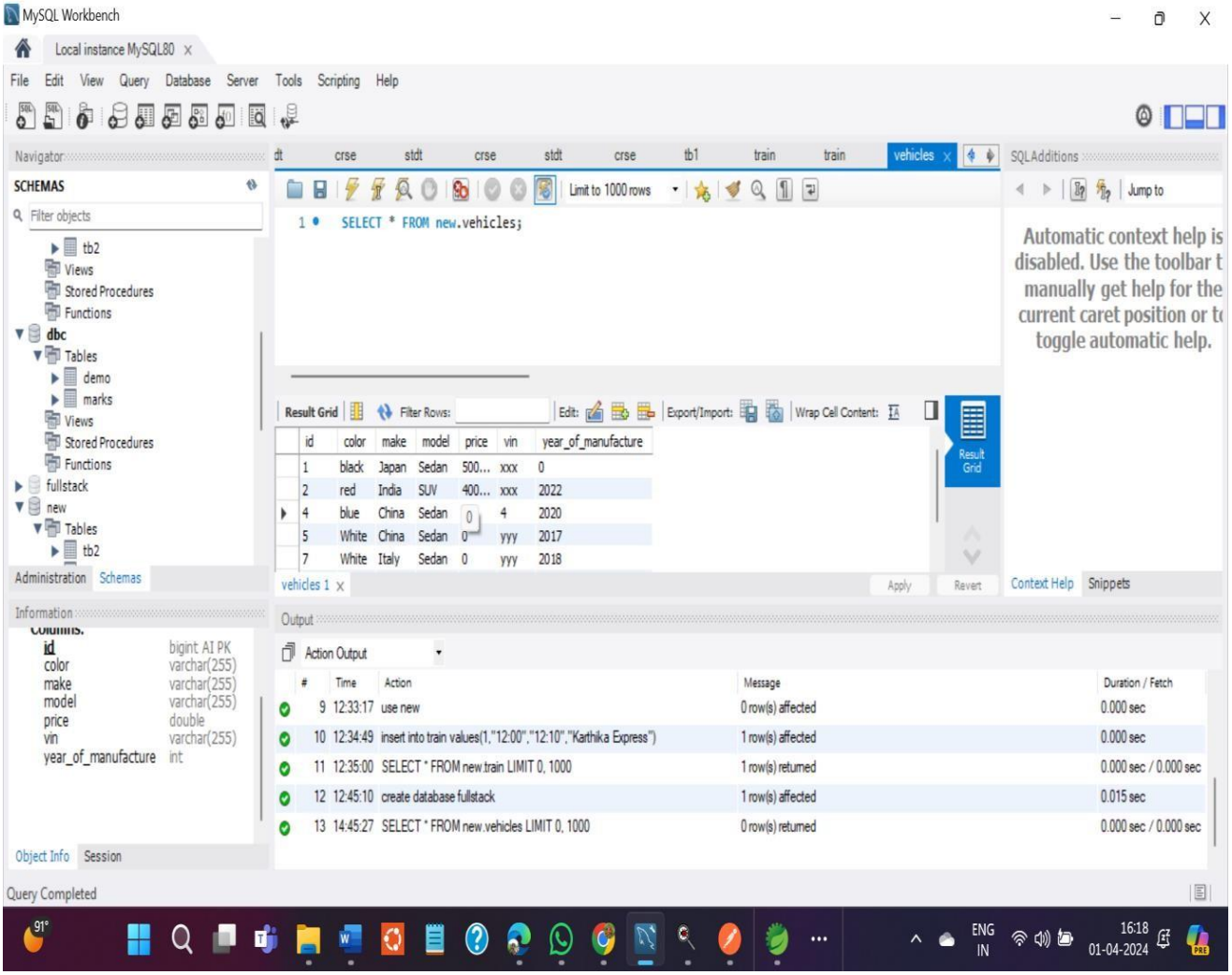
PUT:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Getting started
PUT localhost:8080/vehicles
No environment
Collections
Environments
History
localhost:8080/vehicles/2
Save
PUT
localhost:8080/vehicles/2
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
"make": "India",
"model": "SUV",
"vin": "xxx",
"yearOfManufacture": 2022,
"color": "red",
"price": 800000.0
Body
Cookies
Headers (8)
Test Results
200 OK
16 ms
358 B
Save as example
Pretty
Raw
Preview
Visualize
JSON
"id": 2,
"make": "India",
"model": "SUV",
"vin": "xxx",
"yearOfManufacture": 2022,
"color": "red",
"price": 800000.0
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
ENG
IN
16:16
01-04-2024

DELETE:


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
Getting started
DEL localhost:8080/vehicles
No environment
Collections
Environments
History
localhost:8080/vehicles/6
Save
DELETE
localhost:8080/vehicles/6
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
"make": "India",
"model": "SUV",
"vin": "xxx",
"yearOfManufacture": 2022,
"color": "red",
"price": 800000.0
Body
Cookies
Headers (7)
Test Results
200 OK
58 ms
212 B
Save as example
Pretty
Raw
Preview
Visualize
Text
Create a collection for your requests
A collection lets you group related requests and easily set common authorization, tests, scripts, and variables for all requests in it.
Create Collection
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Trash
ENG
IN
16:16
01-04-2024

DATABASE:



Result:

Thus, the application is implemented in Spring boot.