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:

```java
package com.example.demo.Entity;

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

@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String email;
    private String department;
    private double salary;
      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;
	}
	public Employee(String name, String email, String department, double salary) {
		super();
		this.name = name;
		this.email = email;
		this.department = department;
		this.salary = salary;
	}
	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", email=" + email + ", department=" +
department + ", salary="
				+ salary + "]";
	}

    public Employee() {
		// TODO Auto-generated constructor stub
	}
}
```

Controller:

```
package com.example.demo.Controller;

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

import com.example.demo.Entity.Employee;
import com.example.demo.repo.EmployeeRepository;

import jakarta.transaction.Transactional;



@RestController
@RequestMapping("/employees")
public class EmployeeController {
    @Autowired
    private EmployeeRepository employeeRepository;

    @GetMapping("/")
    public List<Employee> getAllEmployees() {
        return employeeRepository.findAll();
    }

    @GetMapping("/{id}")
    public Employee getEmployeeById(@PathVariable Long id) {
```

```java
        return employeeRepository.findById(id).orElse(null);
    }

    @PostMapping("/path")
    public String addEmployee(@RequestBody Employee employee) {
        employeeRepository.save(employee);
        return "Employee added successfully";

    }

    @PutMapping("/{id}")
    public Employee updateEmployee(@PathVariable Long id, @RequestBody Employee employeeDetails) {
        Employee employee = employeeRepository.findById(id).orElse(null);
        if (employee != null) {
            employee.setName(employeeDetails.getName());
            employee.setEmail(employeeDetails.getEmail());
            employee.setDepartment(employeeDetails.getDepartment());
            employee.setSalary(employeeDetails.getSalary());
            return employeeRepository.save(employee);
        }
        return null;
    }

    @DeleteMapping("/{id}")
    @Transactional
    public String deleteEmployee(@PathVariable Long id) {
        employeeRepository.deleteById(id);
        return "Employee with id " + id + " deleted successfully";
    }
}
```

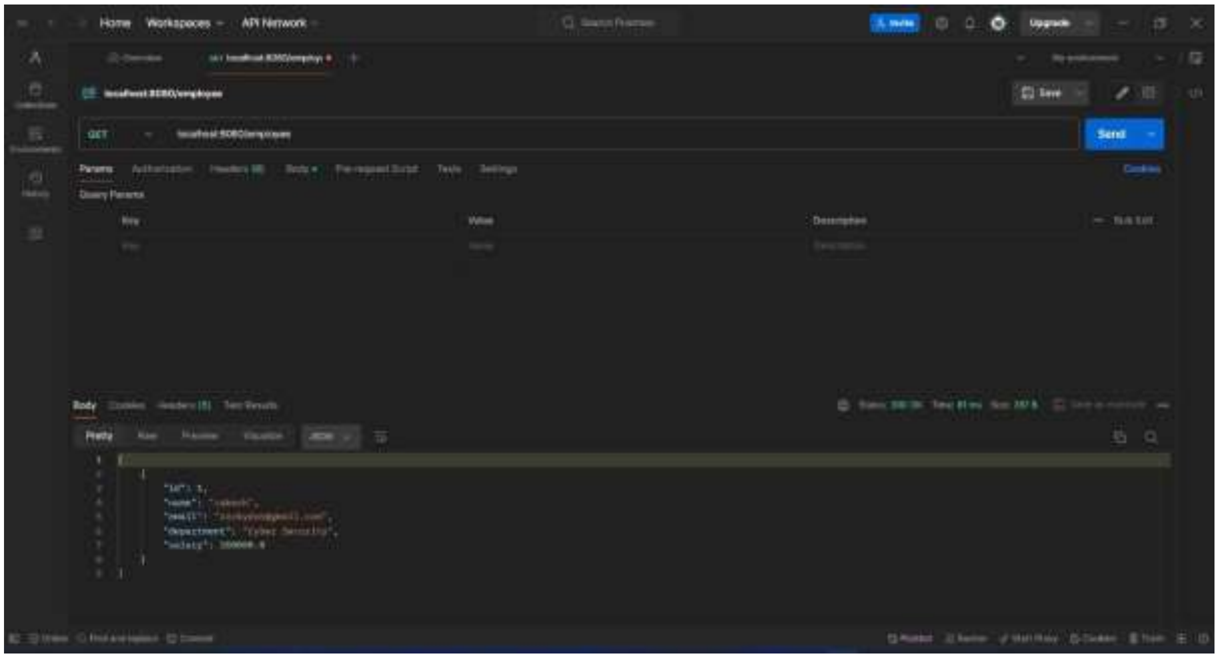
Repo:

```java
package com.example.demo.repo;

import java.util.Optional;

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

import com.example.demo.Entity.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

	void deleteById(Long id);

	Optional<Employee> findById(Long id);

}
```

Application Property:

```
spring.application.name=practice
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/studentdatabase
spring.datasource.username=root
spring.datasource.password=varshu@0503
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

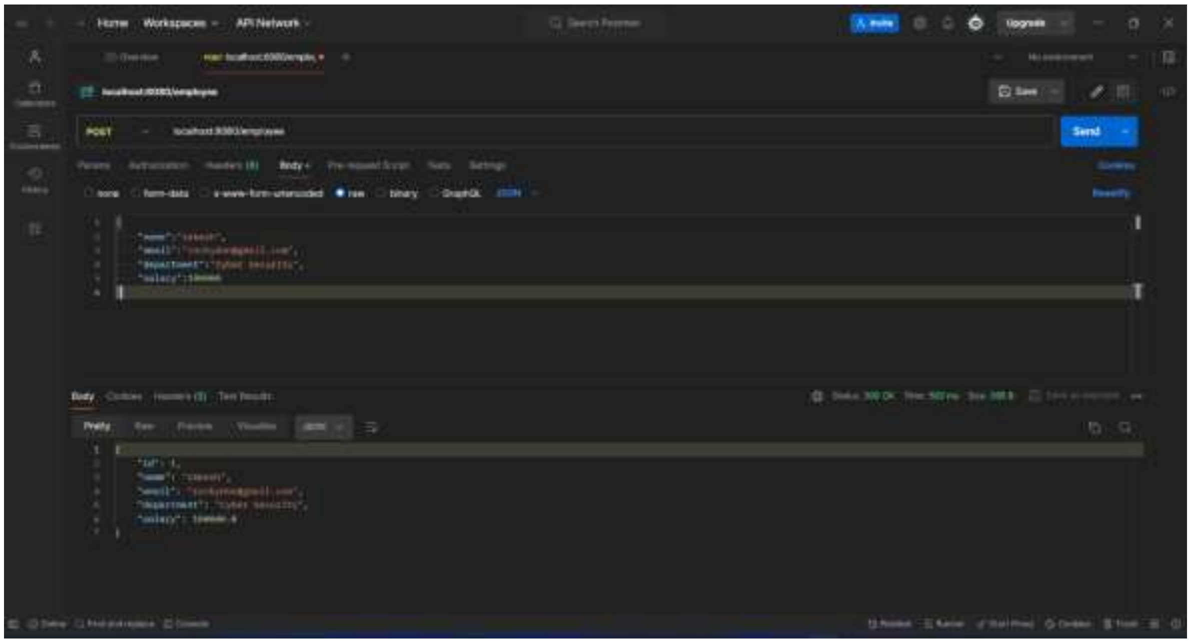
GET:



GET localhost:8080/employee

```
[
    {
        "id": 1,
        "name": "[illegible]",
        "email": "[illegible]@gmail.com",
        "department": "Cyber Security",
        "salary": 100000.0
    }
]
```

POST:

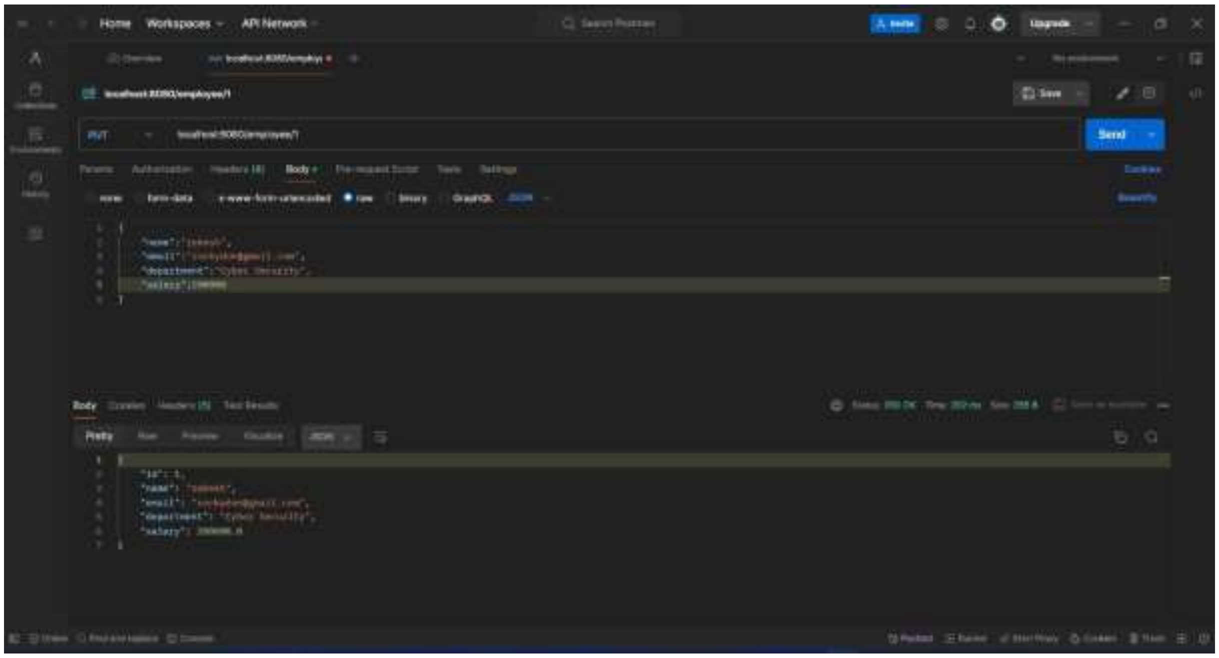


POST localhost:8080/employee

```
{
    "name":"[illegible]",
    "email":"[illegible]@gmail.com",
    "department":"Cyber Security",
    "salary":100000
}
```

```
{
    "id": 1,
    "name": "[illegible]",
    "email": "[illegible]@gmail.com",
    "department": "Cyber Security",
    "salary": 100000.0
}
```

PUT:

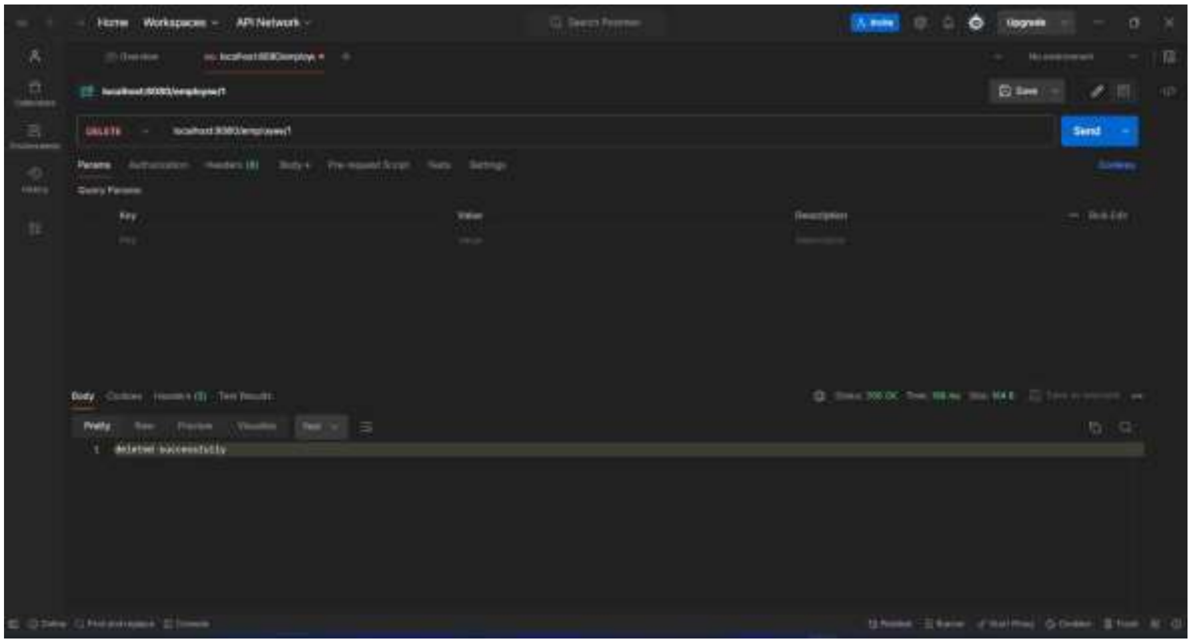


PUT localhost:8080/employee/1

```
{
    "name":"[illegible]",
    "email":"[illegible]",
    "department":"Cyber Security",
    "salary":[illegible]
}
```

```
{
    "id": 1,
    "name": "[illegible]",
    "email": "[illegible]",
    "department": "Cyber Security",
    "salary": [illegible]
}
```

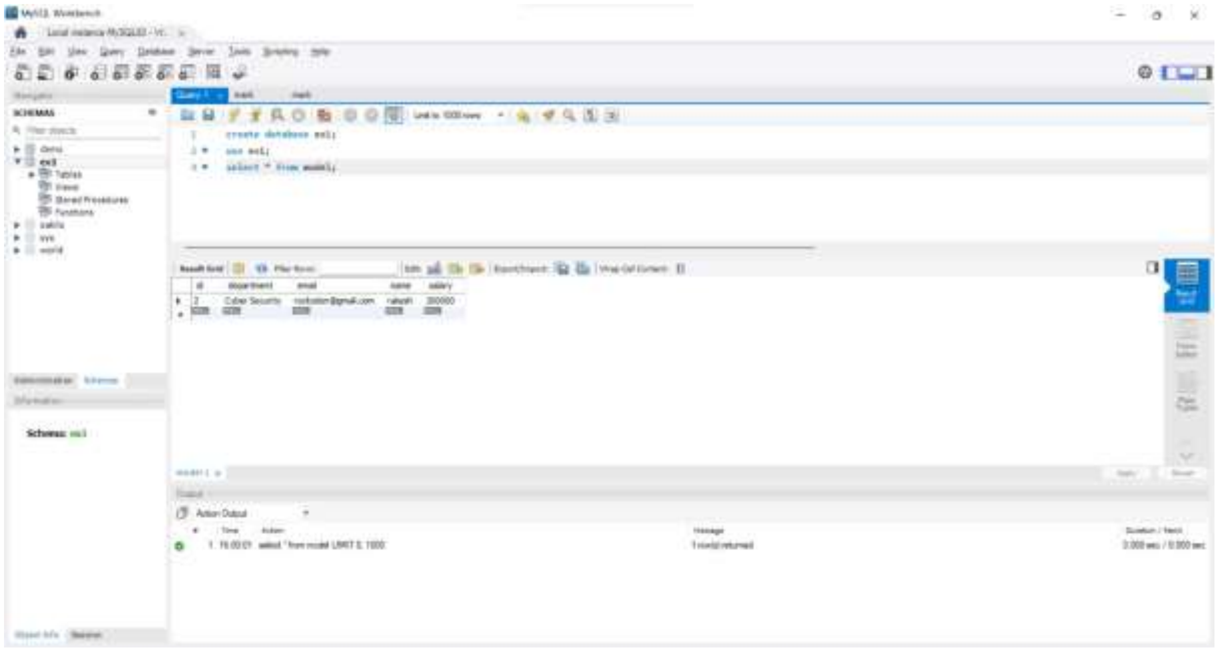
DELETE:



DELETE localhost:8080/employee/1

```
deleted successfully
```

Database:



```
create database ex1;
use ex1;
select * from model;
```

```
id  department      email                  name    salary
2   Cyber Security  [illegible]@gmail.com  rakesh  300000
NULL NULL           NULL                   NULL    NULL
```

```
#  Time      Action                                 Message            Duration / Fetch
1  15:00:01  select * from model LIMIT 0, 1000      1 row(s) returned  0.000 sec / 0.000 sec
```

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> {

}
```

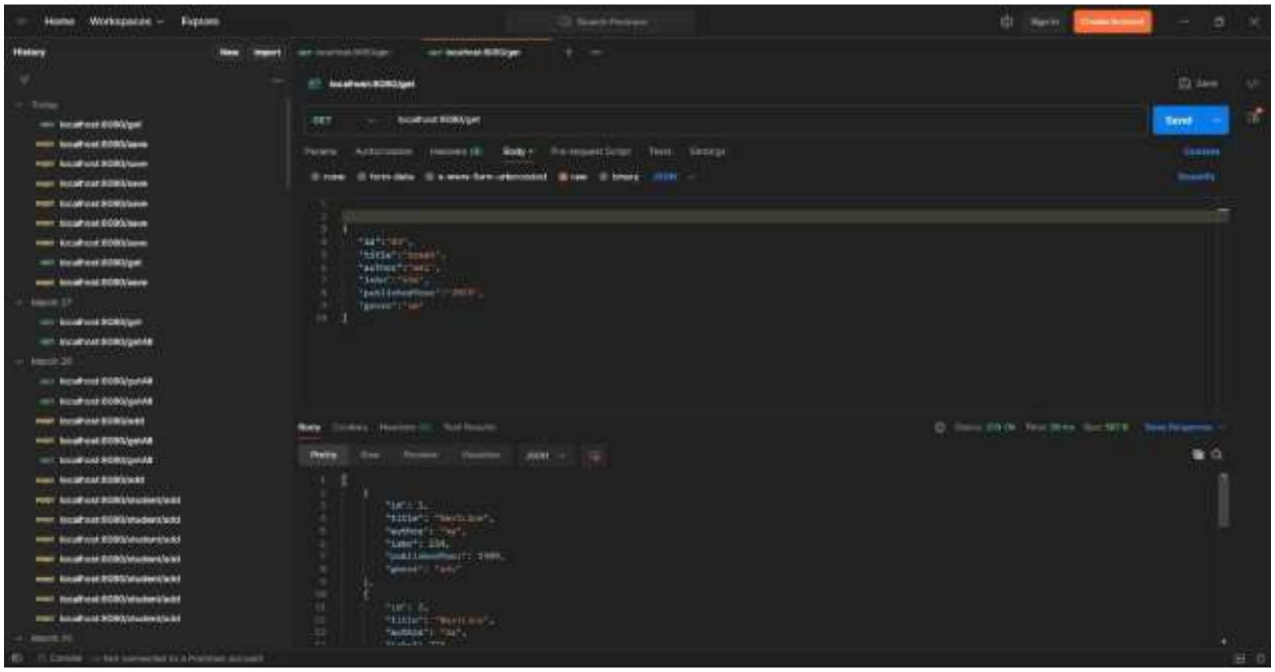
Service:

```
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:

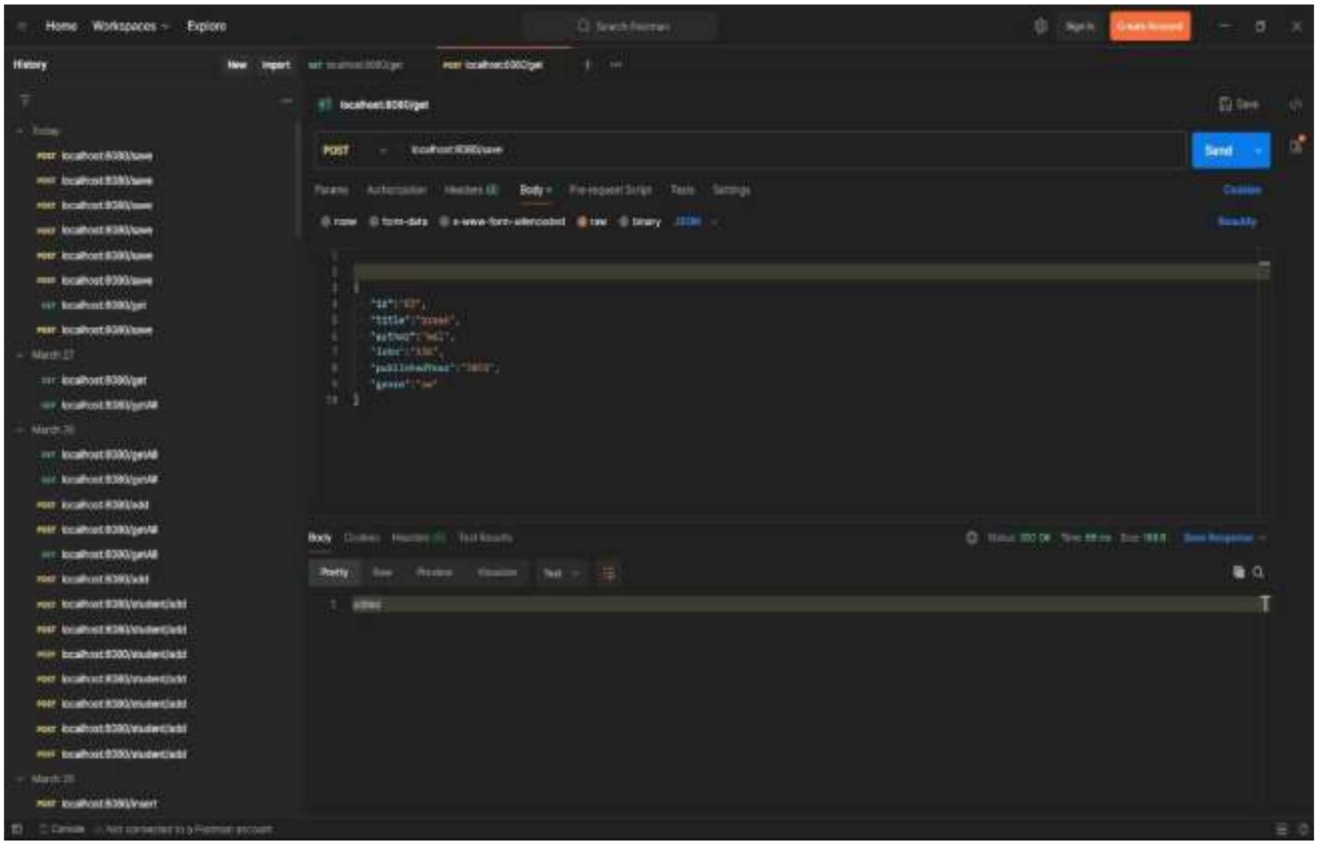
```
spring.application.name=demo-2
spring.datasource.url= jdbc:mysql://localhost:3306/lab
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database= mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=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

GET :



```
GET localhost:8080/get
```

POST:

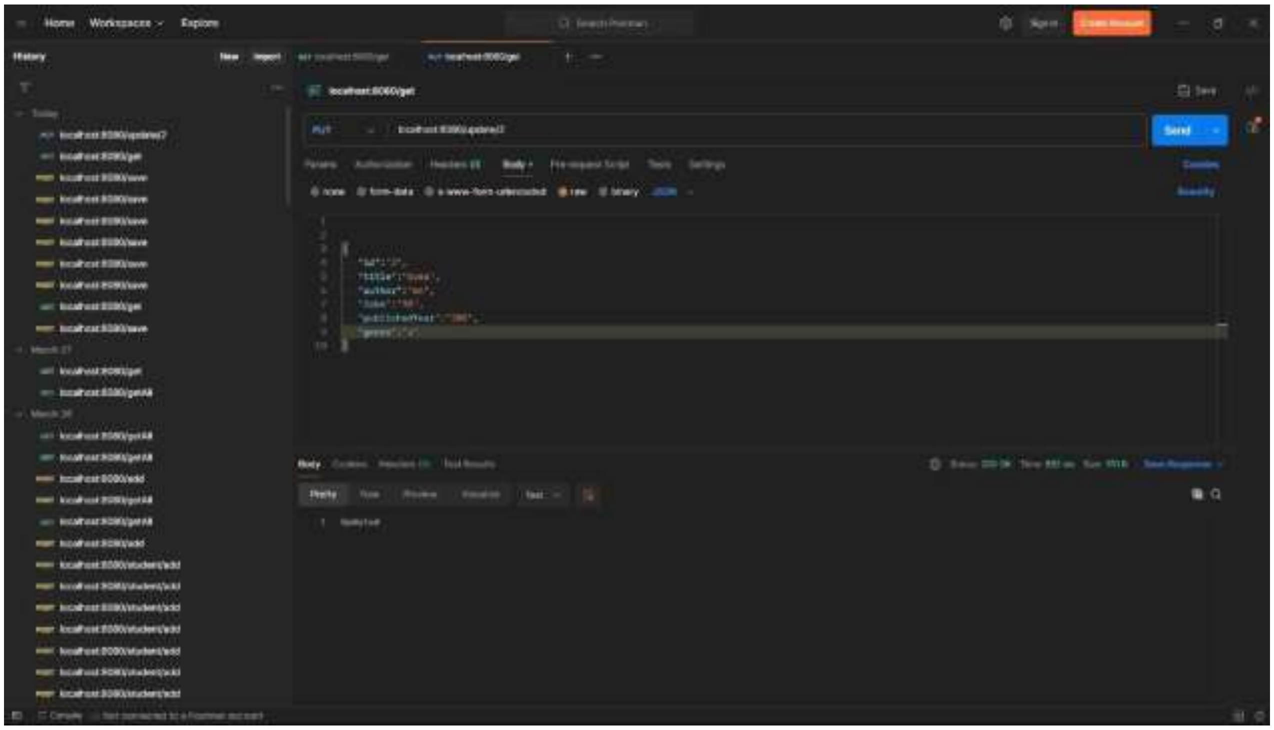


POST localhost:8080/save

```
{
"id":"[illegible]",
"title":"[illegible]",
"author":"[illegible]",
"isbn":"[illegible]",
"publishedYear":"[illegible]",
"genre":"[illegible]"
}
```

```
added
```

PUT:

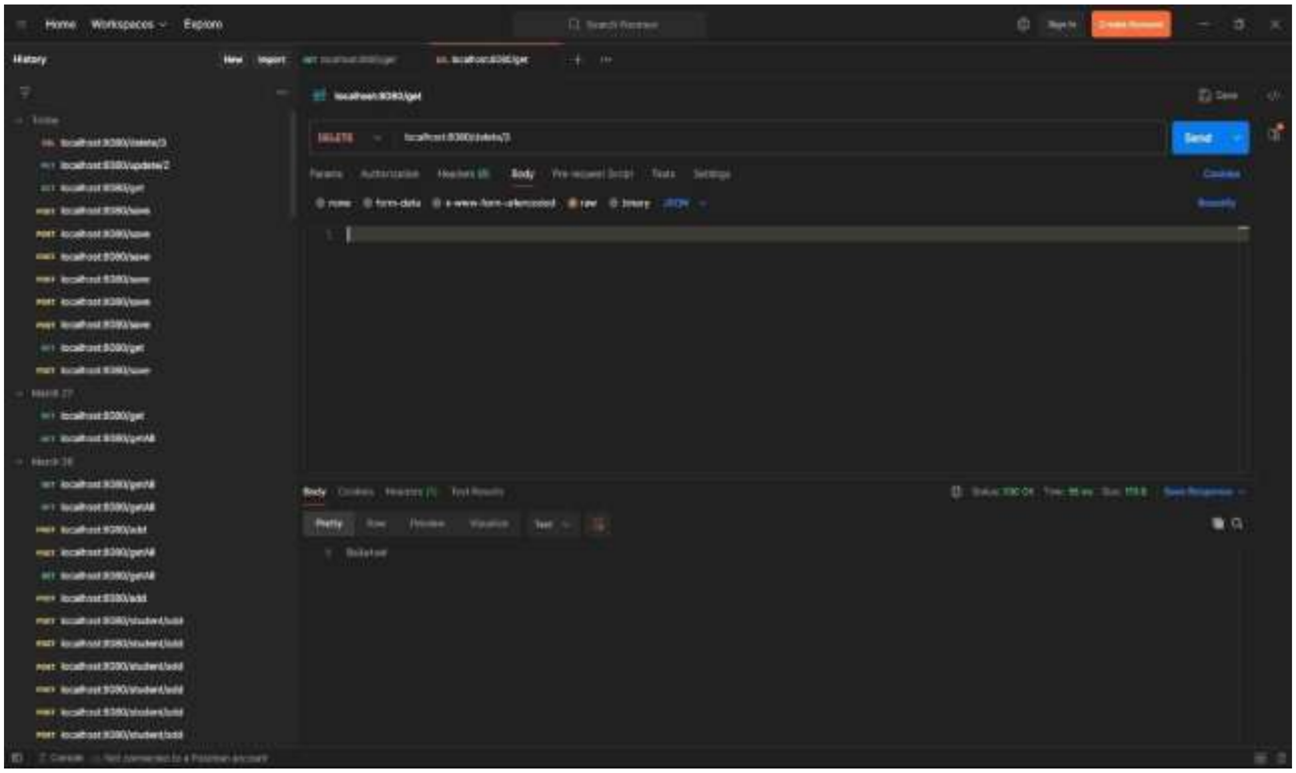


PUT localhost:8080/update/[illegible]

```
{
"id":"[illegible]",
"title":"[illegible]",
"author":"[illegible]",
"isbn":"[illegible]",
"publishedYear":"[illegible]",
"genre":"[illegible]"
}
```

```
Updated
```

DELETE:



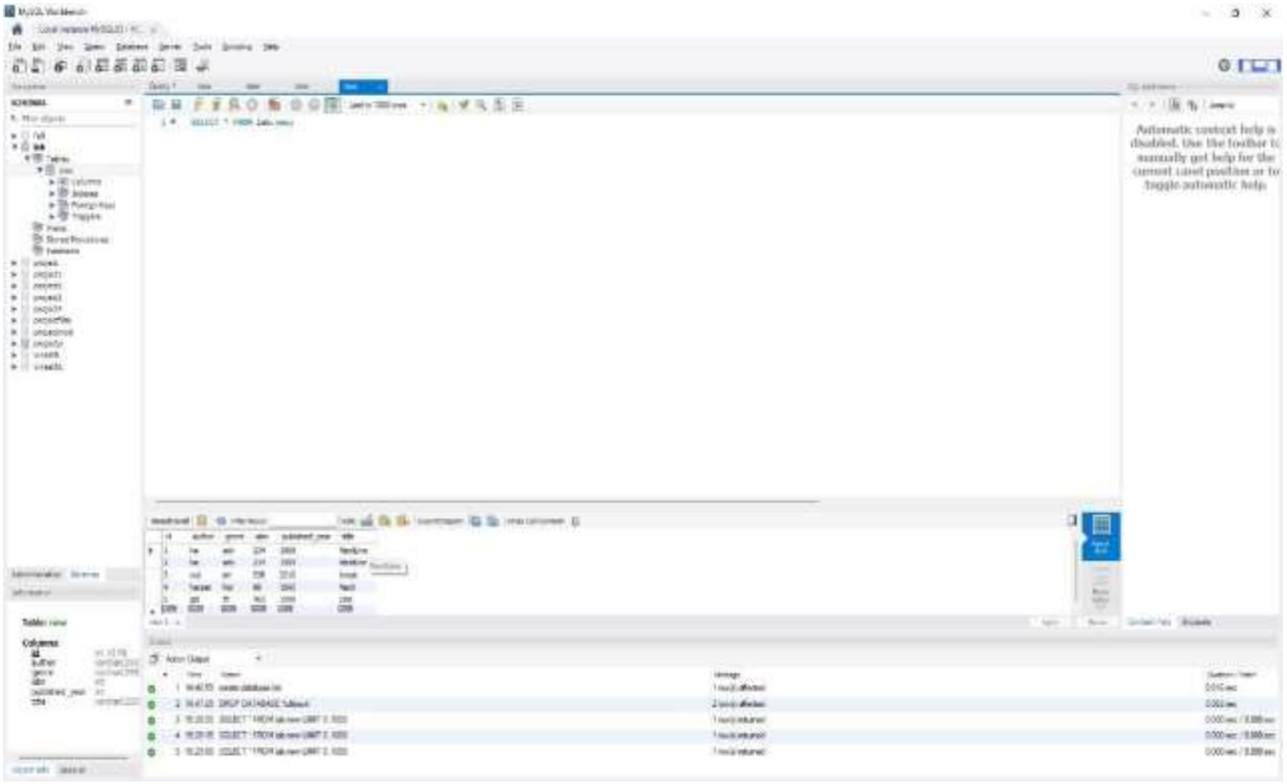
localhost:8080/get

DELETE localhost:8080/delete/3

```
1  Deleted
```

Status: 200 OK

DATABASE:



```
SELECT * FROM lab.new;
```

	id	author	genre	isbn	published_year	title
▶	1	ha	am	224	2003	[illegible]
	2	ha	am	224	2003	[illegible]
	3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
	5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
*	NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message	Duration / Fetch
1	[illegible]	create database lab	1 row(s) affected	[illegible]
2	[illegible]	DROP DATABASE [illegible]	2 row(s) affected	[illegible]
3	[illegible]	SELECT * FROM lab.new LIMIT 0, 1000	[illegible] row(s) returned	[illegible]
4	[illegible]	SELECT * FROM lab.new LIMIT 0, 1000	[illegible] row(s) returned	[illegible]
5	[illegible]	SELECT * FROM lab.new LIMIT 0, 1000	[illegible] row(s) returned	[illegible]

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

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.example.demo.Controller;

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

import com.example.demo.Entity.Staff;
import com.example.demo.repo.StaffRepository;

import jakarta.transaction.Transactional;

import java.util.List;

@RestController
@RequestMapping("/staff")
public class StaffController {

    @Autowired
    private StaffRepository staffRepository;

    @GetMapping("/")
    public List<Staff> getAllStaff() {
        return staffRepository.findAll();
    }

    @GetMapping("/{id}")
    public Staff getStaffById(@PathVariable Long id) {
        return staffRepository.findById(id).orElse(null);
    }

    @PostMapping("/")
    public Staff addStaff(@RequestBody Staff staff) {
        return staffRepository.save(staff);
    }

    @PutMapping("/{id}")
    public Staff updateStaff(@PathVariable Long id, @RequestBody Staff staffDetails) {
        Staff staff = staffRepository.findById(id).orElse(null);
        if (staff != null) {
            staff.setEmployeeId(staffDetails.getEmployeeId());
            staff.setFullName(staffDetails.getFullName());
            staff.setDivision(staffDetails.getDivision());
            staff.setEarnings(staffDetails.getEarnings());
            return staffRepository.save(staff);
        }
        return null;
    }

    @DeleteMapping("/{id}")
    @Transactional
    public String deleteStaff(@PathVariable Long id) {
        staffRepository.deleteById(id);
        return "Staff with id " + id + " deleted successfully";
    }
}
```

Model_Staff:

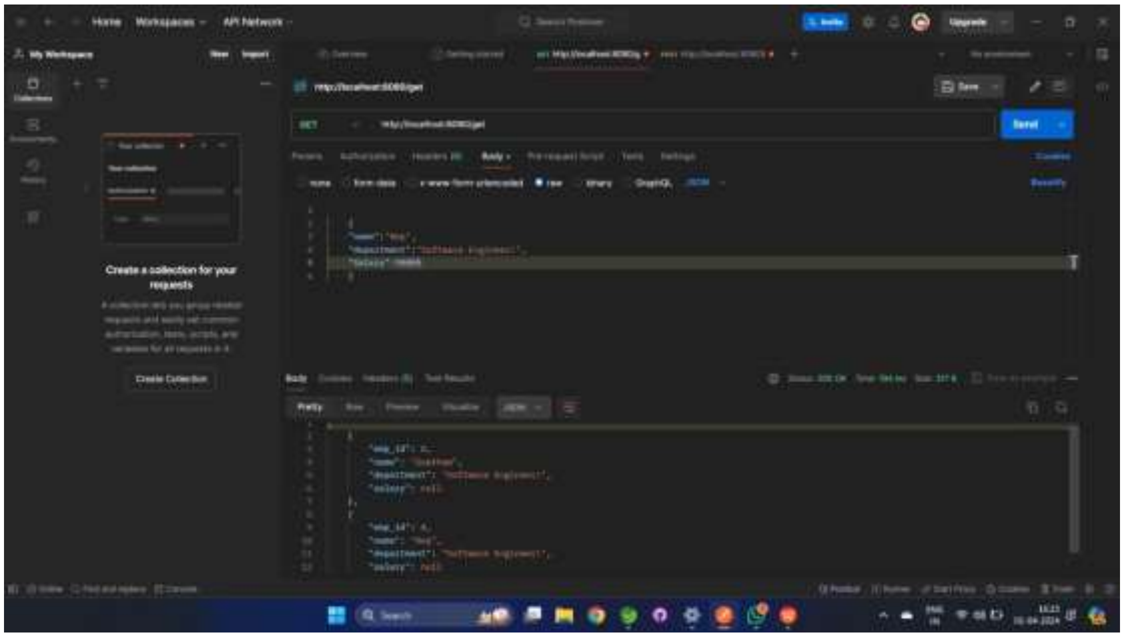
```
package com.example.demo.Entity;

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

@Entity
public class Staff {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String employeeId;
    private String fullName;
    private String division;
    private double earnings;
      public String getEmployeeId() {
            return employeeId;
      }
      public void setEmployeeId(String employeeId) {
            this.employeeId = employeeId;
      }
      public String getFullName() {
            return fullName;
      }
      public void setFullName(String fullName) {
            this.fullName = fullName;
      }
      public String getDivision() {
            return division;
      }
      public void setDivision(String division) {
            this.division = division;
      }
      public double getEarnings() {
            return earnings;
```

```
	}
	public void setEarnings(double earnings) {
		this.earnings = earnings;
	}
	public Staff(String employeeId, String fullName, String division, double earnings) {
		super();
		this.employeeId = employeeId;
		this.fullName = fullName;
		this.division = division;
		this.earnings = earnings;
	}
public Staff() {
	// TODO Auto-generated constructor stub
}
@Override
public String toString() {
	return "Staff [id=" + id + ", employeeId=" + employeeId + ", fullName=" + fullName + ", 
division=" + division
				+ ", earnings=" + earnings + "]";
}
```

Repo_Admin:

```
package com.example.demo.repo;

import java.util.Optional;

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

import com.example.demo.Entity.Staff;

public interface StaffRepository extends JpaRepository<Staff, Integer> {

	Optional<Staff> findById(Long id);

	void deleteById(Long id);

}
```

GET:



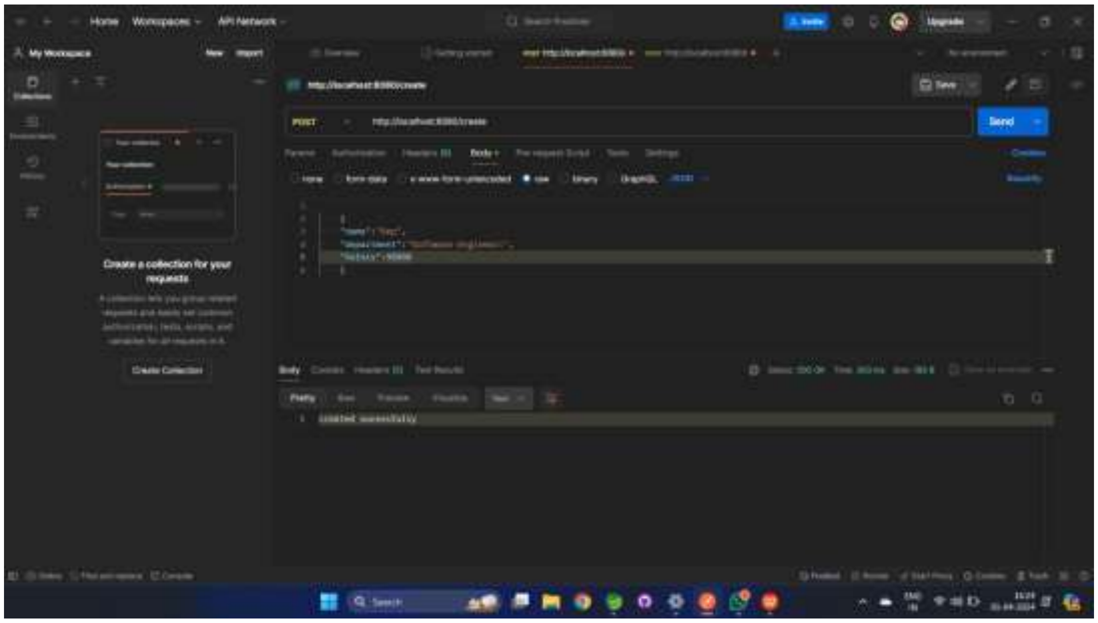
http://localhost:8080/get

GET http://localhost:8080/get

```
{
"name":"Roy",
"department":"Software Engineer",
"salary":[illegible]
}
```

```
[
  {
    "emp_id": 1,
    "name": "[illegible]",
    "department": "Software Engineer",
    "salary": null
  },
  {
    "emp_id": [illegible],
    "name": "Roy",
    "department": "Software Engineer",
    "salary": null
```

POST:



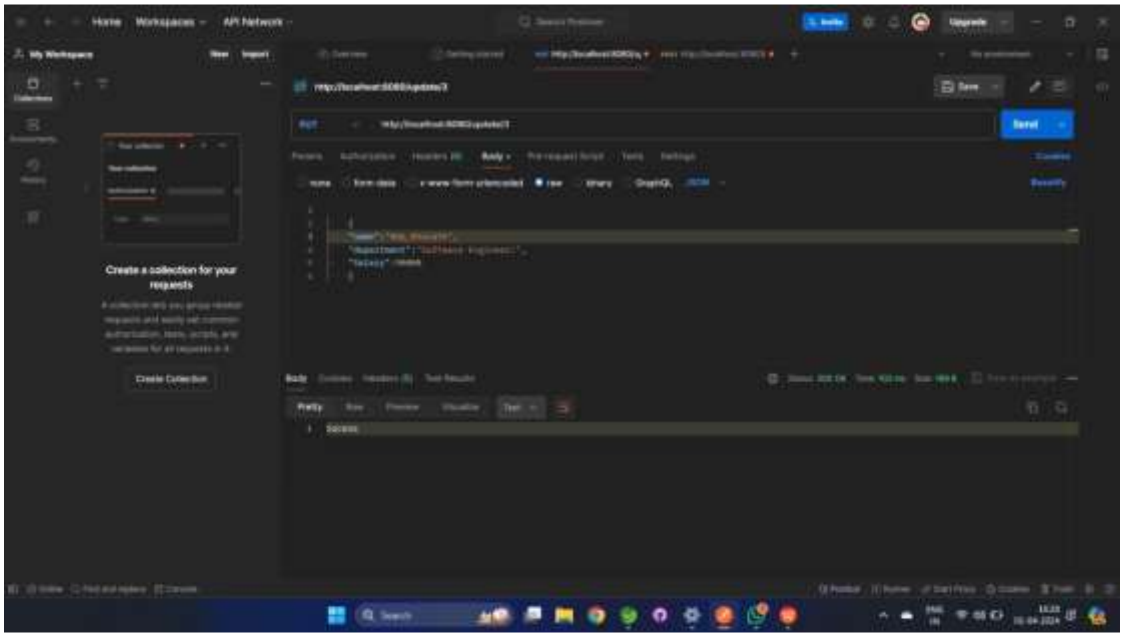
http://localhost:8080/create

POST http://localhost:8080/create

```
{
"name":"Roy",
"department":"Software Engineer",
"salary":[illegible]
}
```

```
created successfully
```

UPDATE:

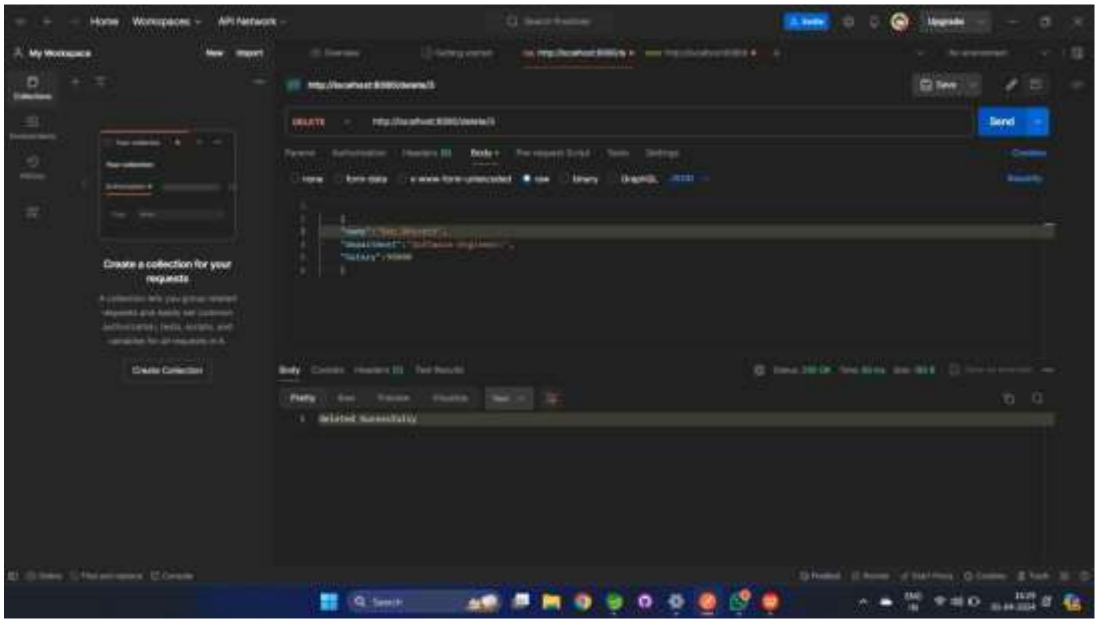


```
PUT    http://localhost:8080/update/3
```

```
{
"name":"[illegible]",
"department":"Software Engineer",
"salary":[illegible]
}
```

```
Success
```

DELETE:

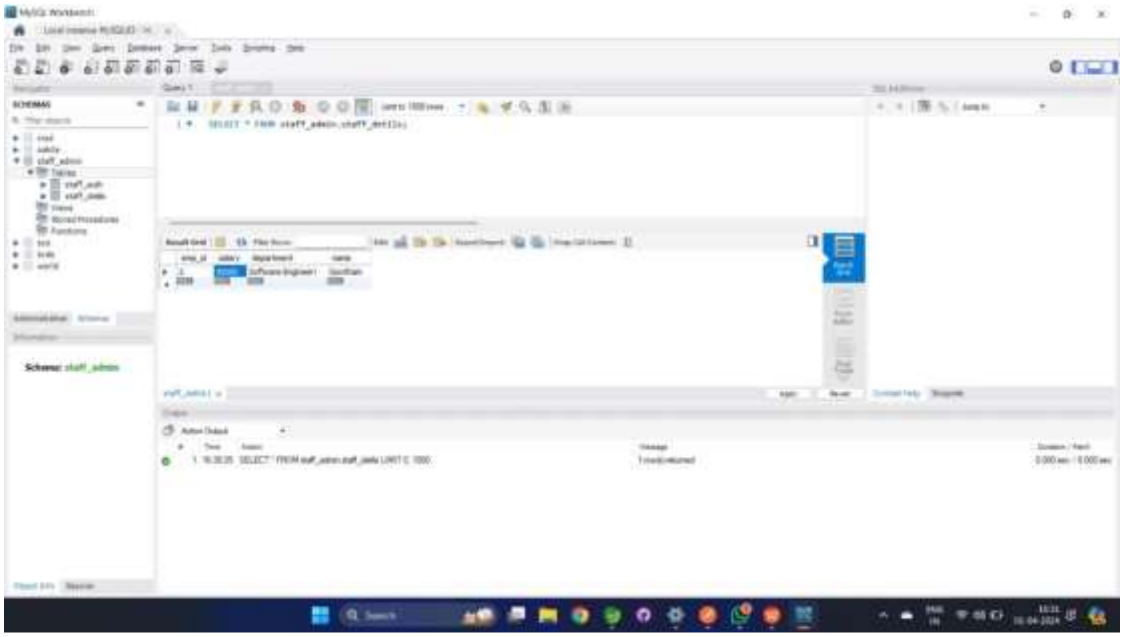


```
DELETE    http://localhost:8080/delete/3
```

```
{
"name":"[illegible]",
"department":"Software Engineer",
"salary":[illegible]
}
```

```
deleted Successfully
```

DB:



```
1  SELECT * FROM staff_admin.staff_detils;
```

emp_id	salary	department	name
1	[illegible]	Software Engineer	Gowtham
NULL	NULL	NULL	NULL

```
#  Time      Action                                                   Message            Duration / Fetch
1  16:30:35  SELECT * FROM staff_admin.staff_detils LIMIT 0, 1000     1 row(s) returned  0.000 sec / 0.000 sec
```

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;
	}
}
```

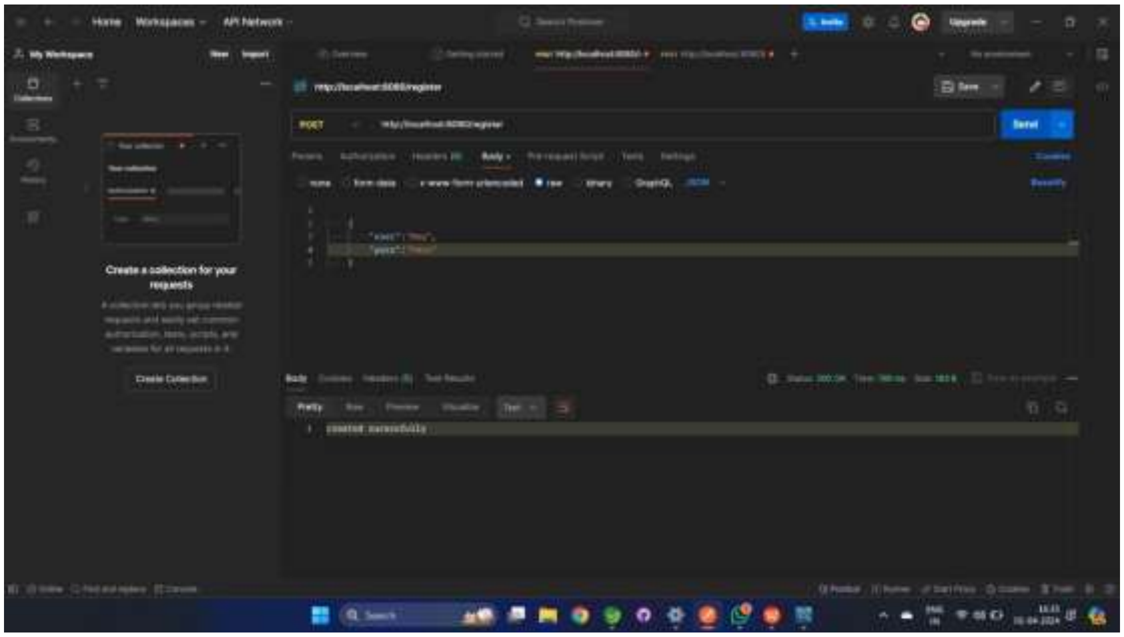
Repo_Auth:

```
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:

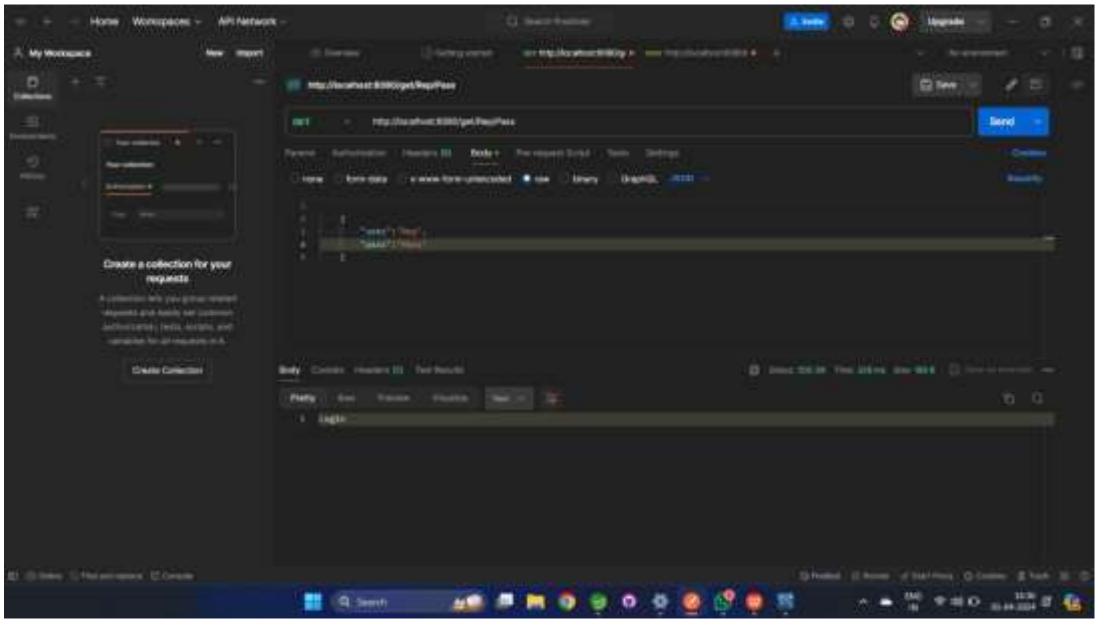


POST http://localhost:8080/register

```
{
    "user": [illegible],
    "pass": [illegible]
}
```

```
created successfully
```

Get (if user name and password is correct):

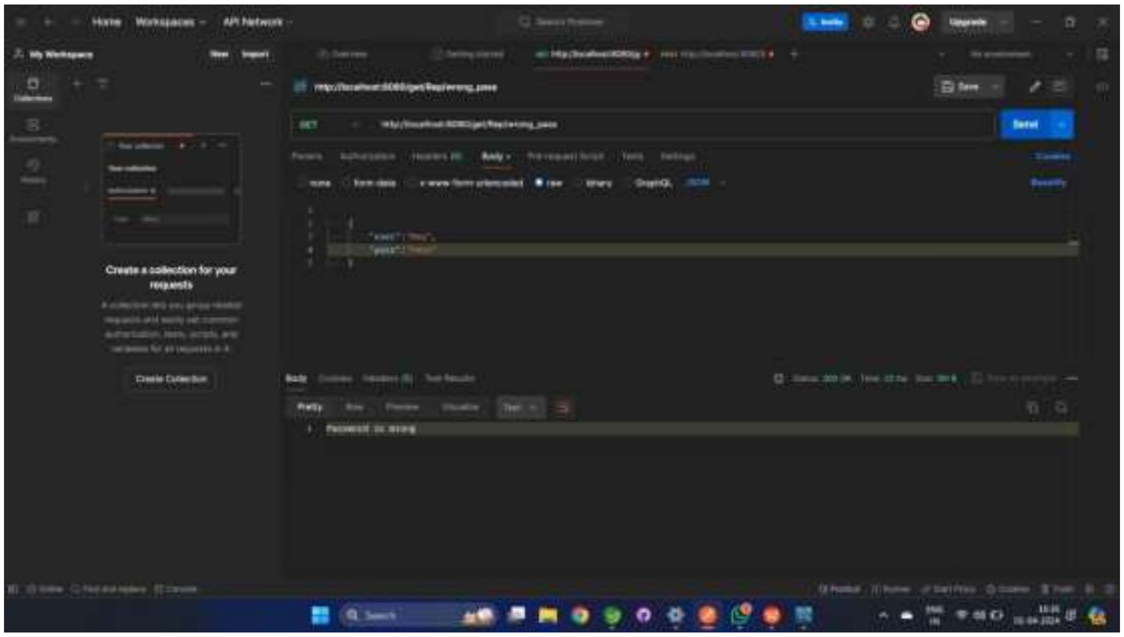


GET http://localhost:8080/get/RegPass

```
{
    "user": [illegible],
    "pass": [illegible]
}
```

```
Login
```

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



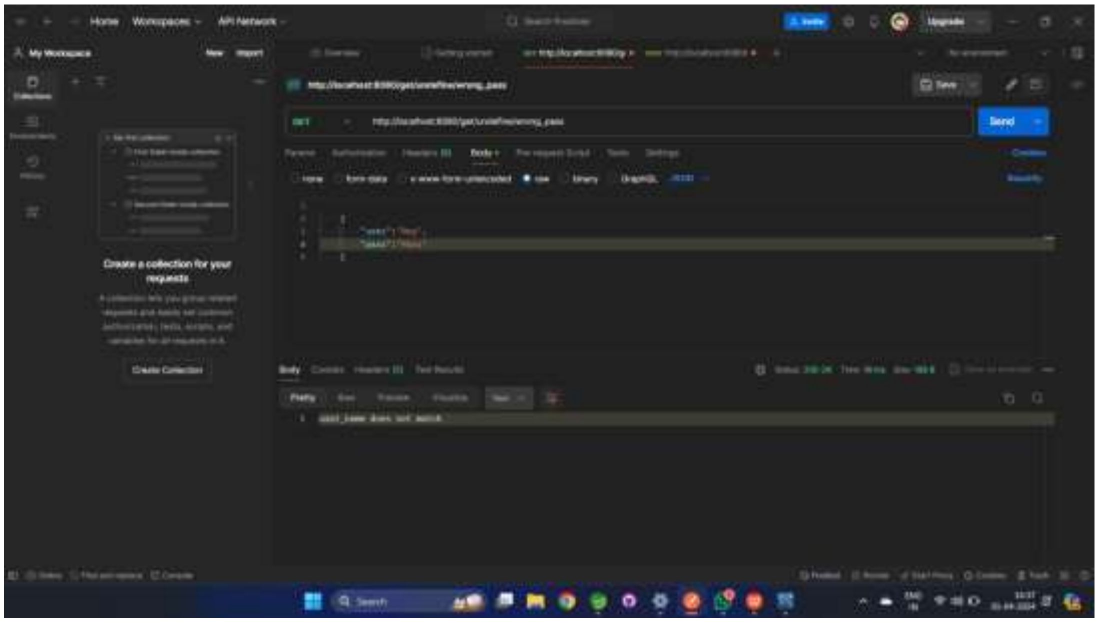
http://localhost:8080/get/Raj/wrong_pass

GET http://localhost:8080/get/Raj/wrong_pass

```
{
    "user": "[illegible]",
    "pass": "[illegible]"
}
```

```
Password is wrong
```

Get (if user name and password also wrong):



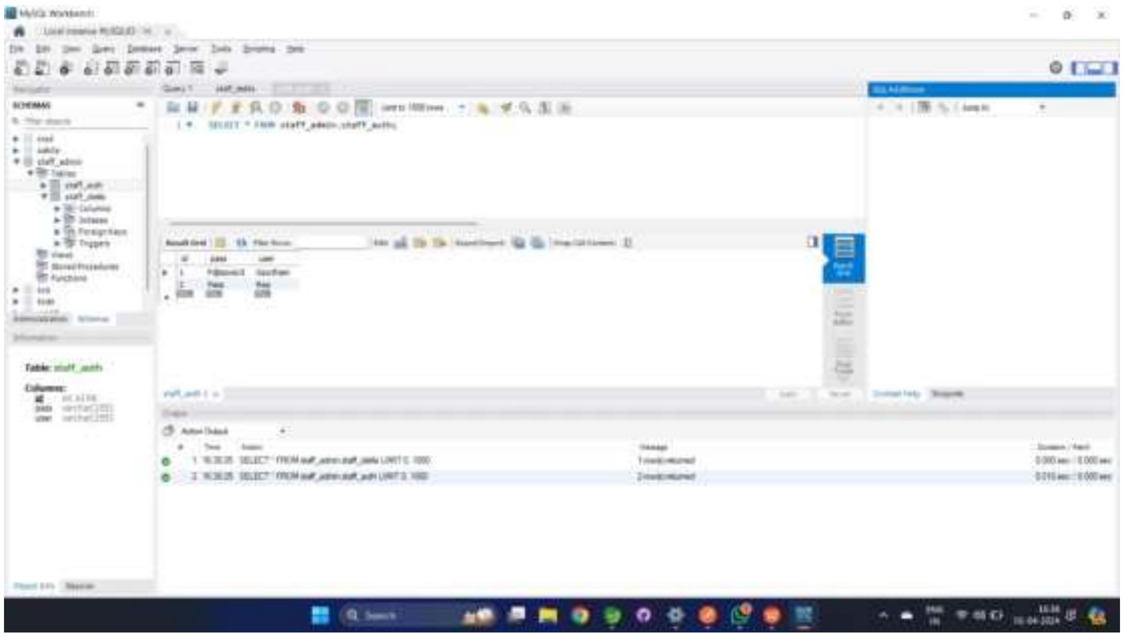
http://localhost:8080/get/undefine/wrong_pass

GET http://localhost:8080/get/undefine/wrong_pass

```
{
    "user": "[illegible]",
    "pass": "[illegible]"
}
```

```
user_name does not match
```

DB :



```
SELECT * FROM staff_admin.staff_auth;
```

	id	pass	user
▶	1	[illegible]	[illegible]
	2	Pass	[illegible]
*	NULL	NULL	NULL

```
#  Time      Action                                              Message            Duration / Fetch
1  16:38:25  SELECT * FROM staff_admin.staff_dets LIMIT 0, 1000  1 row(s) returned  0.000 sec / 0.000 sec
2  16:38:25  SELECT * FROM staff_admin.staff_auth LIMIT 0, 1000  2 row(s) returned  0.016 sec / 0.000 sec
```

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> {
}
```

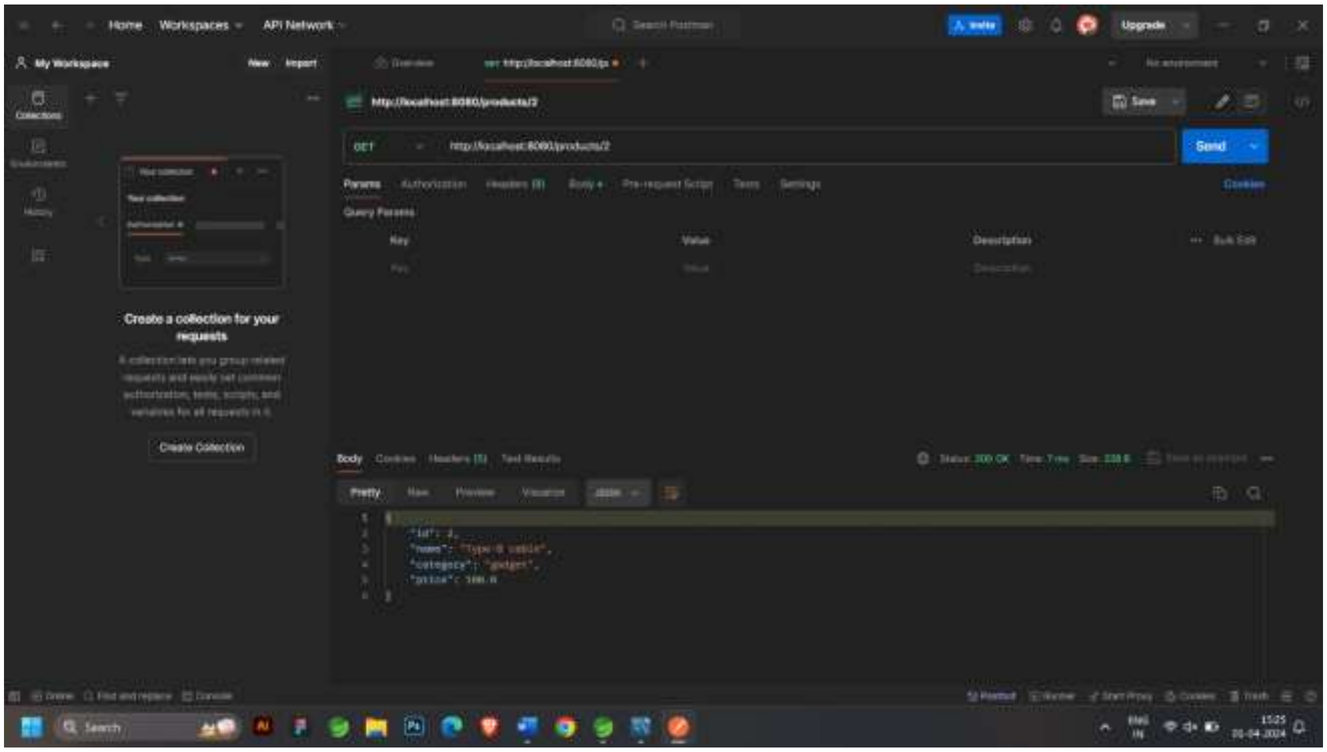
Service

```
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
```

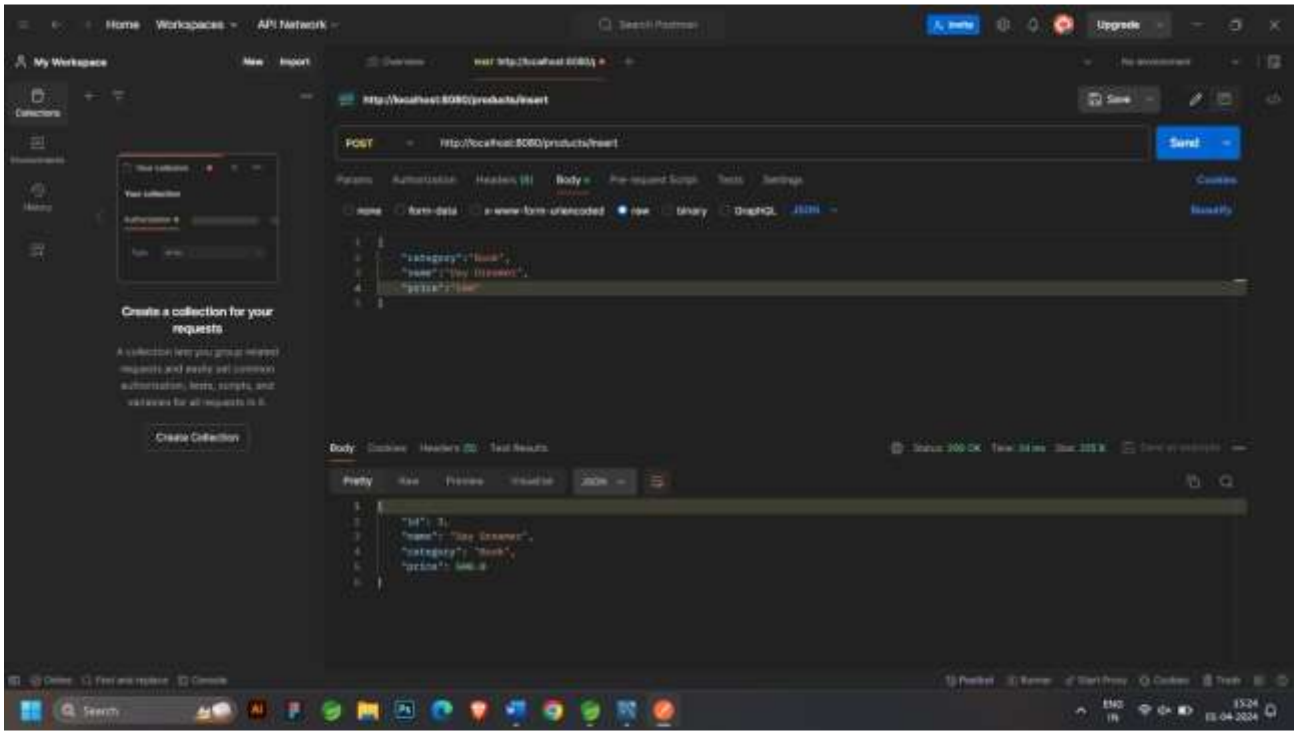
GET :



GET http://localhost:8080/products/2

```
{
    "id": 2,
    "name": "Type-B cable",
    "category": "gadget",
    "price": 100.0
}
```

POST:

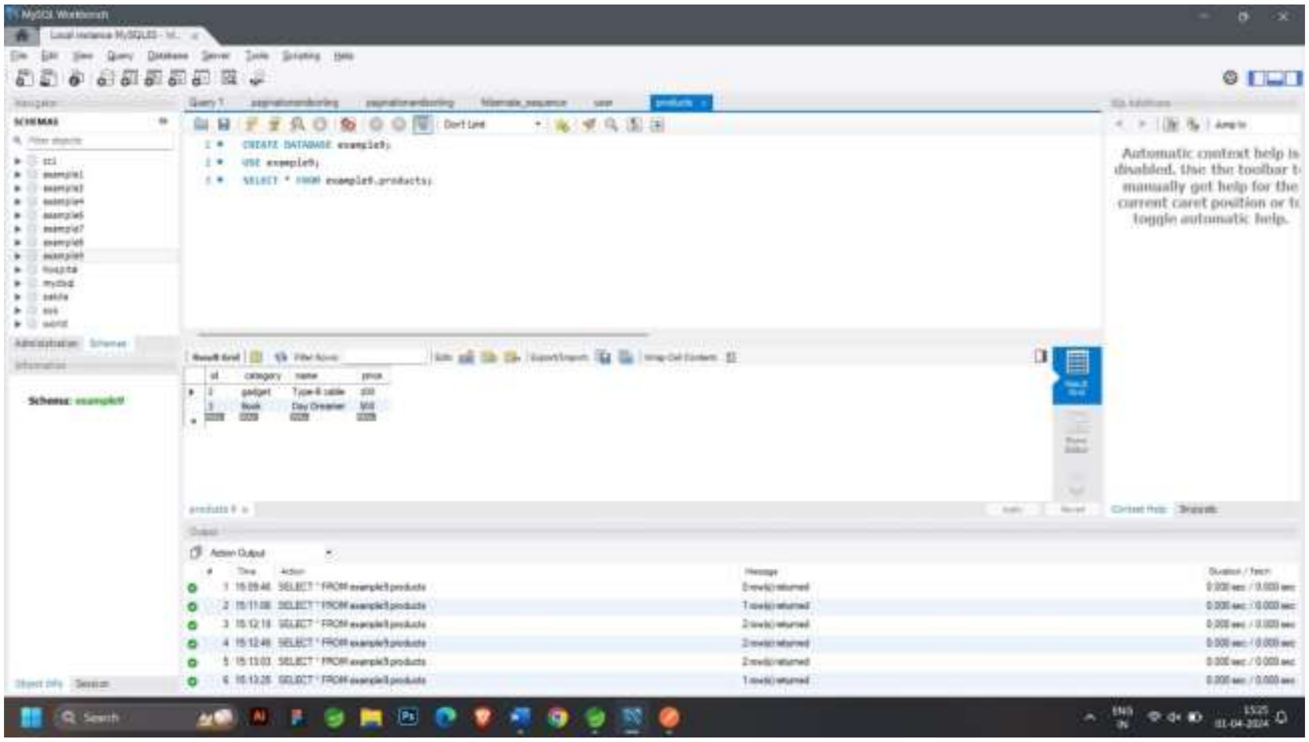


POST http://localhost:8080/products/insert

```
{
    "category":"Book",
    "name":"Day Dreamer",
    "price":"500"
}
```

```
{
    "id": 3,
    "name": "Day Dreamer",
    "category": "Book",
    "price": 500.0
}
```

PUT:



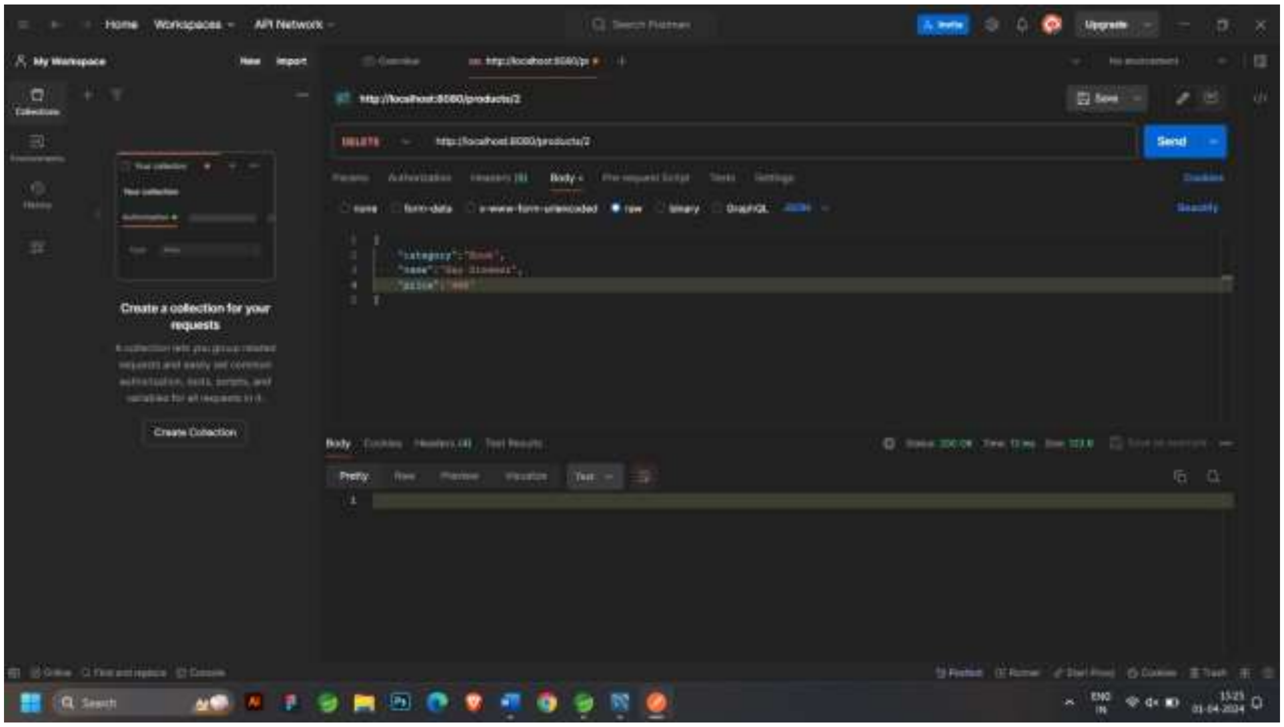
```
CREATE DATABASE example9;
USE example9;
SELECT * FROM example9.products;
```

id	category	name	price
2	gadget	Type-B cable	200
3	Book	Day Dreamer	500
NULL	NULL	NULL	NULL

```
#  Time      Action                               Message              Duration / Fetch
1  15:09:46  SELECT * FROM example9.products      0 row(s) returned    0.000 sec / 0.000 sec
2  15:11:08  SELECT * FROM example9.products      1 row(s) returned    0.000 sec / 0.000 sec
3  15:12:18  SELECT * FROM example9.products      2 row(s) returned    0.000 sec / 0.000 sec
4  15:12:46  SELECT * FROM example9.products      2 row(s) returned    0.000 sec / 0.000 sec
5  15:13:03  SELECT * FROM example9.products      2 row(s) returned    0.000 sec / 0.000 sec
6  15:13:25  SELECT * FROM example9.products      1 row(s) returned    0.000 sec / 0.000 sec
```

DELETE:

DATABASE:



DELETE http://localhost:8080/products/2

```
{
  "category":"Book",
  "name":"Day Dreamer",
  "price":"500"
}
```

Status: 200 OK

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" ;
    }
}
```

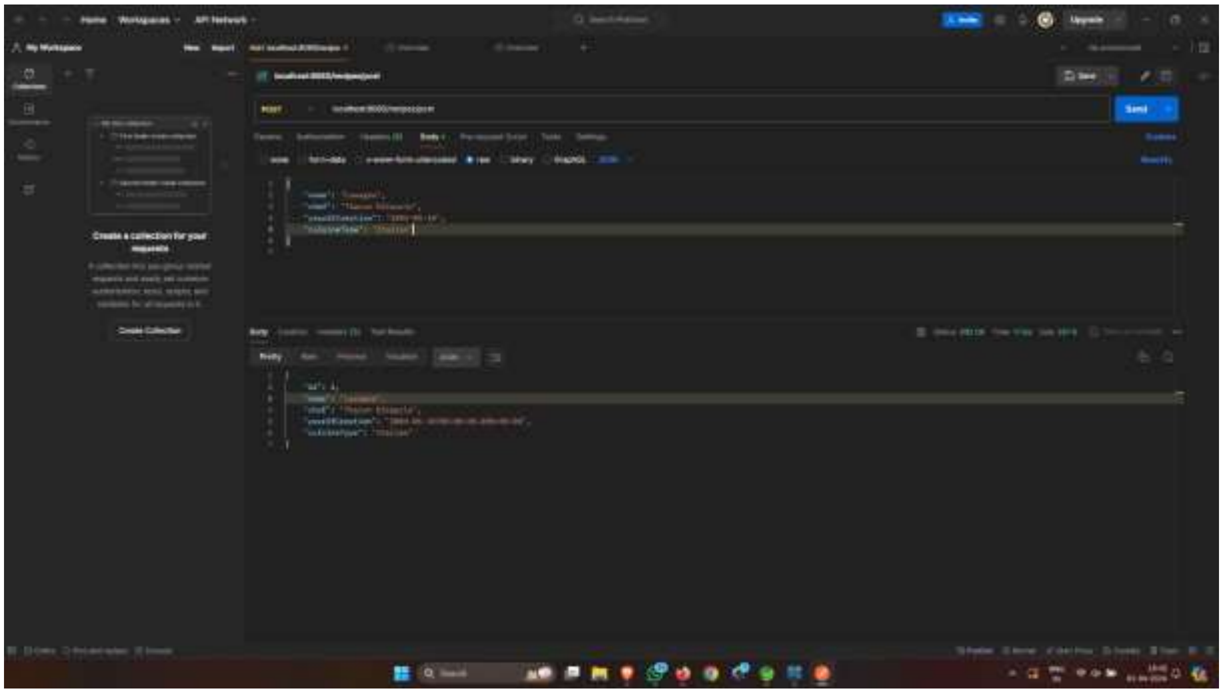
RecipeRepository.java:

```
package com.example.demo;

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

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

Application.properties:

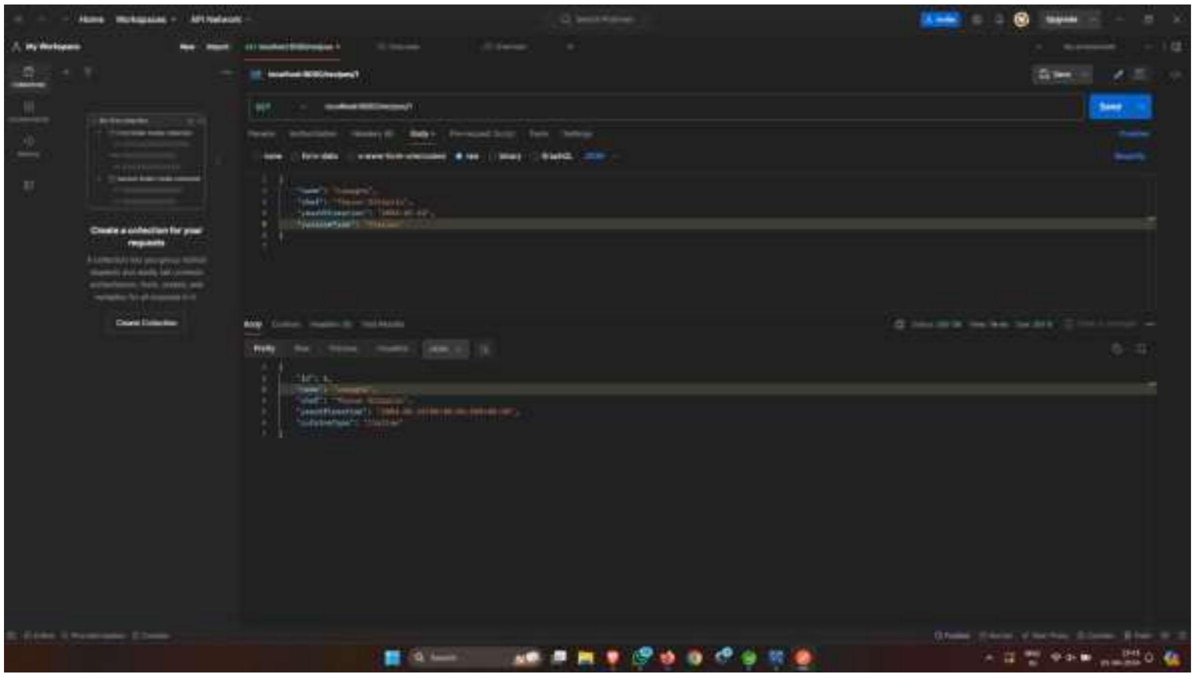
```
spring.application.name=Project5
spring.datasource.url=jdbc:mysql://localhost:3306/Project5
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.database=mysql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

POST:



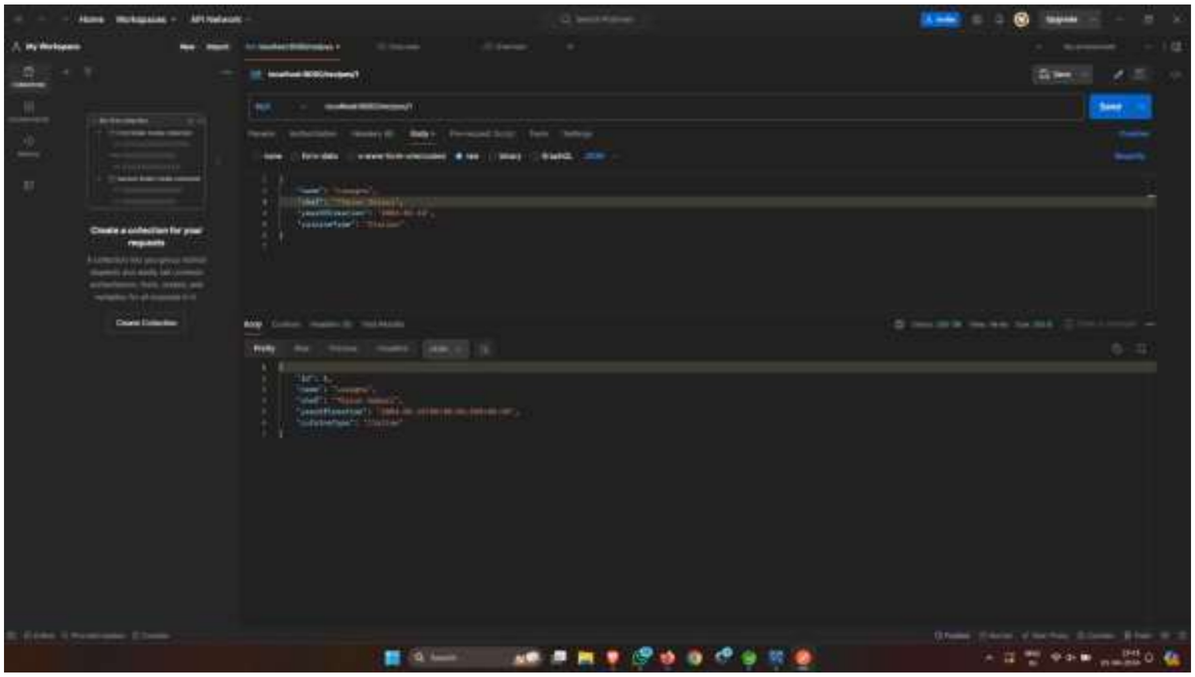
```
[illegible]
```

GET:



```
[illegible]
```

PUT:

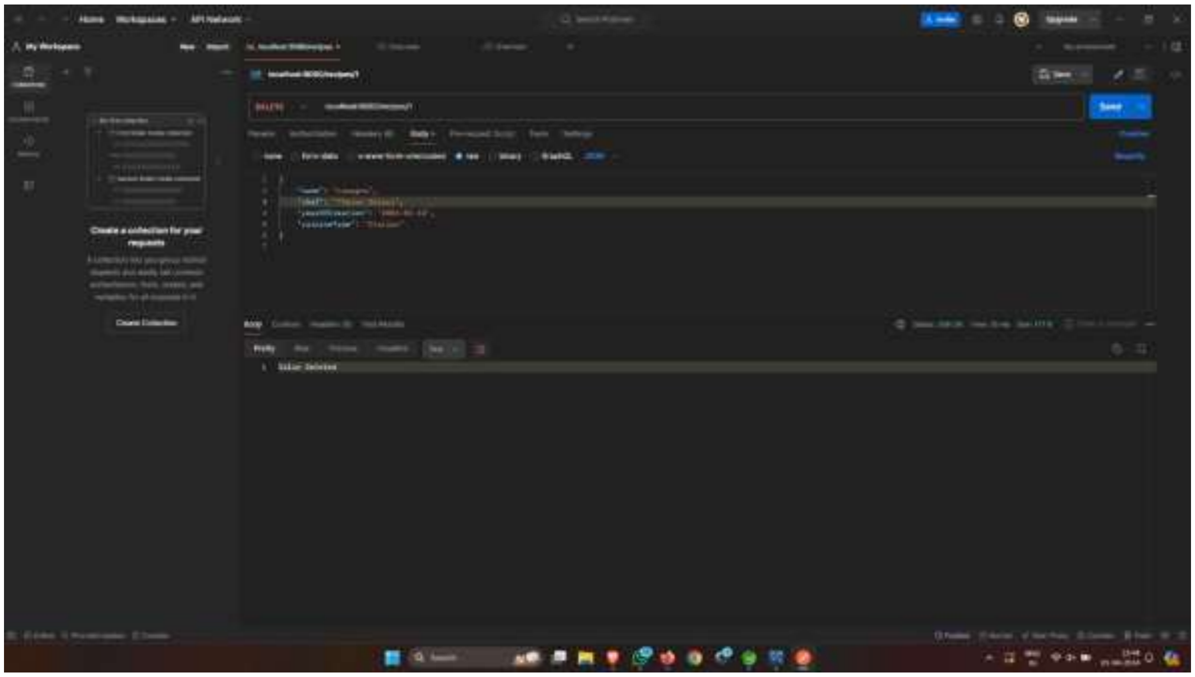


PUT localhost:8080/recipes/1

```
{
    "name": [illegible],
    "chef": [illegible],
    "yearOfCreation": [illegible],
    "cuisineType": [illegible]
}
```

```
{
    "id": 1,
    "name": [illegible],
    "chef": [illegible],
    "yearOfCreation": [illegible],
    "cuisineType": [illegible]
}
```

DELETE:

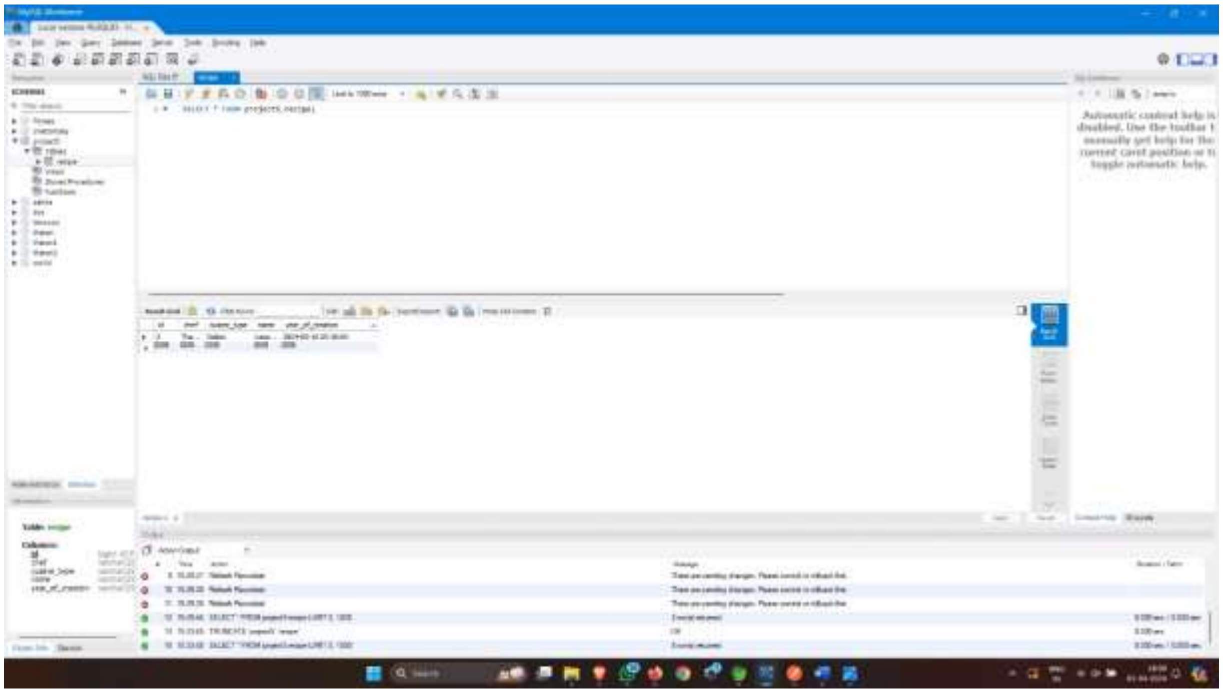


DELETE localhost:8080/recipes/1

```
{
    "name": [illegible],
    "chef": [illegible],
    "yearOfCreation": [illegible],
    "cuisineType": [illegible]
}
```

```
Value Deleted
```

DATABASE:



```
SELECT * FROM projects.recipe;
```

```
id     chef   cuisine_type  name    year_of_creation
3      The...  Italian      Lasa... 2024-02-14 20:30:00
NULL   NULL    NULL         NULL    NULL
```

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);
     }
}
```

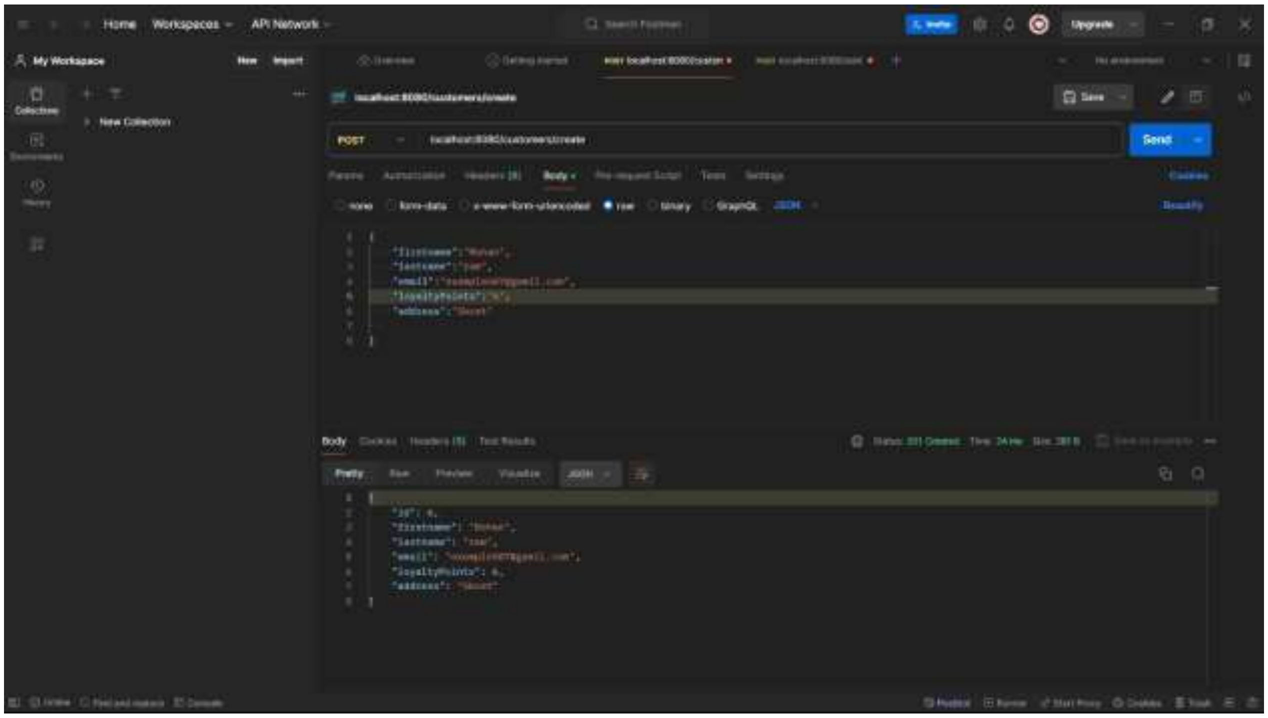
Repo Interface:

```
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:



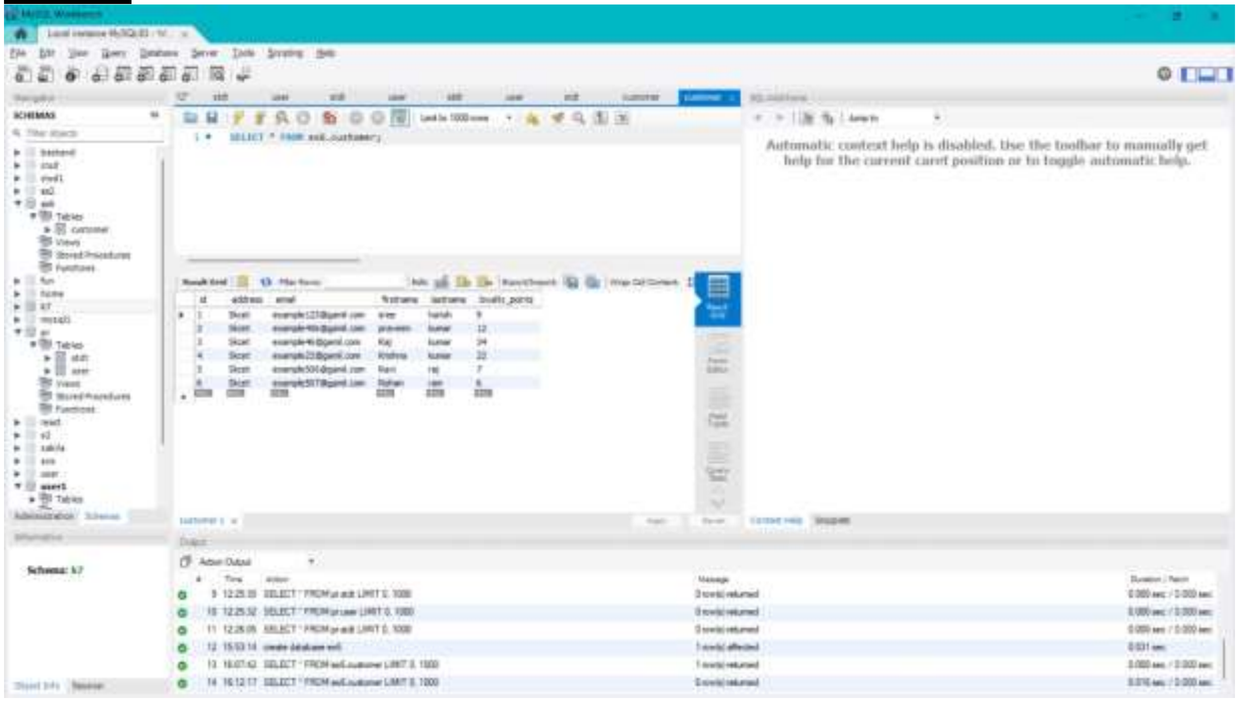
POST localhost:8080/customers/create

```
{
    "firstname":"Rohan",
    "lastname":"ram",
    "email":"example507@gmail.com",
    "loyaltyPoints":"6",
    "address":"Skcet"

}
```

Status: 201 Created Time: 24 ms Size: [illegible]

```
{
    "id": 6,
    "firstname": "Rohan",
    "lastname": "ram",
    "email": "example507@gmail.com",
    "loyaltyPoints": 6,
    "address": "Skcet"
}
```

Database:



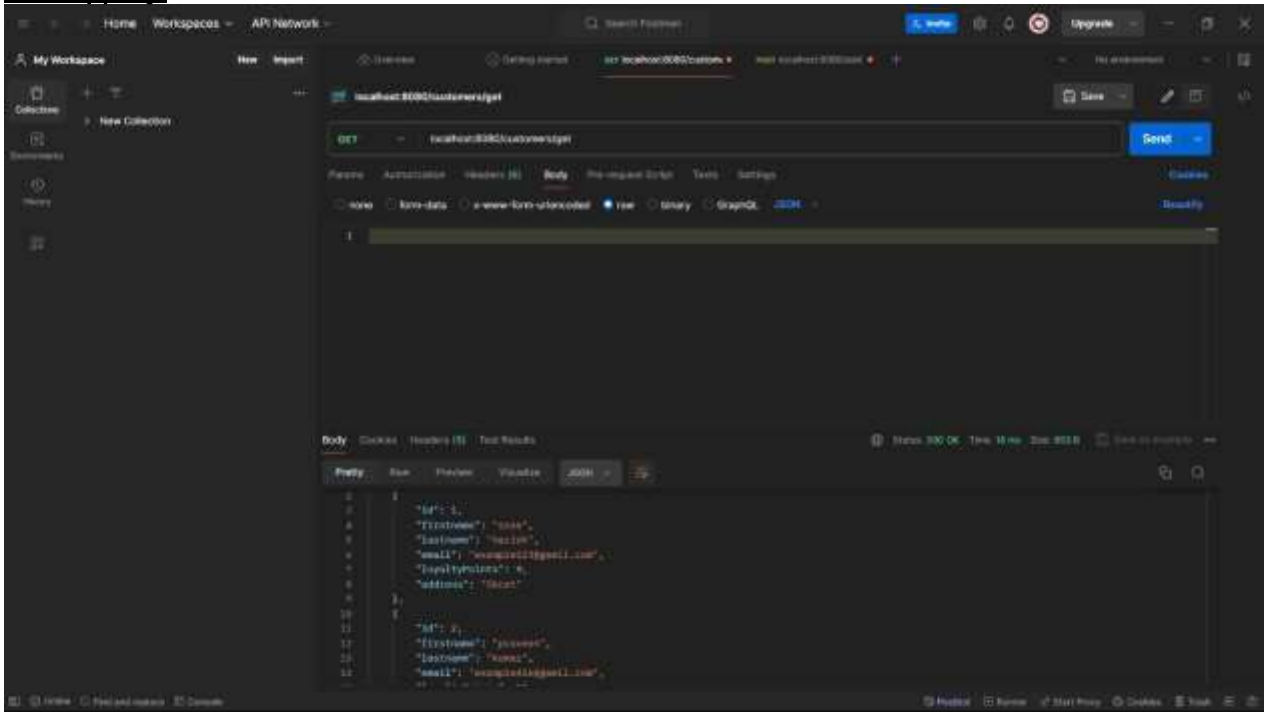
```
SELECT * FROM ex6.customer;
```

id	address	email	firstname	lastname	loyalty_points
1	Skcet	example123@gamil.com	siva	harish	9
2	Skcet	example406@gamil.com	praveen	kumar	12
3	Skcet	example46@gamil.com	Raj	kumar	34
4	Skcet	example23@gamil.com	Krishna	kumar	22
5	Skcet	example500@gamil.com	Ram	raj	7
6	Skcet	example507@gamil.com	Rohan	ram	6
NULL	NULL	NULL	NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
#   Time      Action                                      Message            Duration / Fetch
9   12:25:33  SELECT * FROM pr.skill LIMIT 0, 1000        0 row(s) returned  0.000 sec / 0.000 sec
10  12:25:32  SELECT * FROM pr.user LIMIT 0, 1000         0 row(s) returned  0.000 sec / 0.000 sec
11  12:26:05  SELECT * FROM pr.skill LIMIT 0, 1000        0 row(s) returned  0.000 sec / 0.000 sec
12  15:53:14  create database ex6                         1 row(s) affected  0.031 sec
13  16:07:42  SELECT * FROM ex6.customer LIMIT 0, 1000    1 row(s) returned  0.000 sec / 0.000 sec
14  16:12:17  SELECT * FROM ex6.customer LIMIT 0, 1000    6 row(s) returned  0.016 sec / 0.000 sec
```

GetMapping:

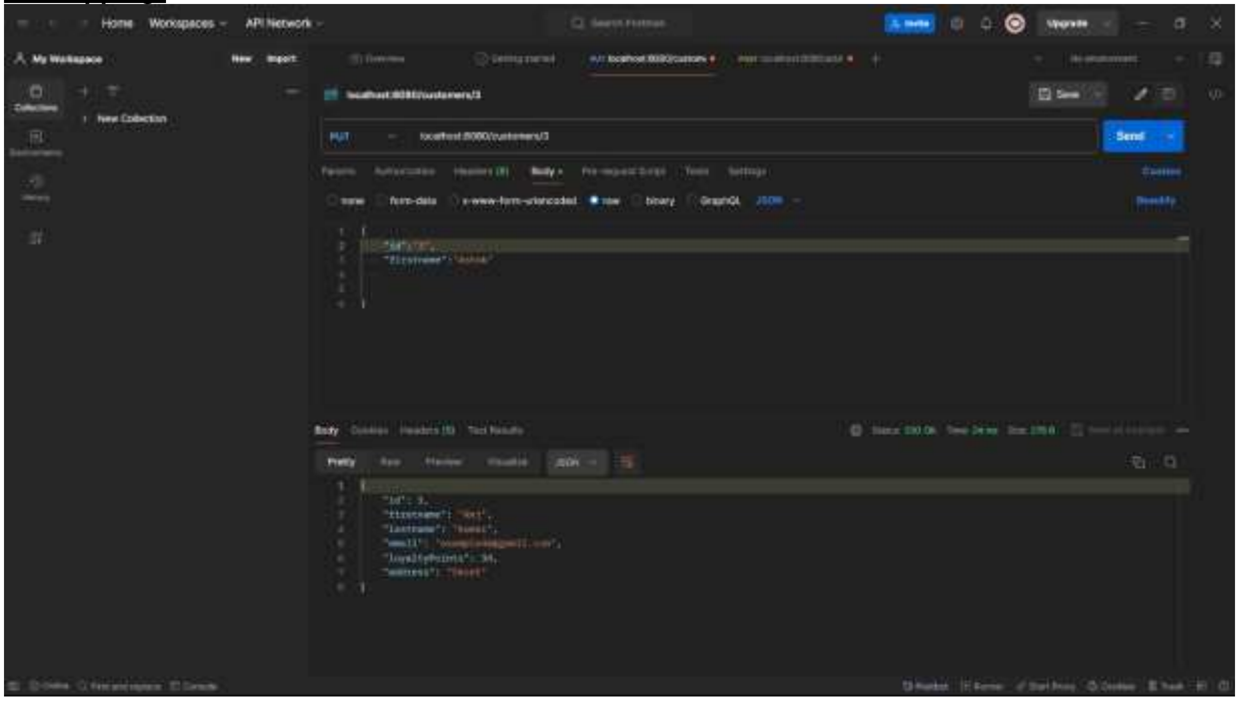


GET localhost:8080/customers/get

```
[
    {
        "id": 1,
        "firstname": "[illegible]",
        "lastname": "[illegible]",
        "email": "[illegible]@gmail.com",
        "loyaltyPoints": [illegible],
        "address": "[illegible]"
    },
    {
        "id": 2,
        "firstname": "[illegible]",
        "lastname": "[illegible]",
        "email": "[illegible]@gmail.com",
```

Status: 200 OK

PutMapping:



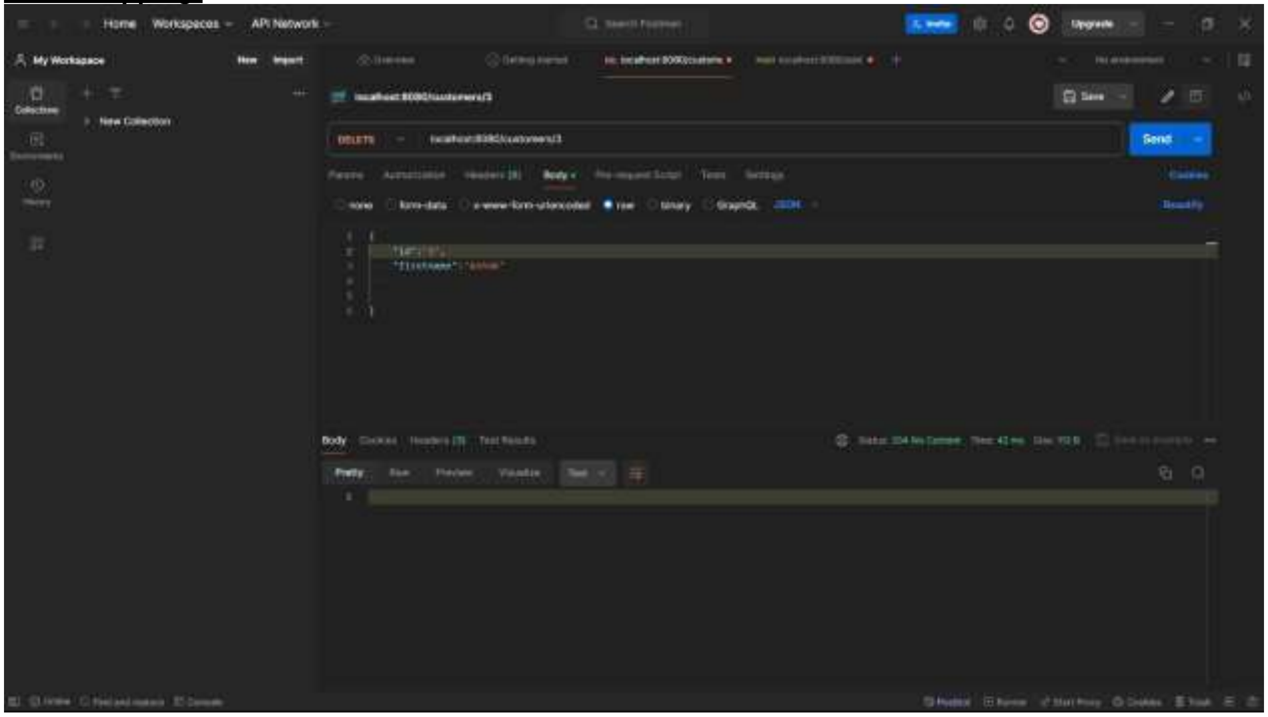
PUT localhost:8080/customers/3

```
{
  "id":"3",
  "firstname":"[illegible]"


}
```

Status: 200 OK

```
{
    "id": 3,
    "firstname": "[illegible]",
    "lastname": "[illegible]",
    "email": "[illegible]@gmail.com",
    "loyaltyPoints": 34,
    "address": "[illegible]"
}
```

DeleteMapping:

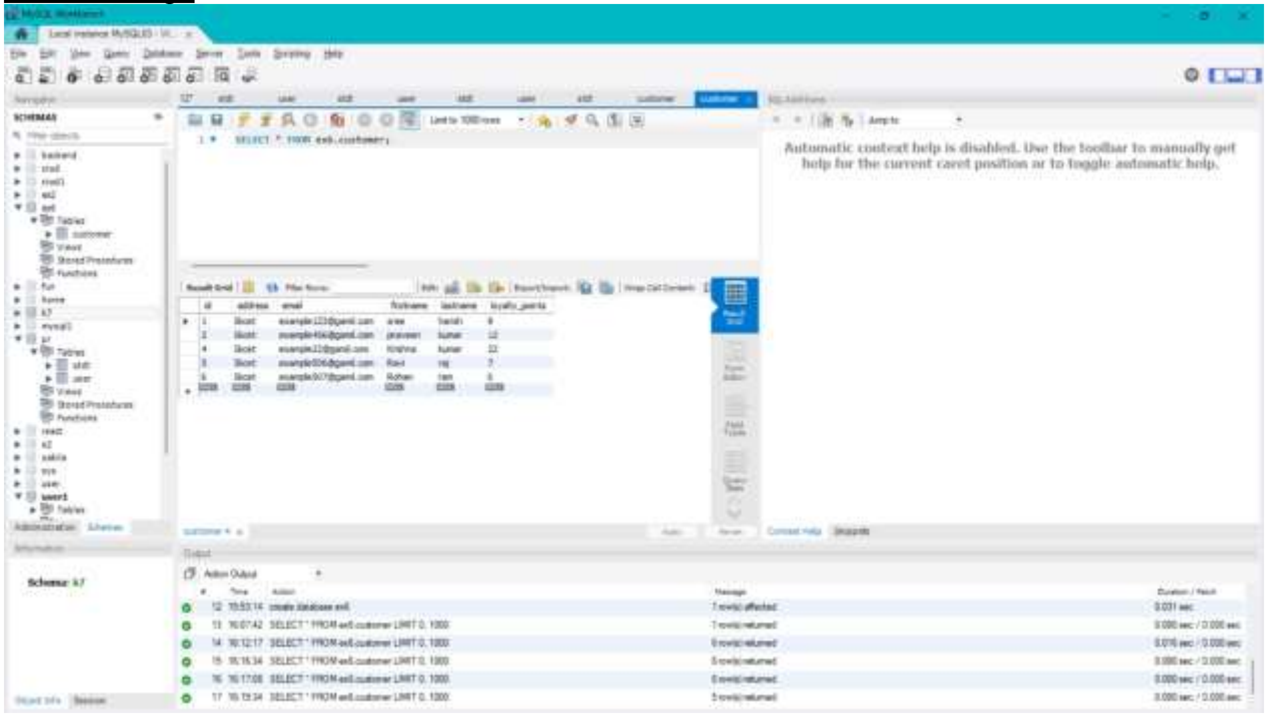


```
DELETE    localhost:8080/customers/3
```

```
{
    "id":"3",
    "firstname":"ashok"

}
```

Status: 204 No Content

After Deleting :



```
SELECT * FROM ex6.customer;
```

id	address	email	firstname	lastname	loyalty_points
1	Skcet	example123@gamil.com	aree	harish	8
2	Skcet	example456@gamil.com	praveen	kumar	12
4	Skcet	example22@gamil.com	Krishna	kumar	22
5	Skcet	example506@gamil.com	Ravi	raj	7
6	Skcet	example507@gamil.com	Rohan	ram	5
NULL	NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                     Message              Duration / Fetch
12  15:53:14  create database ex6                        1 row(s) affected    0.031 sec
13  16:07:42  SELECT * FROM ex6.customer LIMIT 0, 1000   1 row(s) returned    0.000 sec / 0.000 sec
14  16:12:17  SELECT * FROM ex6.customer LIMIT 0, 1000   6 row(s) returned    0.016 sec / 0.000 sec
15  16:16:34  SELECT * FROM ex6.customer LIMIT 0, 1000   5 row(s) returned    0.000 sec / 0.000 sec
16  16:17:08  SELECT * FROM ex6.customer LIMIT 0, 1000   6 row(s) returned    0.000 sec / 0.000 sec
17  16:19:34  SELECT * FROM ex6.customer LIMIT 0, 1000   5 row(s) returned    0.000 sec / 0.000 sec
```

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> {
}
```

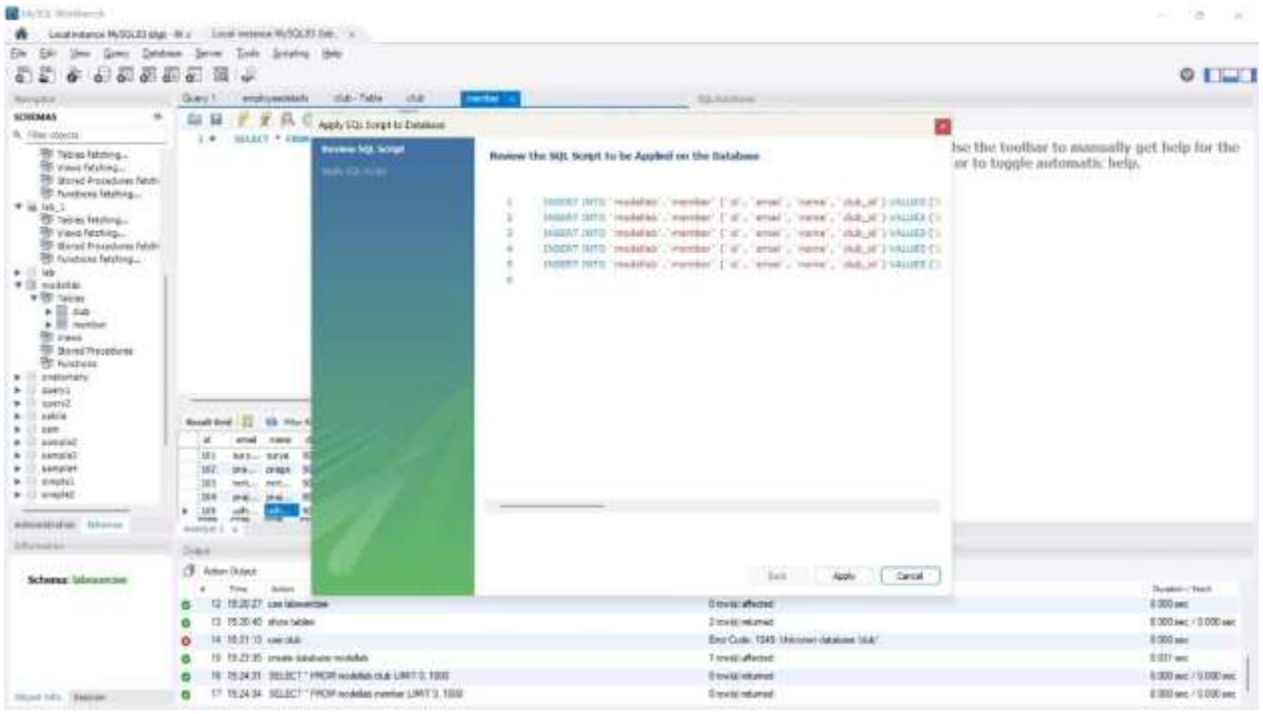
ClubRepository.java:

```
package com.example.pipe;

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

Application.properties:

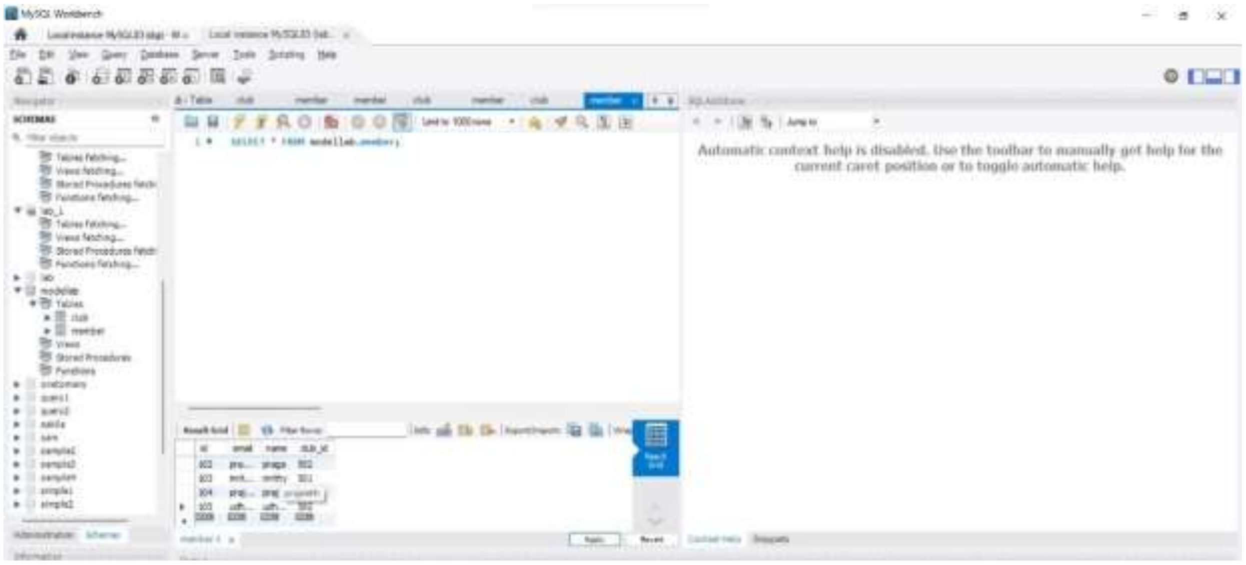
```
spring.application.name=lab
spring.datasource.url= jdbc:mysql://localhost:3306/modellab
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.database= mysql
spring.jpa.generate-ddl=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=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

OUTPUT:



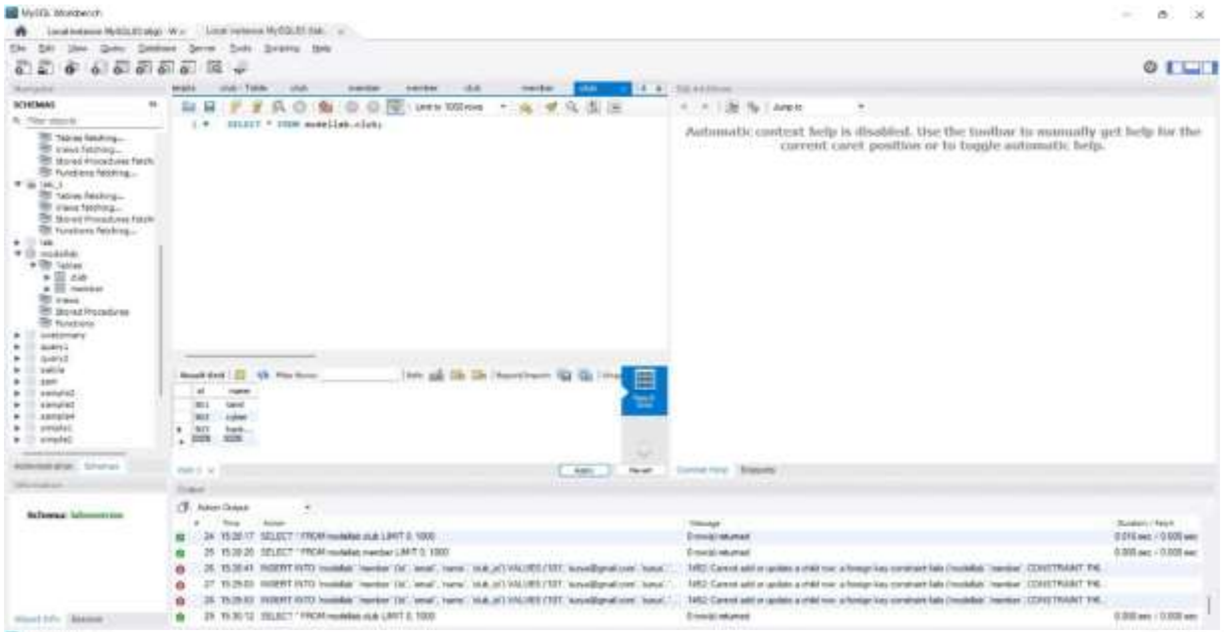
```
INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES (
INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES (
INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES (
INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES (
INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES (
```

```
12 15:20:27 use labexercise                                   0 row(s) affected
13 15:20:40 show tables                                       2 row(s) returned
14 15:21:13 use club                                          Error Code: 1049. Unknown database 'club'
15 15:23:35 create database modellab                          1 row(s) affected
16 15:24:31 SELECT * FROM modellab.club LIMIT 0, 1000         0 row(s) returned
17 15:24:34 SELECT * FROM modellab.member LIMIT 0, 1000       0 row(s) returned
```



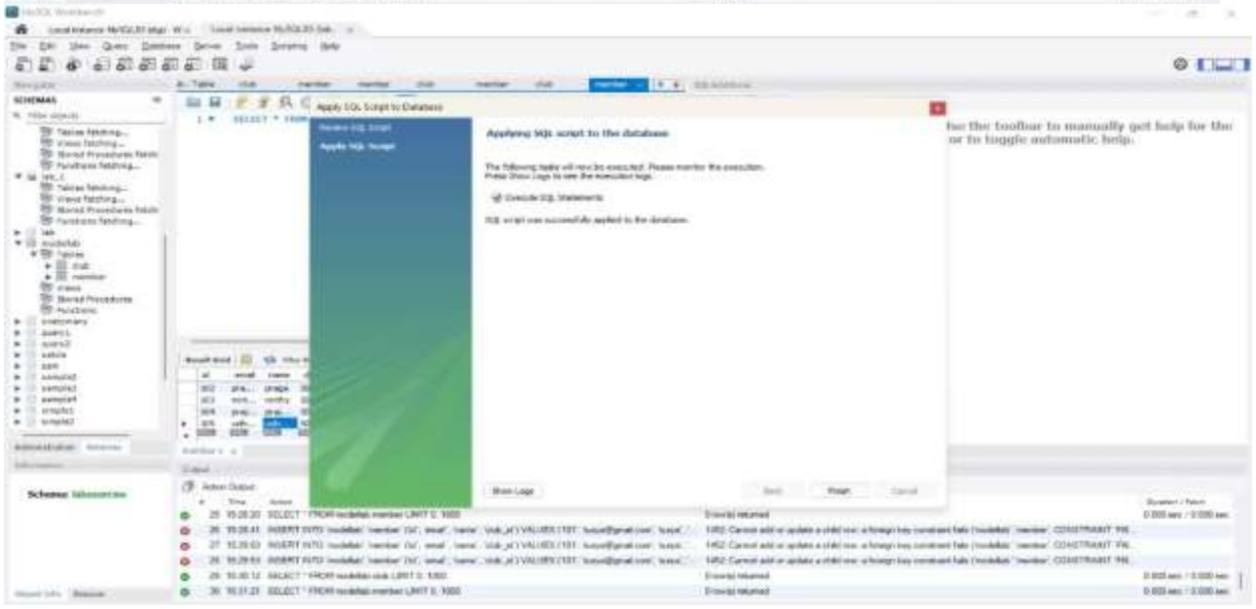
```
SELECT * FROM modellab.member;
```

id	email	name	club_id
102	pra...	praga	502
103	smit...	smithy	501
104	praj...	praj	[illegible]
105	udh...	udh...	502
NULL	NULL	NULL	NULL



```
SELECT * FROM modellab.club;
```

```
24 15:20:17 SELECT * FROM modellab.club LIMIT 0, 1000      0 row(s) returned
25 15:20:20 SELECT * FROM modellab.member LIMIT 0, 1000    0 row(s) returned
26 15:28:41 INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES ('101', 'suya@gmail.com', 'suya'...   1452: Cannot add or update a child row: a foreign key constraint fails (`modellab`.`member`, CONSTRAINT `FK...
27 15:29:03 INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES ('101', 'suya@gmail.com', 'suya'...   1452: Cannot add or update a child row: a foreign key constraint fails (`modellab`.`member`, CONSTRAINT `FK...
28 15:29:53 INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES ('101', 'suya@gmail.com', 'suya'...   1452: Cannot add or update a child row: a foreign key constraint fails (`modellab`.`member`, CONSTRAINT `FK...
29 15:30:12 SELECT * FROM modellab.club LIMIT 0, 1000      0 row(s) returned
```



Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution. Press Show Logs to see the execution logs.

Execute SQL Statements

SQL script was successfully applied to the database.

```
25 15:20:20 SELECT * FROM modellab.member LIMIT 0, 1000    0 row(s) returned
26 15:28:41 INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES ('101', 'suya@gmail.com', 'suya'...   1452: Cannot add or update a child row: a foreign key constraint fails (`modellab`.`member`, CONSTRAINT `FK...
27 15:29:03 INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES ('101', 'suya@gmail.com', 'suya'...   1452: Cannot add or update a child row: a foreign key constraint fails (`modellab`.`member`, CONSTRAINT `FK...
28 15:29:53 INSERT INTO `modellab`.`member` (`id`, `email`, `name`, `club_id`) VALUES ('101', 'suya@gmail.com', 'suya'...   1452: Cannot add or update a child row: a foreign key constraint fails (`modellab`.`member`, CONSTRAINT `FK...
29 15:30:12 SELECT * FROM modellab.club LIMIT 0, 1000      0 row(s) returned
30 15:31:23 SELECT * FROM modellab.member LIMIT 0, 1000    0 row(s) returned
```

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);
    }
}
```

<u>2) DoctorService.java</u>

```
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);
    }
}
```

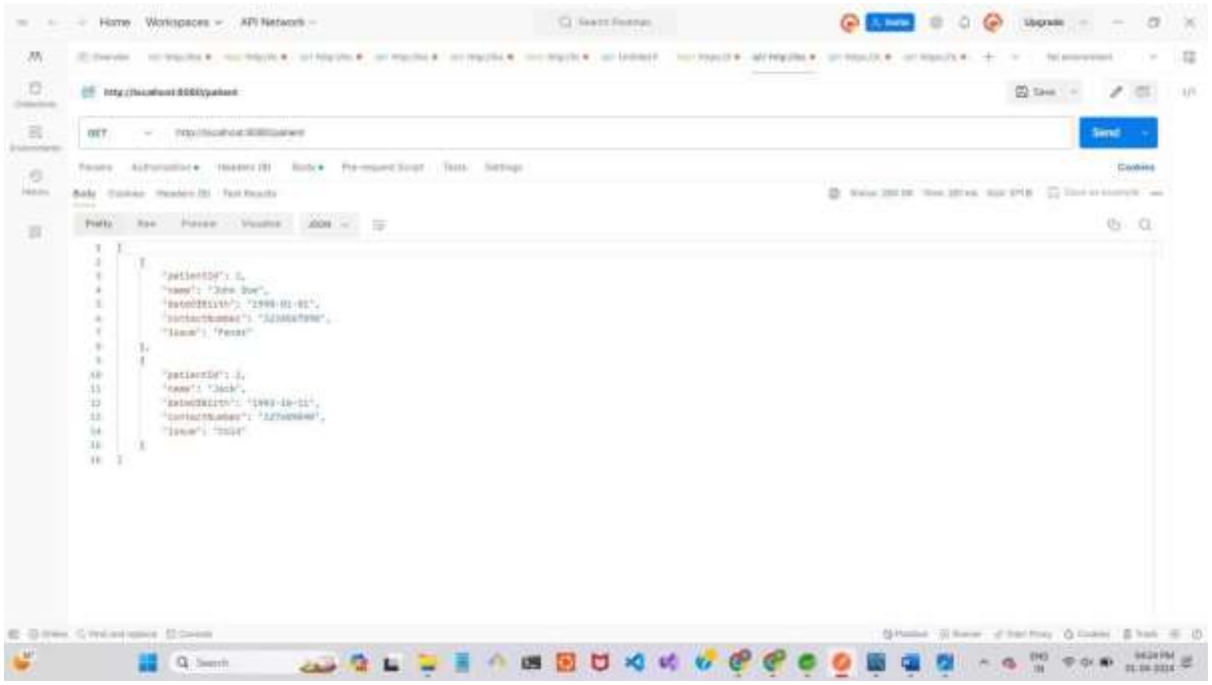
3) AppointmentController.java

```
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:

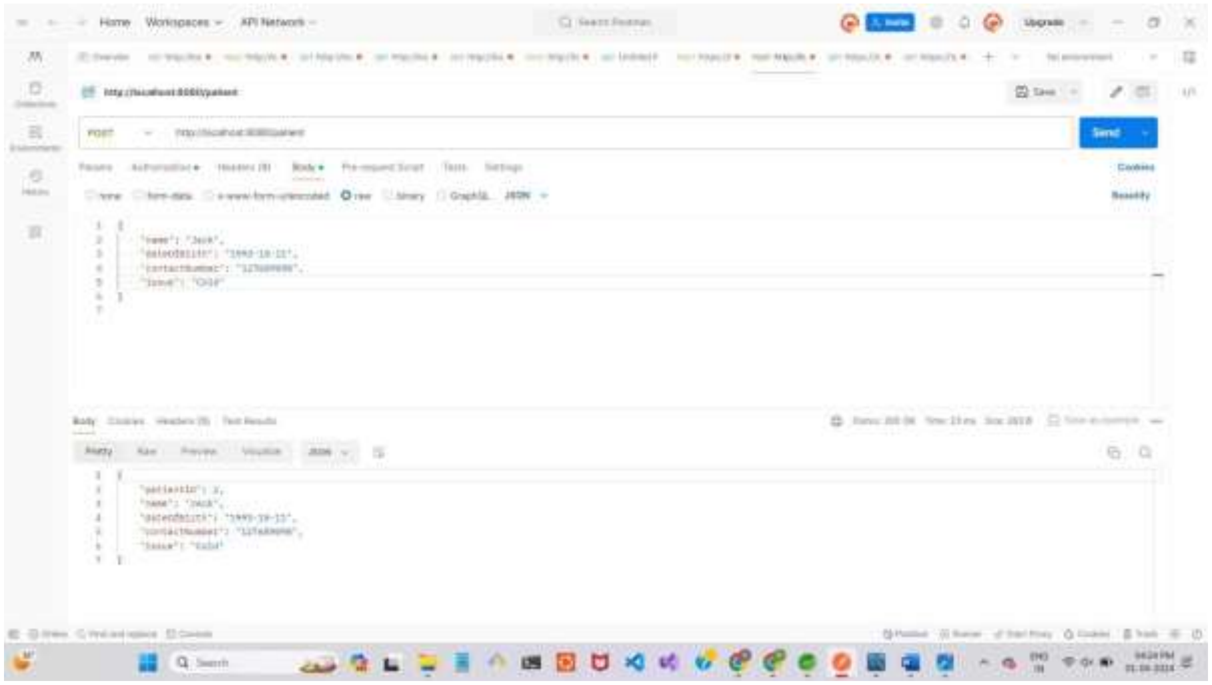


http://localhost:8080/patient

GET http://localhost:8080/patient

```
[
    {
        "patientId": 1,
        "name": "John Doe",
        "dateOfBirth": "1990-01-01",
        "contactNumber": "1234567890",
        "issue": "Fever"
    },
    {
        "patientId": 2,
        "name": "Jack",
        "dateOfBirth": "1993-10-11",
        "contactNumber": "[illegible]",
        "issue": "Cold"
    }
]
```

POST:



http://localhost:8080/patient

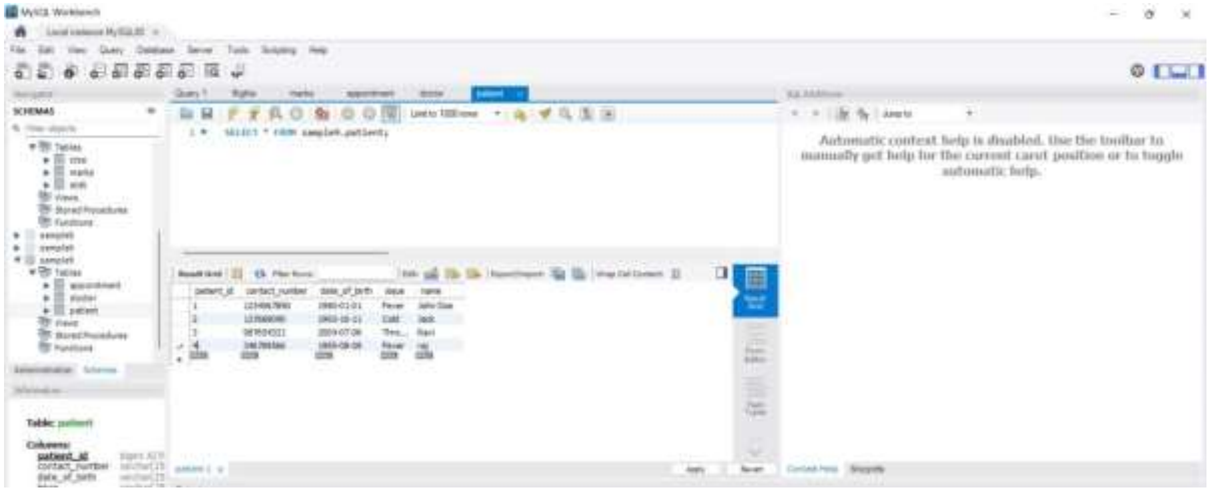
POST http://localhost:8080/patient

```
{
    "name": "Jack",
    "dateOfBirth": "1993-10-11",
    "contactNumber": "[illegible]",
    "issue": "Cold"
}
```

```
{
    "patientId": 2,
    "name": "Jack",
    "dateOfBirth": "1993-10-11",
    "contactNumber": "[illegible]",
    "issue": "Cold"
}
```

DATABASE:

PATIENT:

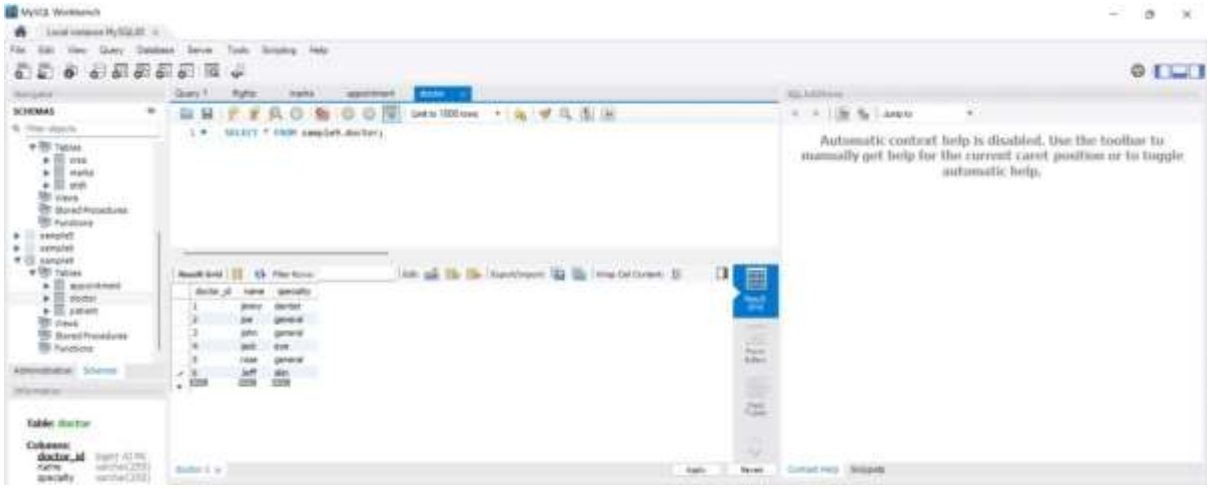


```
SELECT * FROM sample4.patient;
```

patient_id	contact_number	date_of_birth	issue	name
1	1234567890	1980-01-01	Fever	John Doe
2	123569090	1960-10-11	Cold	Jack
3	9876543211	2009-07-06	Thro...	Ravi
4	346789566	1969-08-08	Fever	raj
NULL	NULL	NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

DOCTOR

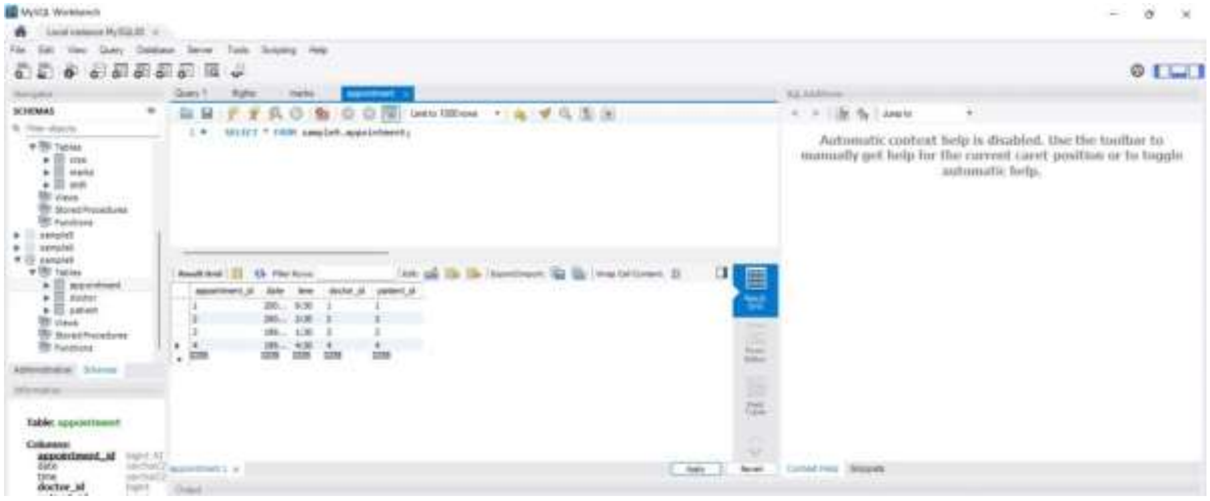


```
SELECT * FROM sample4.doctor;
```

doctor_id	name	specialty
1	jenny	dentist
2	joe	general
3	john	general
4	jack	eye
5	rose	general
6	Jeff	skin
NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

APPOINTMENT



```
SELECT * FROM sample6.appointment;
```

appointment_id	date	time	doctor_id	patient_id
1	200...	9:30	1	1
2	200...	2:30	2	2
3	199...	1:30	3	3
4	199...	4:30	4	4
NULL	NULL	NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Table: appointment

Columns:
appointment_id bigint AI
date varchar
time varchar
doctor_id bigint

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);
}
```

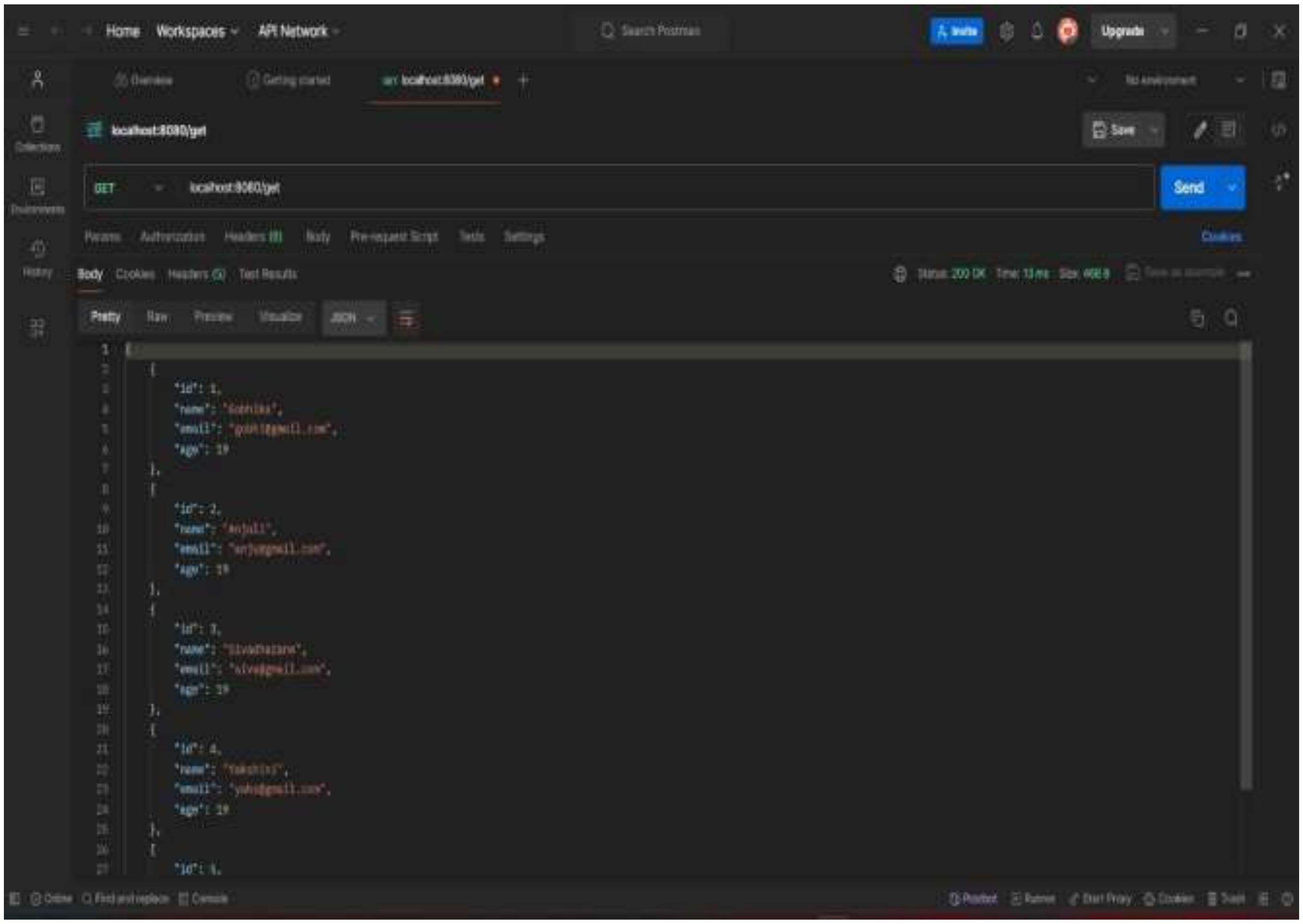
service:

```
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:

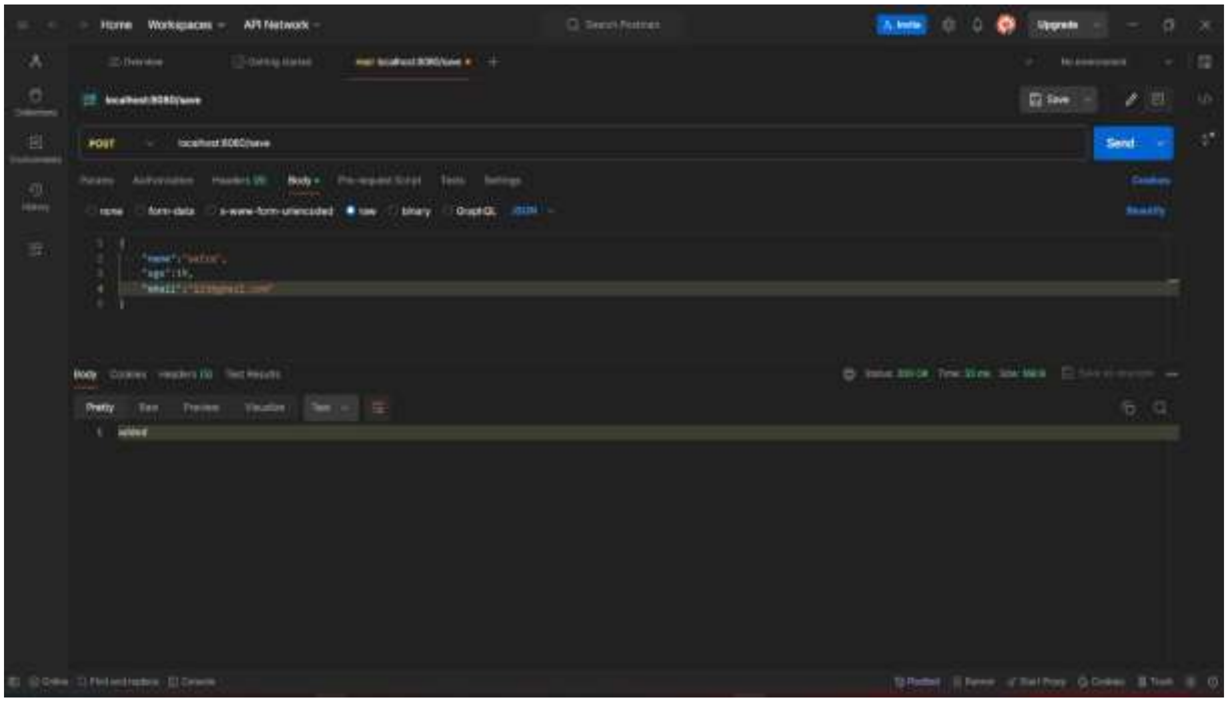


GET localhost:8080/get

Status: 200 OK Time: 13 ms

```
[
    {
        "id": 1,
        "name": "Gobhika",
        "email": "gobhi@gmail.com",
        "age": 19
    },
    {
        "id": 2,
        "name": "Anjuli",
        "email": "arju@gmail.com",
        "age": 19
    },
    {
        "id": 3,
        "name": "Sivadharani",
        "email": "siva@gmail.com",
        "age": 19
    },
    {
        "id": 4,
        "name": "Yokshini",
        "email": "yoki@gmail.com",
        "age": 19
    },
    {
        "id": 5,
```

Post:

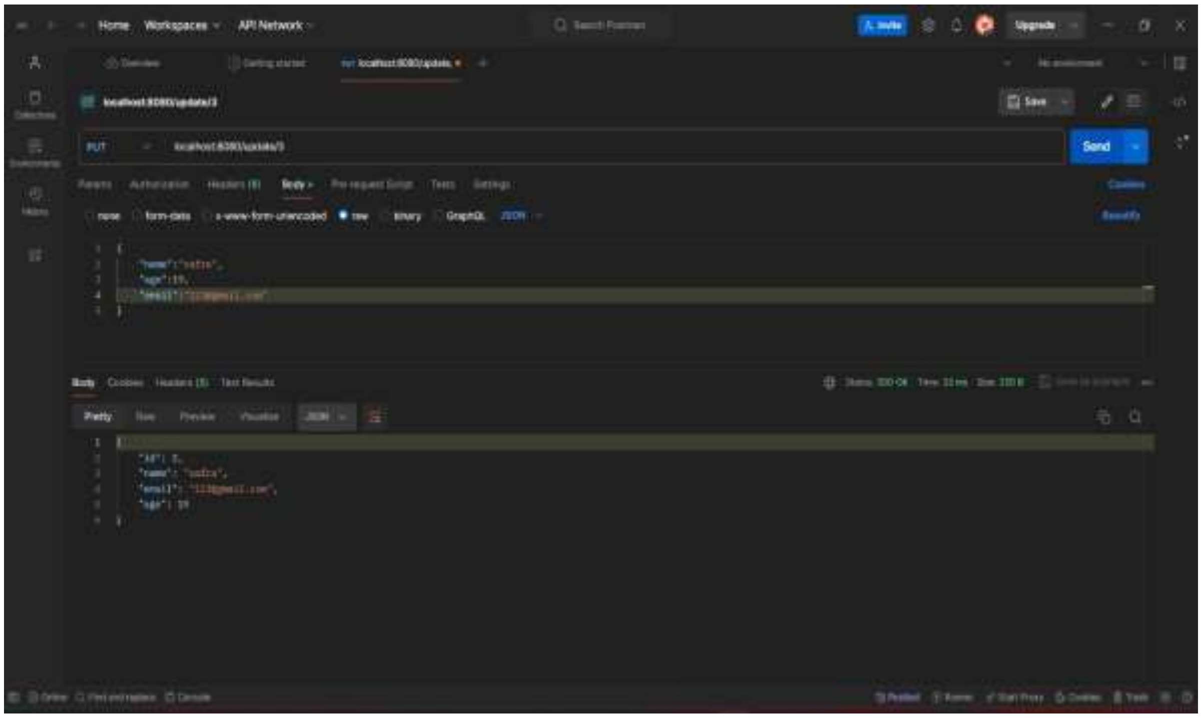


POST localhost:8080/save

```
{
    "name":"safira",
    "age":19,
    "email":"123@gmail.com"
}
```

```
added
```

Put:

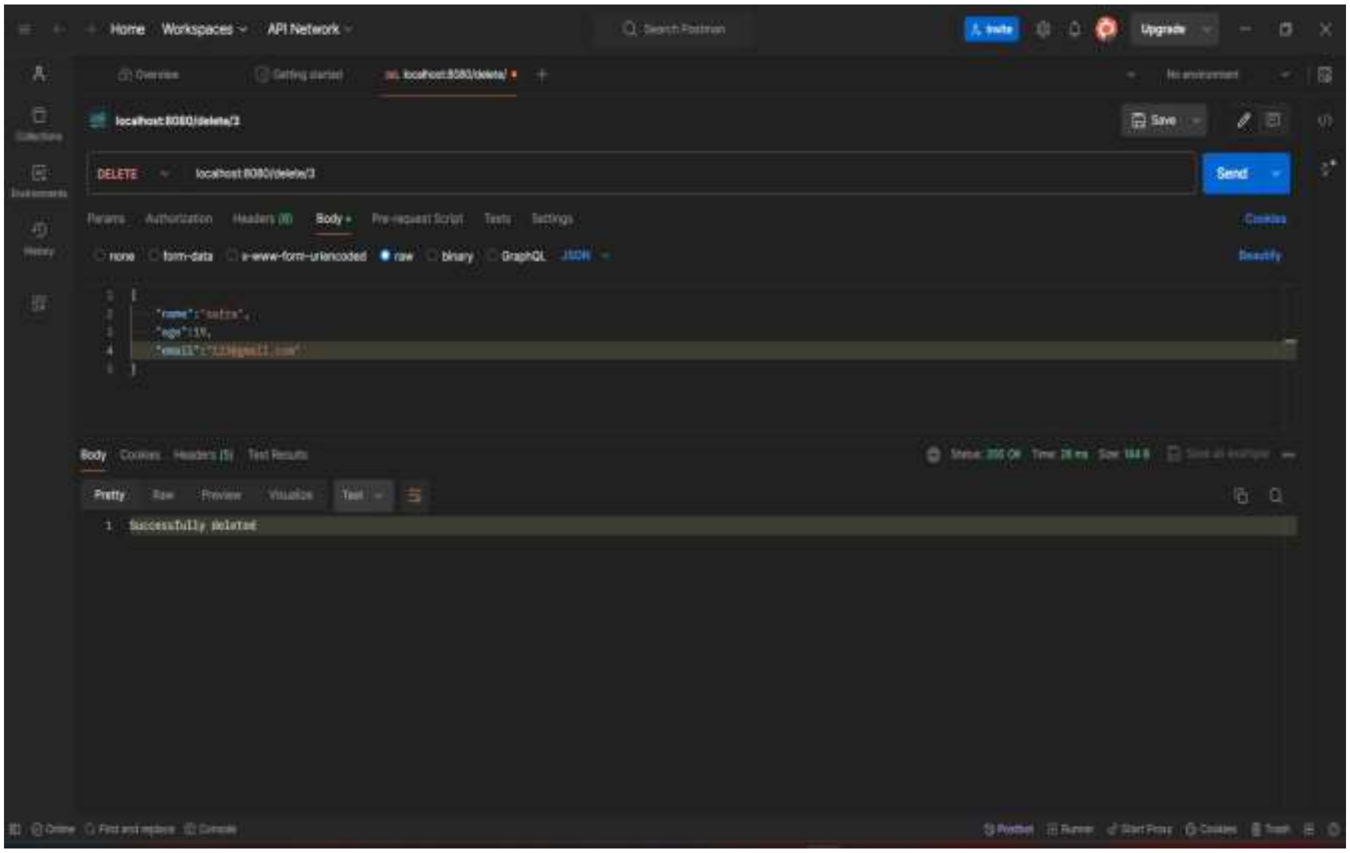


PUT localhost:8080/update/3

```
{
    "name":"safira",
    "age":19,
    "email":"123@gmail.com"
}
```

```
{
    "id": 3,
    "name": "safira",
    "email": "123@gmail.com",
    "age": 19
}
```

Delete:



DELETE localhost:8080/delete/3

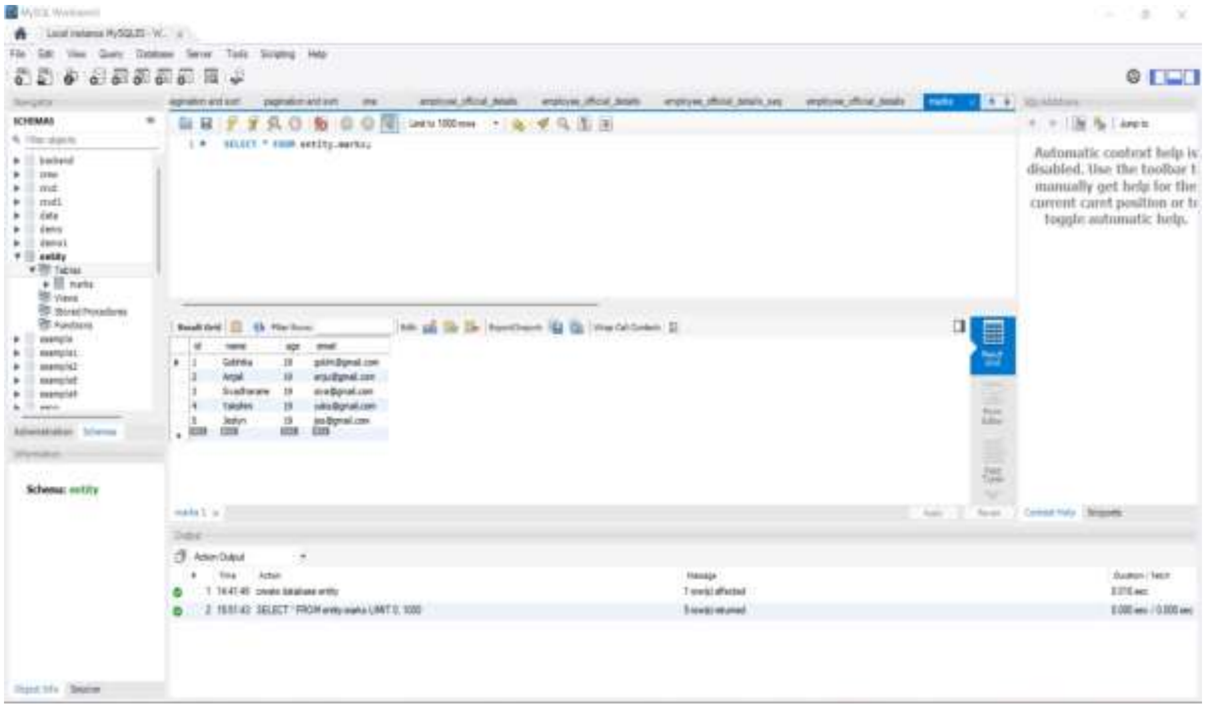
```
{
    "name":"satra",
    "age":19,
    "email":"123@gmail.com"
}
```

Response Body (Pretty, Text):

```
Successfully deleted
```

Status: 200 OK Time: 28 ms

Database:



```
SELECT * FROM entity.marks;
```

id	name	age	email
1	Gobhika	19	gobhi@gmail.com
2	Anjali	19	anju@gmail.com
3	Sivadharane	19	siva@gmail.com
4	Yakshini	19	yaks@gmail.com
5	Jeslyn	19	jes@gmail.com
NULL	NULL	NULL	NULL

Schema: entity

```
#  Time      Action                                    Message            Duration / Fetch
1  14:41:48  create database entity                    1 row(s) affected  0.016 sec
2  15:51:43  SELECT * FROM entity.marks LIMIT 0, 1000  5 row(s) returned  0.000 sec / 0.000 sec
```

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>{

}
```

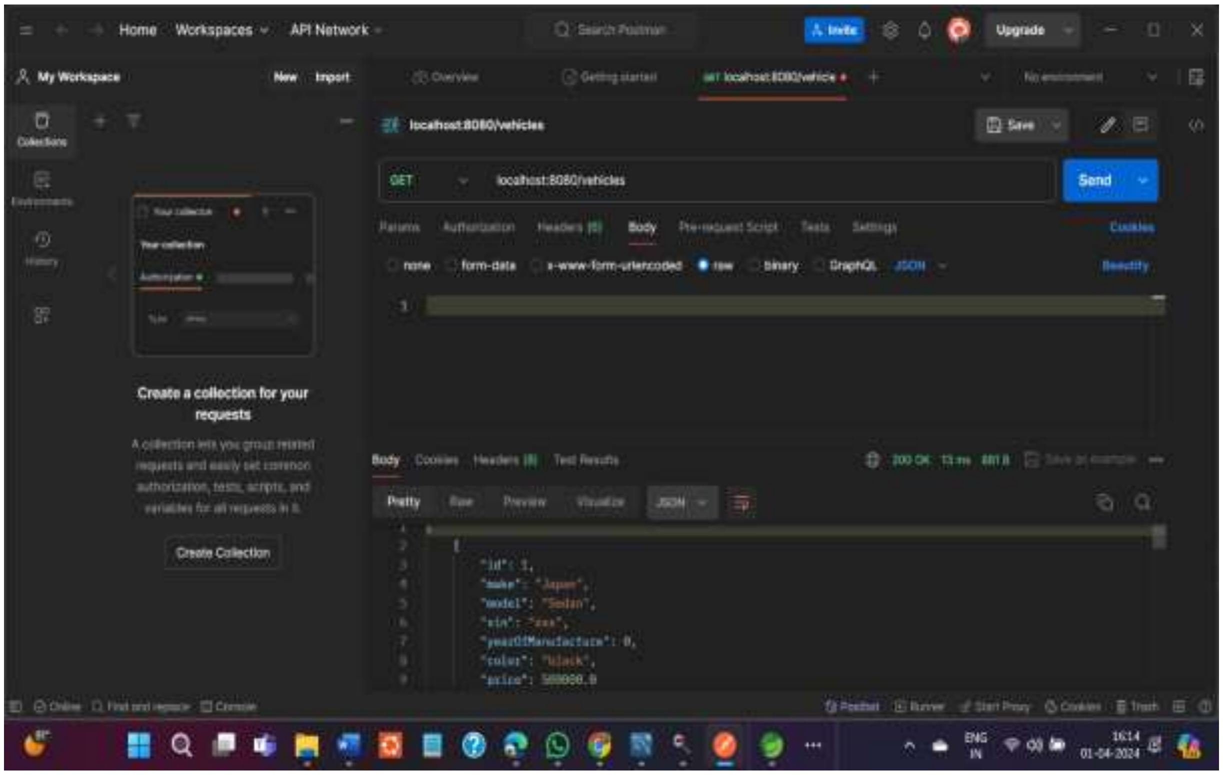
SERVICE

```
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:



localhost:8080/vehicles

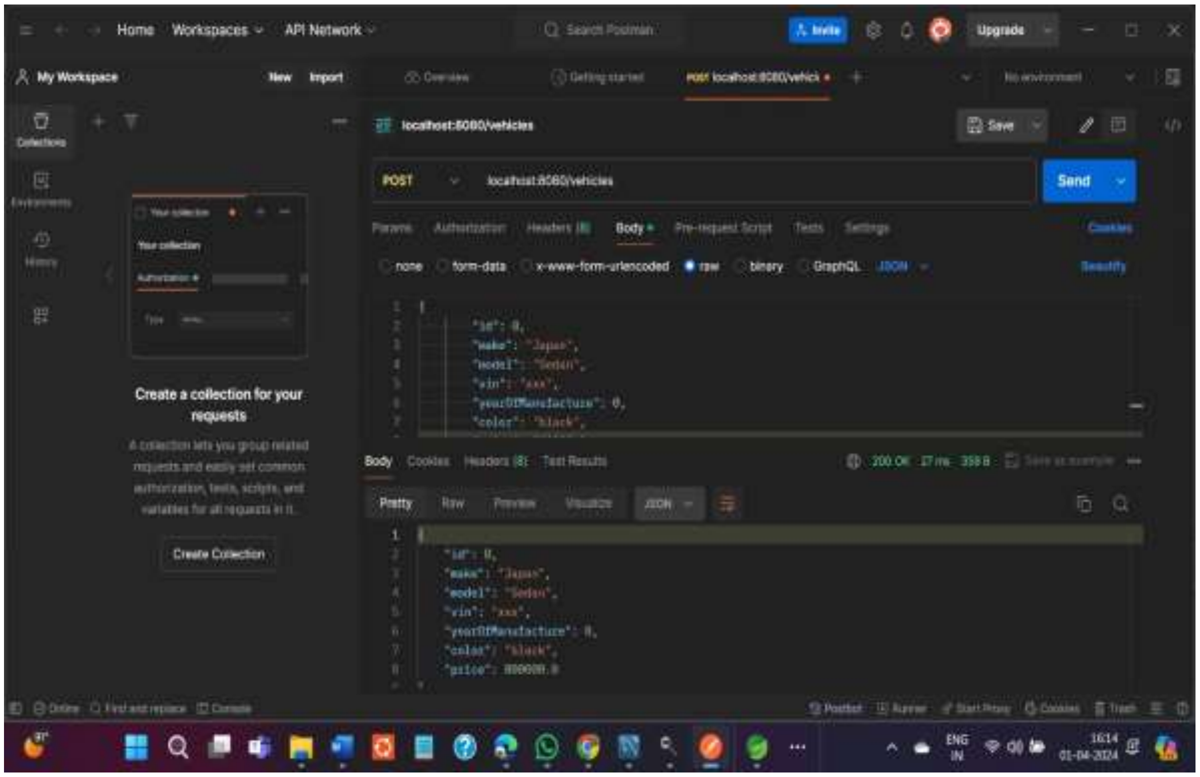
GET localhost:8080/vehicles

Body: raw JSON

200 OK 13 ms 481 B

```
[
    {
        "id": 1,
        "make": "Japan",
        "model": "Sedan",
        "vin": "xxx",
        "yearOfManufacture": 0,
        "color": "black",
        "price": 500000.0
```

POST:



localhost:8080/vehicles

POST localhost:8080/vehicles

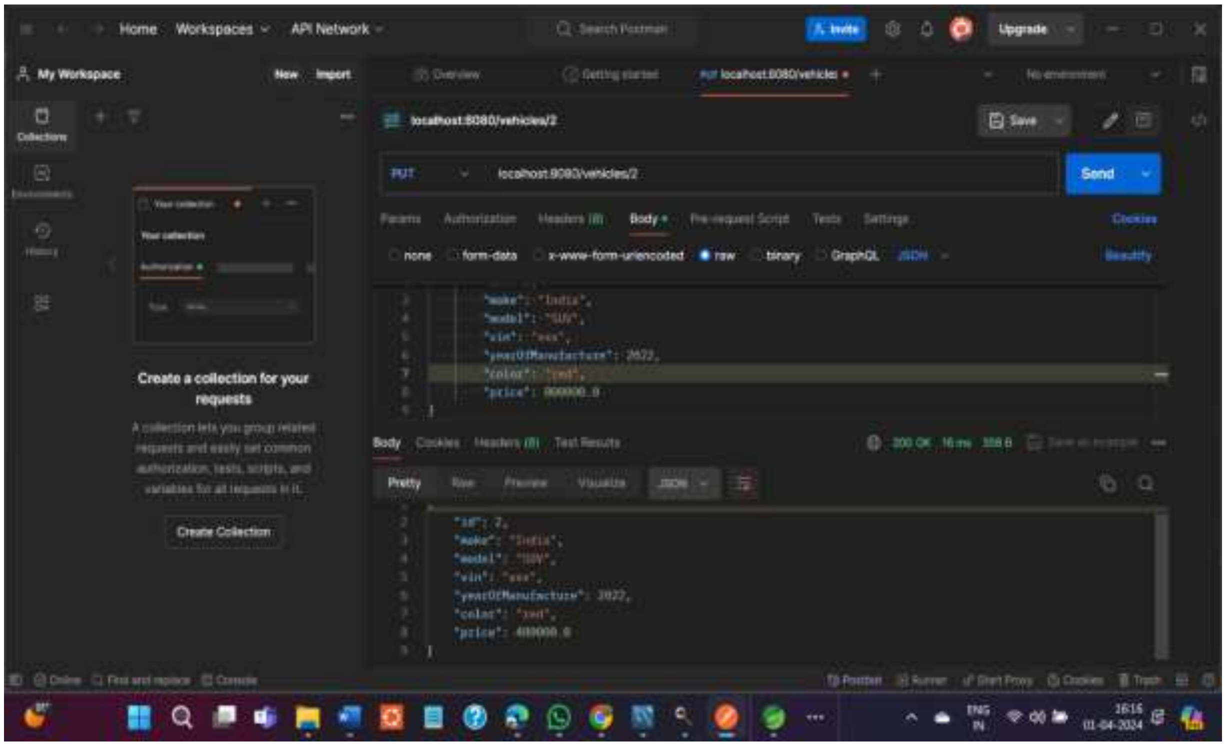
Body: raw JSON

```
{
    "id": 0,
    "make": "Japan",
    "model": "Sedan",
    "vin": "xxx",
    "yearOfManufacture": 0,
    "color": "black",
```

200 OK 17 ms 359 B

```
{
    "id": 0,
    "make": "Japan",
    "model": "Sedan",
    "vin": "xxx",
    "yearOfManufacture": 0,
    "color": "black",
    "price": 800000.0
}
```

PUT:



PUT localhost:8080/vehicles/2

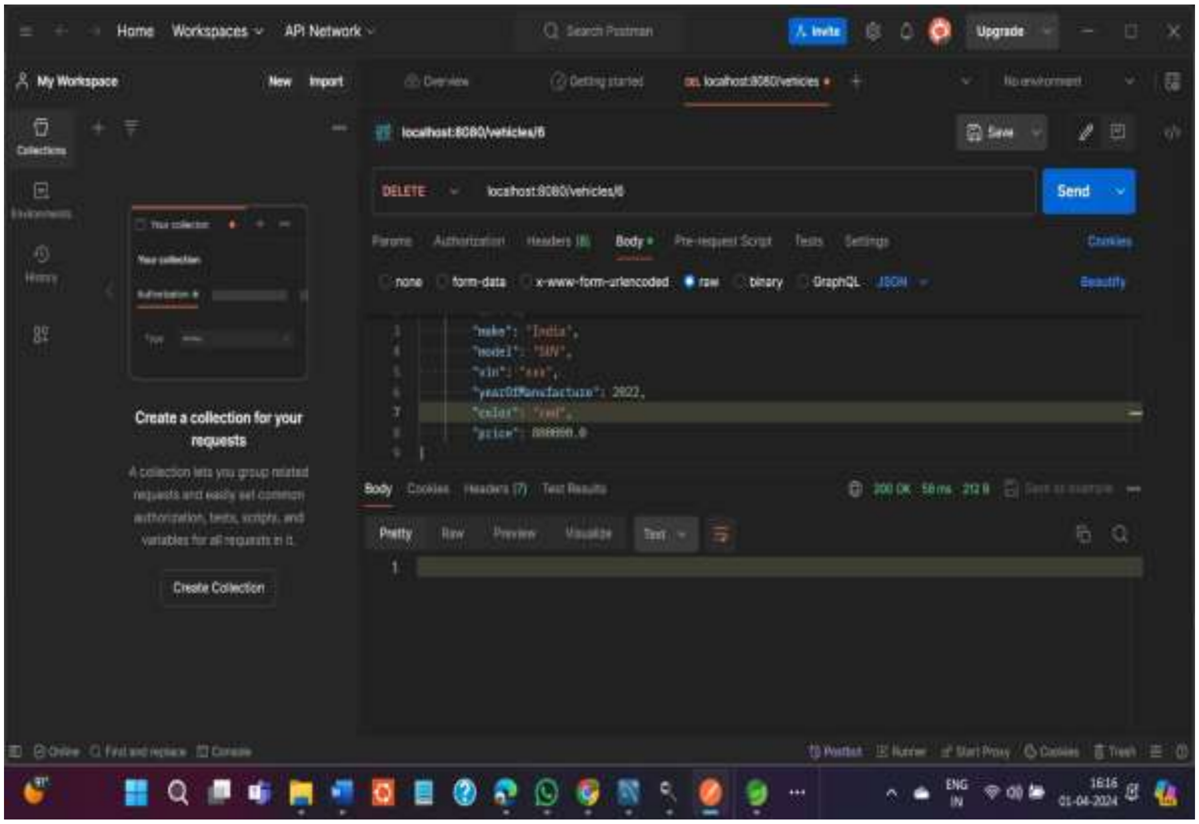
Body: raw JSON

```
    "make": "India",
    "model": "SUV",
    "vin": "xxx",
    "yearOfManufacture": 2022,
    "color": "red",
    "price": 800000.0
}
```

Body — 200 OK 16 ms 358 B

```
  "id": 2,
  "make": "India",
  "model": "SUV",
  "vin": "xxx",
  "yearOfManufacture": 2022,
  "color": "red",
  "price": 400000.0
}
```

DELETE:



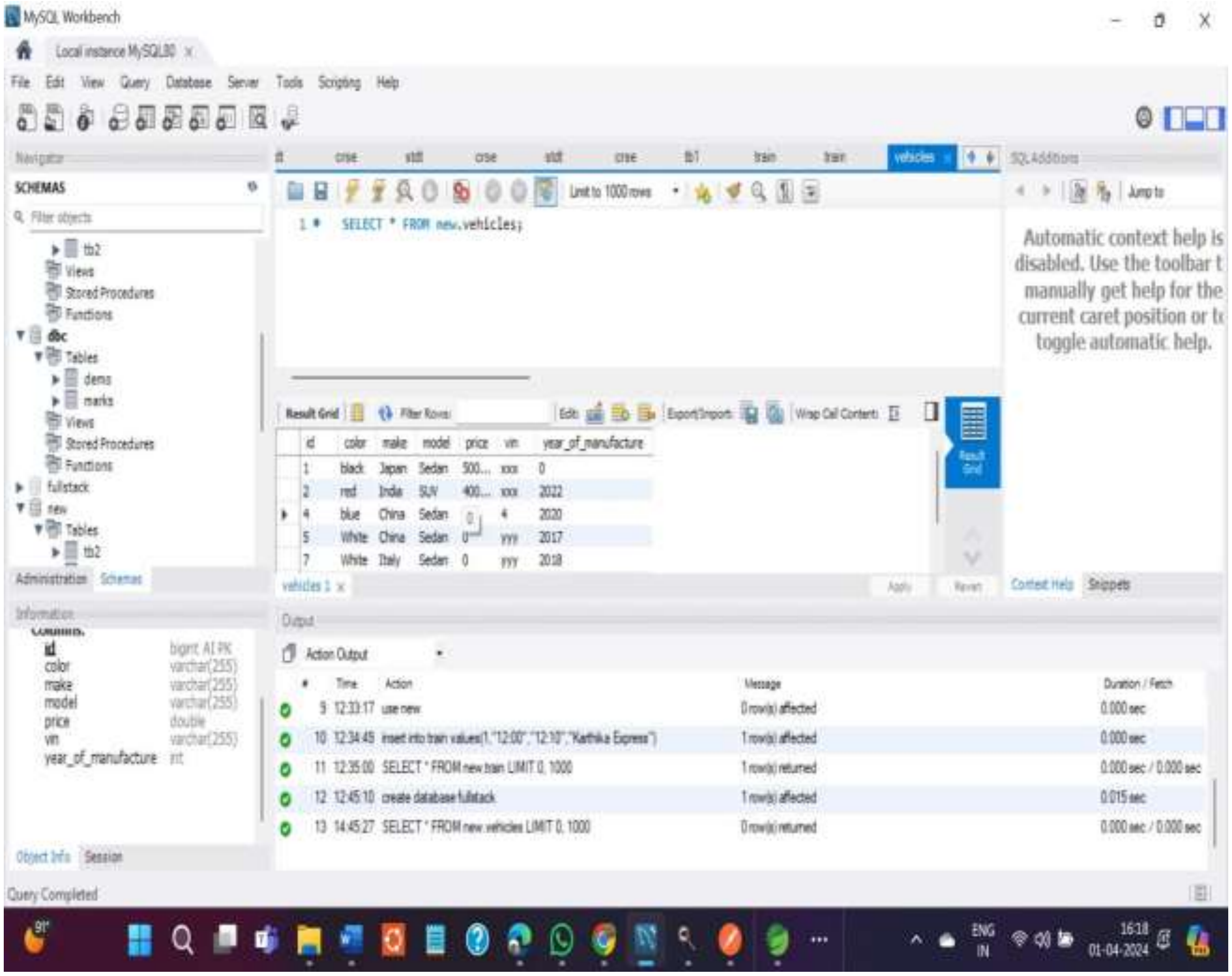
DELETE localhost:8080/vehicles/6

Body: raw JSON

```
    "make": "India",
    "model": "SUV",
    "vin": "xxx",
    "yearOfManufacture": 2022,
    "color": "red",
    "price": 888888.0
}
```

Body — 200 OK 58 ms 212 B

DATABASE:



```
SELECT * FROM new.vehicles;
```

id	color	make	model	price	vin	year_of_manufacture
1	black	Japan	Sedan	500...	xxx	0
2	red	India	SUV	400...	xxx	2022
4	blue	China	Sedan	0	4	2020
5	White	China	Sedan	0	yyy	2017
7	White	Italy	Sedan	0	yyy	2018

```
Columns:
id                  bigint AI PK
color               varchar(255)
make                varchar(255)
model               varchar(255)
price               double
vin                 varchar(255)
year_of_manufacture int
```

```
#   Time      Action                                                    Message             Duration / Fetch
9   12:33:17  use new                                                   0 row(s) affected   0.000 sec
10  12:34:45  insert into train values(1,"12:00","12:10","Karthika Express")  1 row(s) affected   0.000 sec
11  12:35:00  SELECT * FROM new.train LIMIT 0, 1000                     1 row(s) returned   0.000 sec / 0.000 sec
12  12:45:10  create database fullstack                                 1 row(s) affected   0.015 sec
13  14:45:27  SELECT * FROM new.vehicles LIMIT 0, 1000                  0 row(s) returned   0.000 sec / 0.000 sec
```

Query Completed

Result:

Thus, the application is implemented in Spring boot.