INDEX

**Subject** : Current Computing trends in Java

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.no** | **Practical Name** | **Date** | **Sign** | **Remark** |
| 1 | Bootstrapping simple spring-boot Project. |  |  |  |
| 2 | Practical for Restful spring boot application. |  |  |  |
| 3 | Practical to create student entity using CRUD Operation with exception handling create spring boot (use hibernate MYSQL for backend) |  |  |  |
| 4 | Develop an employee entity where the user can create employee data. Use Hibernate to map the address entity and its relationship(one-to-one) with employee entity |  |  |  |
| 5 | Develop an employee entity where the user can create employee data. Use Hibernate to map the address entity and its relationship (one-to- many) with employee entity |  |  |  |
| 6 | Develop the micro service for employee management implement endpoint for retrieve employee details. Utilize spring boot and spring JPA to store and retrieve employee data. |  |  |  |
| 7 | Develop a library management system where users can borrow and return books. Use Hibernate to map the "Book" and "User" entities and their relationship as a many-to- many association. Implement CRUD operations to manage books and user records, as well as handling book borrowing and returning  operations. |  |  |  |
| 8 | Build a blog management system where users can create, read, update, and delete blog posts. Use Hibernate to map the "Blog" entity and its relationships with other entities such as "User" and "Comment." Implement CRUD operations to manage blog posts. |  |  |  |

# /\*\* Practical 1: Bootstrapping simple spring-boot Project.\*\*/

**//Class SimplespringbootApplication**

Package com.example.simplespringboot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SimplespringbootApplication {

public static void main(String[] args) { SpringApplication.run(SimplespringbootApplication.class, args);

}

}

**//HelloController**

package com.example.simplespringboot;

import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloController {

@RequestMapping("/getString") private String getString()

{

return "This is the first Spring Boot Practical";

}

}

**//pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="[http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation=["http://maven.apa](http://maven.apache.org/POM/4.0.0)c[he.org/POM/4.0.0](http://maven.apache.org/POM/4.0.0) https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>simplespringboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>simplespringboot</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

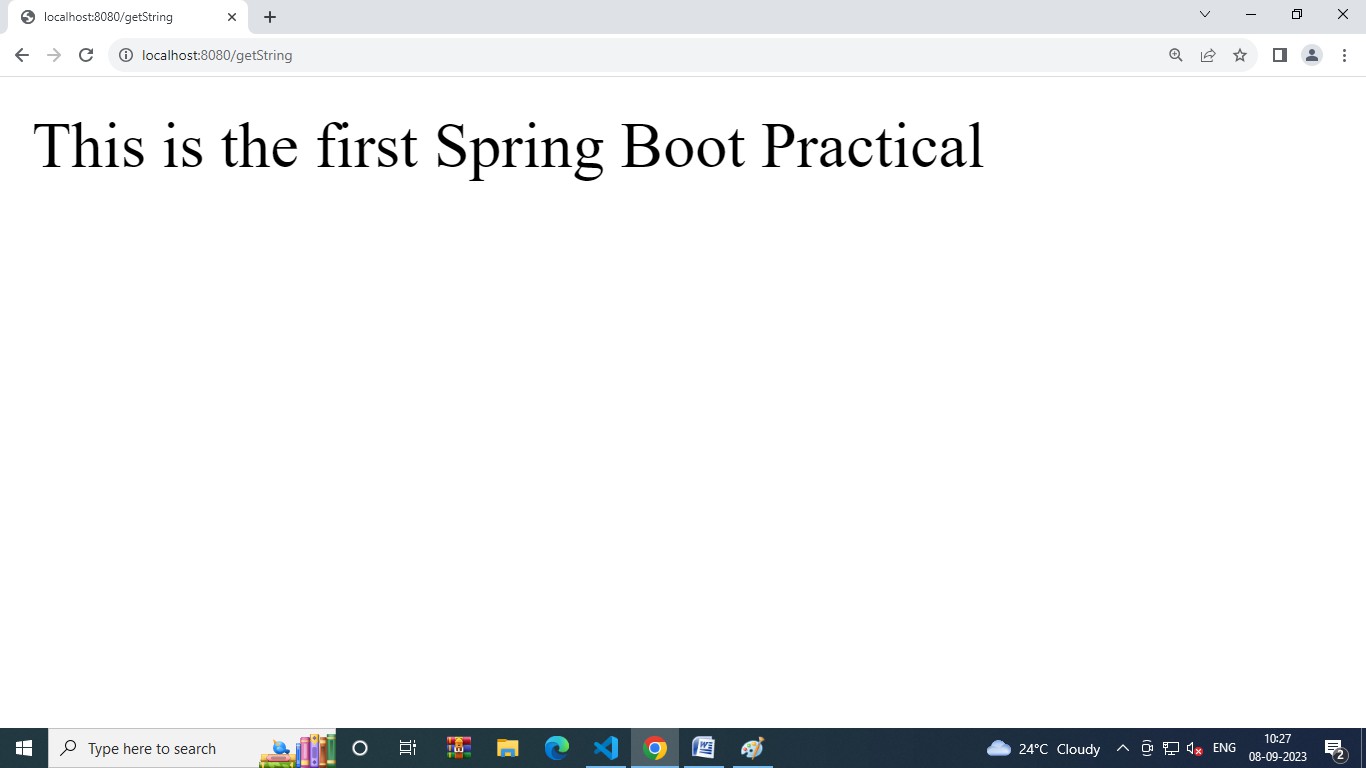
<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

# /\*\* Practical 2: Practical for Restful spring boot application. \*\*/

**//Practical No 2: Practical For RESTFul Spring Boot Application To Create CRUD operations for department entity.**

**//entity**

package com.example.student.entity;

// Importing required classes

import lombok.AllArgsConstructor; import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor; import jakarta.persistence.\*;

// Annotations @Entity @Data

@NoArgsConstructor @AllArgsConstructor @Builder

// Class

public class Department { @jakarta.persistence.Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Long departmentId;

private String departmentName; private String departmentAddress;

private String departmentCode;

}

**//Controller**

package com.example.student.controller;

// Java Program to Illustrate DepartmentController File

// Importing package module to this code import com.example.student.entity.Department;

import com.example.student.service.DepartmentService; import java.util.List;

// Importing required classes

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

// Annotation @RestController

// Class

public class DepartmentController {

// Annotation

@Autowired private DepartmentService departmentService;

// Save operation @PostMapping("/departments") public Department saveDepartment(

@Validated @RequestBody Department department)

{

return departmentService.saveDepartment(department);

}

// Read operation @GetMapping("/departments")

public List<Department> fetchDepartmentList()

{

return departmentService.fetchDepartmentList();

}

// Update operation @PutMapping("/departments/{id}") public Department

updateDepartment(@RequestBody Department department, @PathVariable("id") Long departmentId)

{

return departmentService.updateDepartment( department, departmentId);

}

// Delete operation @DeleteMapping("/departments/{id}")

public String deleteDepartmentById(@PathVariable("id") Long departmentId)

{

departmentService.deleteDepartmentById( departmentId);

return "Deleted Successfully";

}

}

**//service**

package com.example.student.service;

// Java Program to Illustrate DepartmentService File

// Importing package to this code fragment

import com.example.student.entity.Department;

// Importing required classes import java.util.List;

// Interface

public interface DepartmentService {

// Save operation

Department saveDepartment(Department department);

// Read operation

List<Department> fetchDepartmentList();

// Update operation

Department updateDepartment(Department department, Long departmentId);

// Delete operation

void deleteDepartmentById(Long departmentId);

}

**//DEPARTMENTSERVICEIMPL**

package com.example.student.service;

// Java Program to Illustrate DepartmentServiceImpl File

// Importing package module to this code

import com.example.student.entity.Department;

import com.example.student.repository.DepartmentRepository;// Importing required classes import java.util.List;

import java.util.Objects;

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

// Annotation @Service

// Class

public class DepartmentServiceImpl implements DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

// Save operation @Override

public Department saveDepartment(Department department)

{

return departmentRepository.save(department);

}

// Read operation

@Override public List<Department> fetchDepartmentList()

{

return (List<Department>) departmentRepository.findAll();

}

// Update operation @Override

public Department updateDepartment(Department department,

Long departmentId)

{

Department depDB

= departmentRepository.findById(departmentId)

.get();

if (Objects.nonNull(department.getDepartmentName()) && !"".equalsIgnoreCase(

department.getDepartmentName())) { depDB.setDepartmentName(

department.getDepartmentName());

}

if (Objects.nonNull( department.getDepartmentAddress())

&& !"".equalsIgnoreCase( department.getDepartmentAddress())) {

depDB.setDepartmentAddress( department.getDepartmentAddress());

}

if (Objects.nonNull(department.getDepartmentCode()) && !"".equalsIgnoreCase(

department.getDepartmentCode())) { depDB.setDepartmentCode(

department.getDepartmentCode());

}

return departmentRepository.save(depDB);

}

// Delete operation @Override

public void deleteDepartmentById(Long departmentId)

{

departmentRepository.deleteById(departmentId);

}

}

**//repository**

package com.example.student.repository;

// Java Program to Illustrate DepartmentRepository File

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

import com.example.student.entity.Department;

// Annotation @Repository

// Interface

public interface DepartmentRepository

extends JpaRepository<Department, Long> {

}

//DepartmentApplication package com.example.student;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication

public class DepartmentApplication { public static void main(String[] args) {

SpringApplication.run(DepartmentApplication.class, args);

}

}

**//deparment application**

package com.example.student;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication

public class DepartmentApplication { public static void main(String[] args) {

SpringApplication.run(DepartmentApplication.class, args);

}

}

**//Appliocation.properties**

server.port=8082

# Configuration for MySQL Database spring.jpa.hibernate.ddl-auto=update spring.datasource.url=jdbc:mysql://localhost:3306/mydb spring.datasource.username=root spring.datasource.password=msc@1234

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.jpa.show-sql:true

//pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="[http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation=["http://maven.apa](http://maven.apache.org/POM/4.0.0)c[he.org/POM/4.0.0](http://maven.apache.org/POM/4.0.0) https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.4</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>student</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>student</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

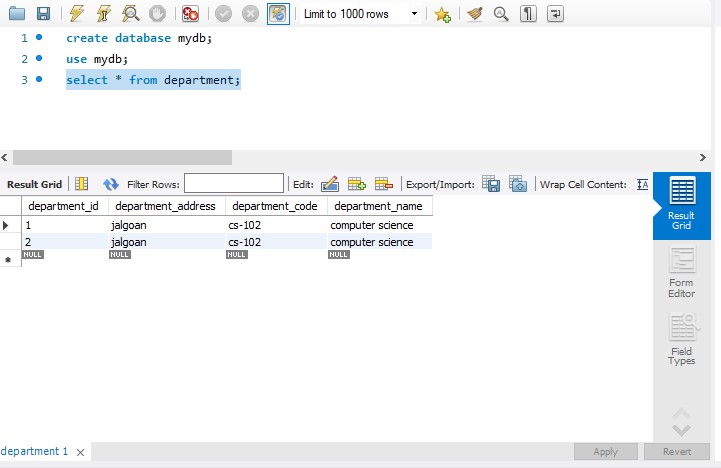
</plugins>

</build>

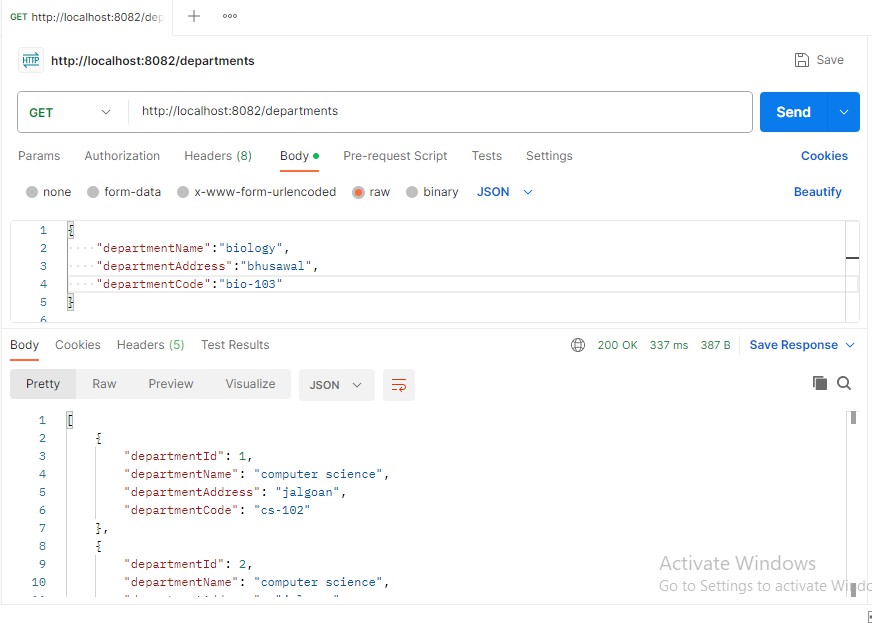
</project>

**Output:**

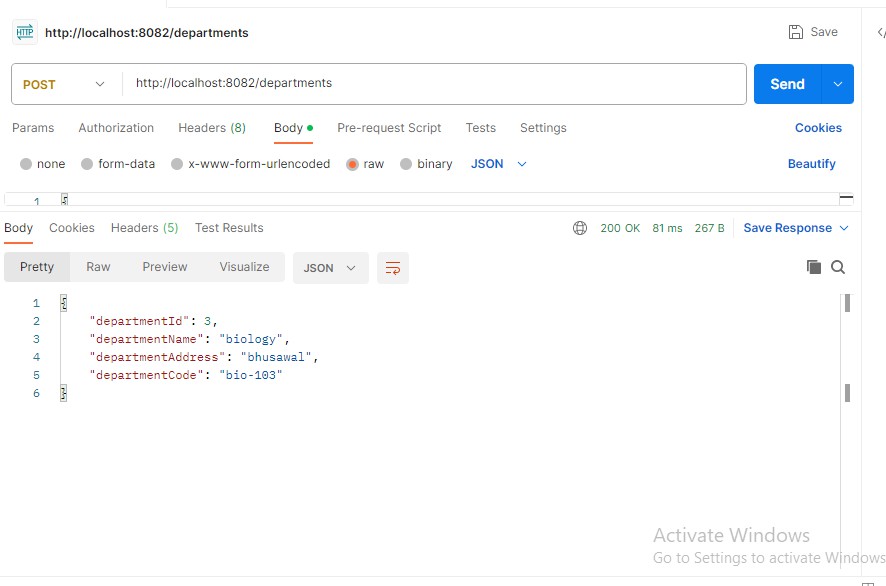
**MYSQL WORKBEnCH**



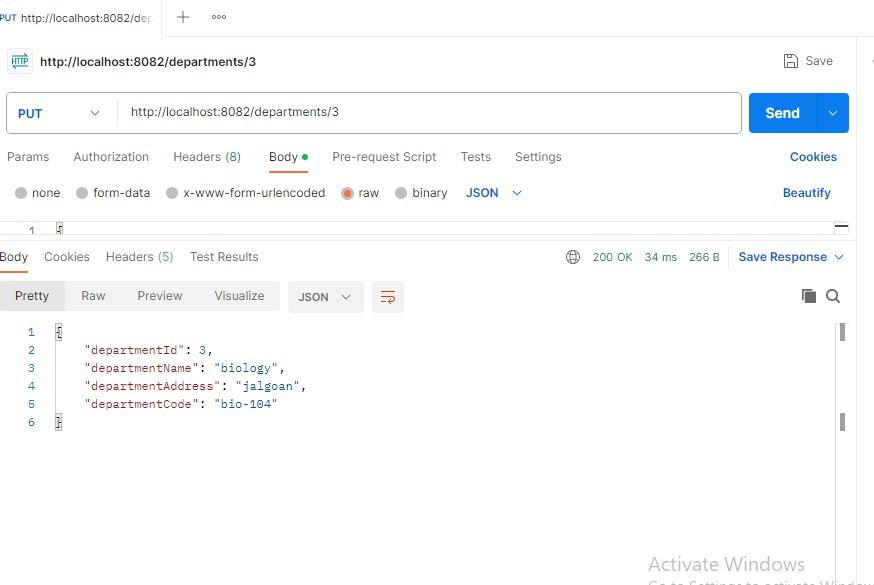
**CREATE operation**



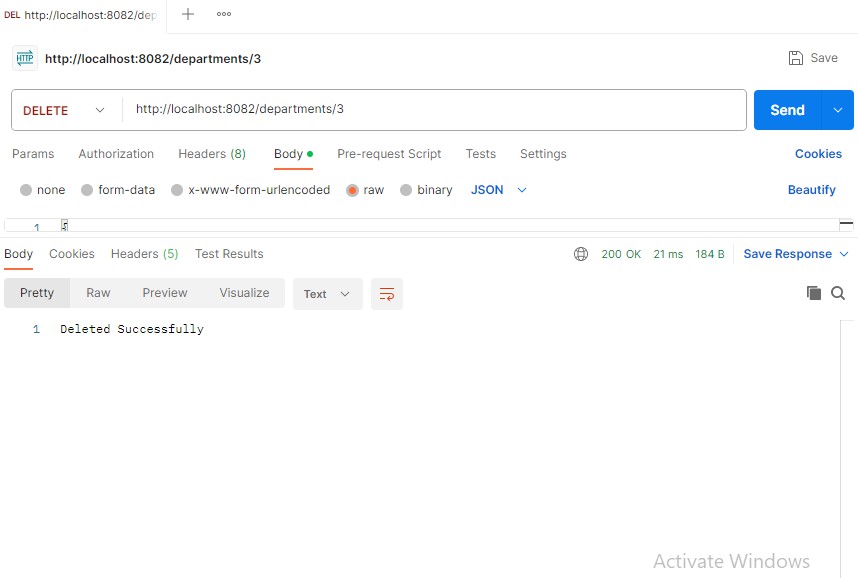
**READ operation**



**UPDATE operation**



**DELETE operation**



# /\*\* Practical 3: Practical to create student entity using CRUD Operation with exception handling create spring boot (use hibernate mysql for backend) \*\*/

**//Student.java**

package com.example.studentprojects.entity; import jakarta.persistence.Entity;

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

//import jakarta.persistence.Table;

@Entity

//@Table (name = "Student") public class Student {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

// @Column (name = "StudID") private Long id;

public Long getId() { return id;

}

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

}

// @Column (name = "FirstName") private String firstName;

public String getFirstName() { return firstName;

}

public void setFirstName(String firstName) { this.firstName = firstName;

}

// @Column (name = "LastName") private String lastName;

public String getLastName() { return lastName;

}

public void setLastName(String lastName) { this.lastName = lastName;

}

// @Column (name = "Email") private String email;

public String getEmail() { return email;

}

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

}

// Getters and setters

}

**// StudentRepository.java**

package com.example.studentprojects.repository;

import com.example.studentprojects.entity.Student;

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

public interface StudentRepository extends JpaRepository<Student, Long> {

}

**//StudentService.java**

package com.example.studentprojects.service;

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

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

import com.example.studentprojects.repository.StudentRepository; import com.example.studentprojects.entity.Student;

import com.example.studentprojects.exception.StudentNotFoundException;

@Service

public class StudentService {

private final StudentRepository studentRepository;

@Autowired

public StudentService(StudentRepository studentRepository) { this.studentRepository = studentRepository;

}

public List<Student> getAllStudents() { return studentRepository.findAll();

}

public Student getStudentById(Long id) {

Optional<Student> studentOptional = studentRepository.findById(id); if (studentOptional.isPresent()) {

return studentOptional.orElse(null);

}

else

{

throw new StudentNotFoundException("Student with ID " + id + " not found");

}

}

public void saveStudent(Student student) { studentRepository.save(student);

}

public void deleteStudent(Long id) {

Optional<Student> studentOptional = studentRepository.findById(id); if (studentOptional.isPresent()) {

studentRepository.deleteById(id);

} else {

throw new StudentNotFoundException("Student with ID " + id + " not found");

}

}

}

**//StudentNotFoundException.java**

package com.example.studentprojects.exception;

public class StudentNotFoundException extends RuntimeException {

public StudentNotFoundException(String message) { super(message);

}

}

**// StudentController.java**

package com.example.studentprojects.controller;

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

import org.springframework.web.bind.annotation.\*;

import com.example.studentprojects.service.StudentService; import com.example.studentprojects.entity.Student;

import com.example.studentprojects.exception.StudentNotFoundException; import java.util.List;

@RestController @RequestMapping("/api/students") public class StudentController {

private final StudentService studentService; @Autowired

public StudentController(StudentService studentService) { this.studentService = studentService;

}

@GetMapping

public List<Student> getAllStudents() { return studentService.getAllStudents();

}

@GetMapping("/{id}")

public Student getStudentById(@PathVariable Long id) { return studentService.getStudentById(id);

}

@PostMapping

public void saveStudent(@RequestBody Student student) { studentService.saveStudent(student);

}

@PutMapping("/{id}")

public void updateStudent(@PathVariable Long id, @RequestBody Student student) { student.setId(id);

studentService.saveStudent(student);

}

@DeleteMapping("/{id}")

public void deleteStudent(@PathVariable Long id) { studentService.deleteStudent(id);

}

@ExceptionHandler(StudentNotFoundException.class) @ResponseStatus(HttpStatus.NOT\_FOUND)

public String handleStudentNotFound(StudentNotFoundException ex) { return ex.getMessage();

}

}

**// application.properties**

server.port=8082

# Configuration for MySQL Database spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/studdb?createDatabaseIfNotExist=true spring.datasource.username=root

spring.datasource.password=msc@1234 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.jpa.show-sql:true

**// pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="[http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.4</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>studentprojects</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>studentprojects</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<excludes>

<exclude>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</exclude>

</excludes>

</configuration>

</plugin>

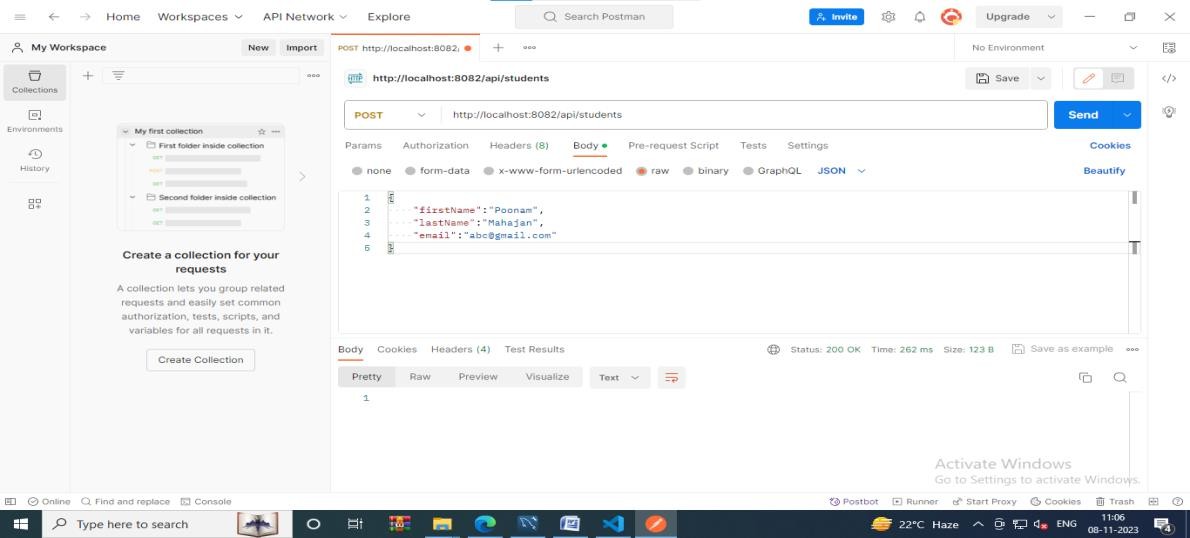
</plugins>

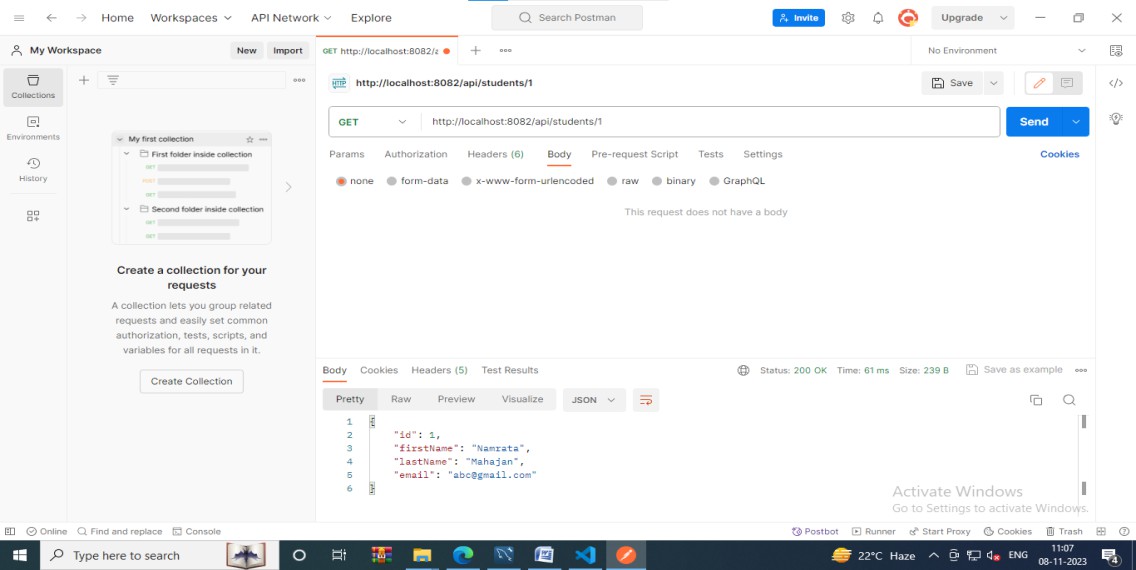
</build>

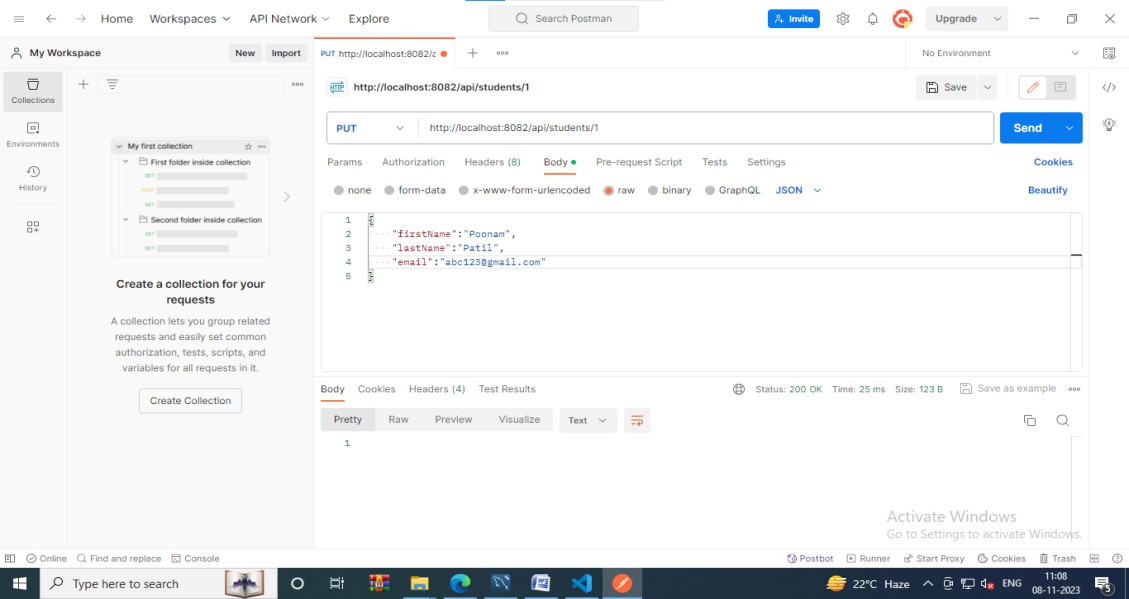
</project>

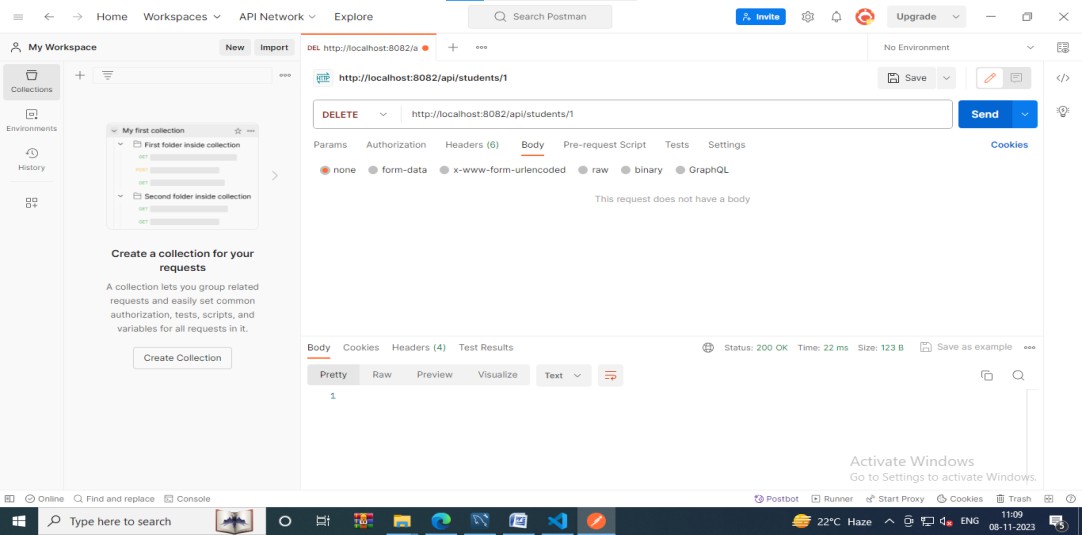
**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**// Create Operation**

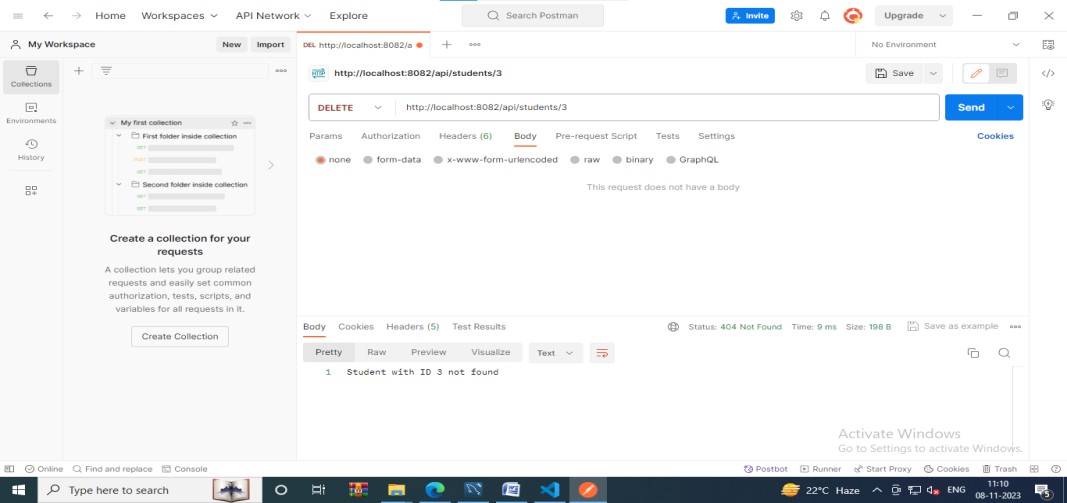


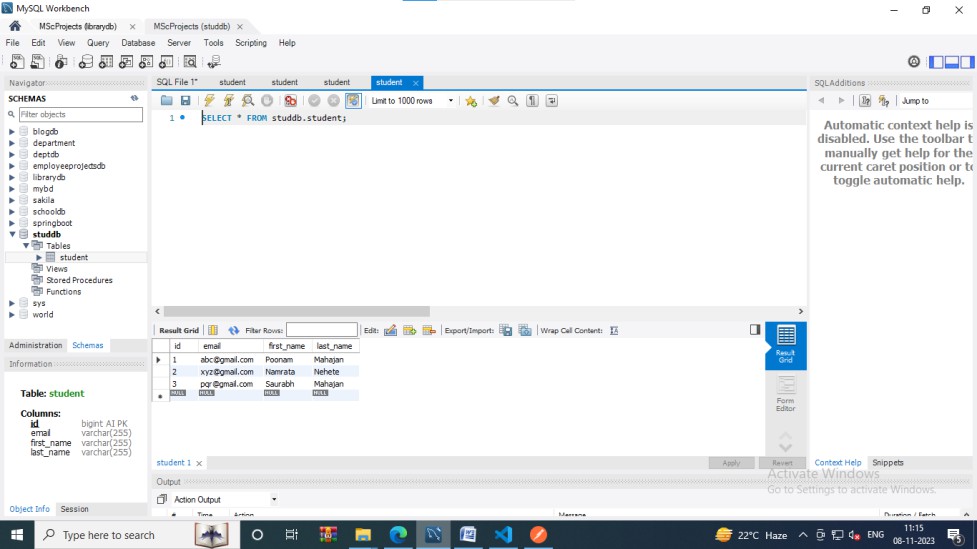
**//Retrieve Operation**

**//Update Operation**

**//Delete Operation**

**// Exception**



**// Database auto-updating at MYSQL Workbench**

# /\*\* Practical 4: Develop an employee entity where the user can create employee data. Use Hibernate to map the address entity and its relationship (one-to-one) with employee entity\*\*/

**//Employee.java**

package com.example.employeeprojects.entity;

import jakarta.persistence.CascadeType; import jakarta.persistence.Column; import jakarta.persistence.Entity;

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

import jakarta.persistence.JoinColumn; import jakarta.persistence.OneToMany; import jakarta.persistence.OneToOne; import jakarta.persistence.Table;

import jakarta.transaction.Transactional; import lombok.Data;

import lombok.NoArgsConstructor; import java.util.List;

@Entity @Transactional @Data @NoArgsConstructor

@Table(name = "employee\_details") public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.AUTO) @Column(name = "emp\_id")

private Long empId; private String empName; private Integer empAge;

@OneToOne(cascade = CascadeType.ALL) @JoinColumn(name = "fk\_add\_id")

// @OneToMany(cascade = CascadeType.ALL)

// @JoinColumn(name = "fk\_emp\_id", referencedColumnName = "emp\_id")

private Address address;

// private List<Address> address;

}

**//Address.java**

package com.example.employeeprojects.entity; import jakarta.persistence.Column;

import jakarta.persistence.Entity;

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

import jakarta.persistence.OneToOne; import jakarta.persistence.Table;

import jakarta.transaction.Transactional; import lombok.Data;

import lombok.NoArgsConstructor;

@Entity @Transactional @Data @NoArgsConstructor

@Table(name = "address") public class Address {

@Id

@GeneratedValue(strategy = GenerationType.AUTO) @Column(name = "add\_id")

private Long addressid; private String city;

private String addressType; @OneToOne(mappedBy = "address") private Employee employee;

}

**//EmployeeRepository.java**

package com.example.employeeprojects.repository;

import org.springframework.data.jpa.repository.JpaRepository; import com.example.employeeprojects.entity.Employee;

public interface EmployeeRepository extends JpaRepository<Employee,Long>{

}

**//AddressRepository.java**

package com.example.employeeprojects.repository;

import org. springframework.data.jpa.repository.JpaRepository; import com.example.employeeprojects.entity.Address;

public interface AddressRepository extends JpaRepository<Address,Long>{

}

**//EmployeeController.java**

package com.example.employeeprojects.controller;

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

import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestBody; import org.springframework.web.bind.annotation.RestController; import java.util.List;

import com.example.employeeprojects.entity.Employee;

import com.example.employeeprojects.repository.EmployeeRepository;

@RestController

public class EmployeeController { @Autowired

private EmployeeRepository employeeRepository;

@PostMapping("/saveEmployees")

public ResponseEntity<String> saveEmployees(@RequestBody List<Employee> empData){ employeeRepository.saveAll(empData);

return ResponseEntity.ok("Data Saved");

}

}

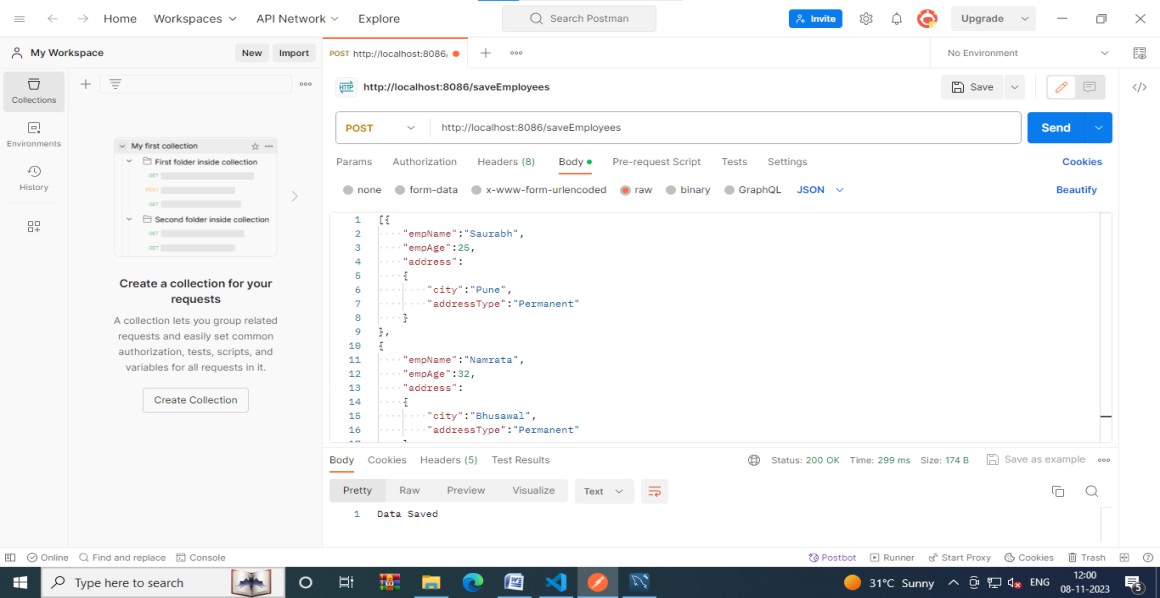
**//application.properties**

server.port=8086

# Configuration for MySQL Database spring.jpa.hibernate.ddl-auto=update

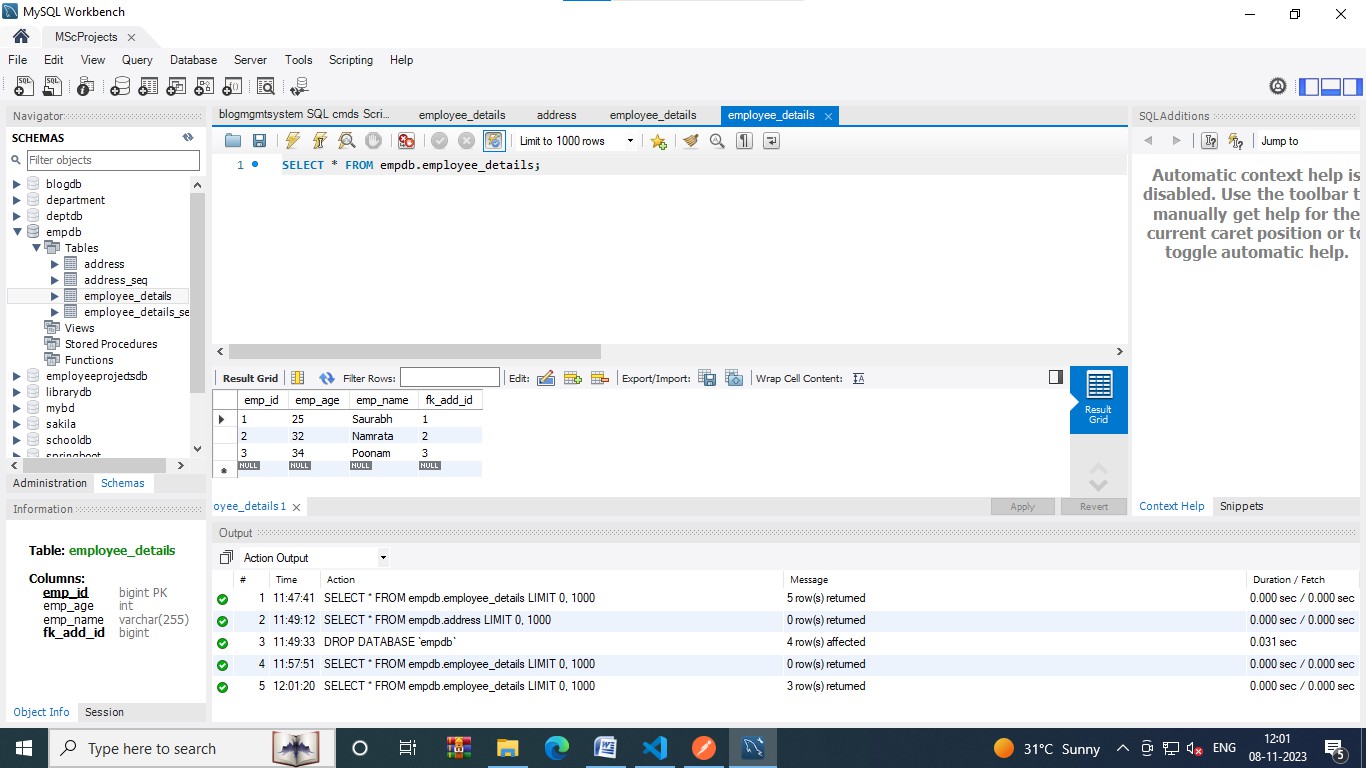
spring.datasource.url=jdbc:mysql://localhost:3306/empdb?createDatabaseIfNotExist=true

spring.datasource.username=root spring.datasource.password=msc@1234 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.jpa.show-sql:true

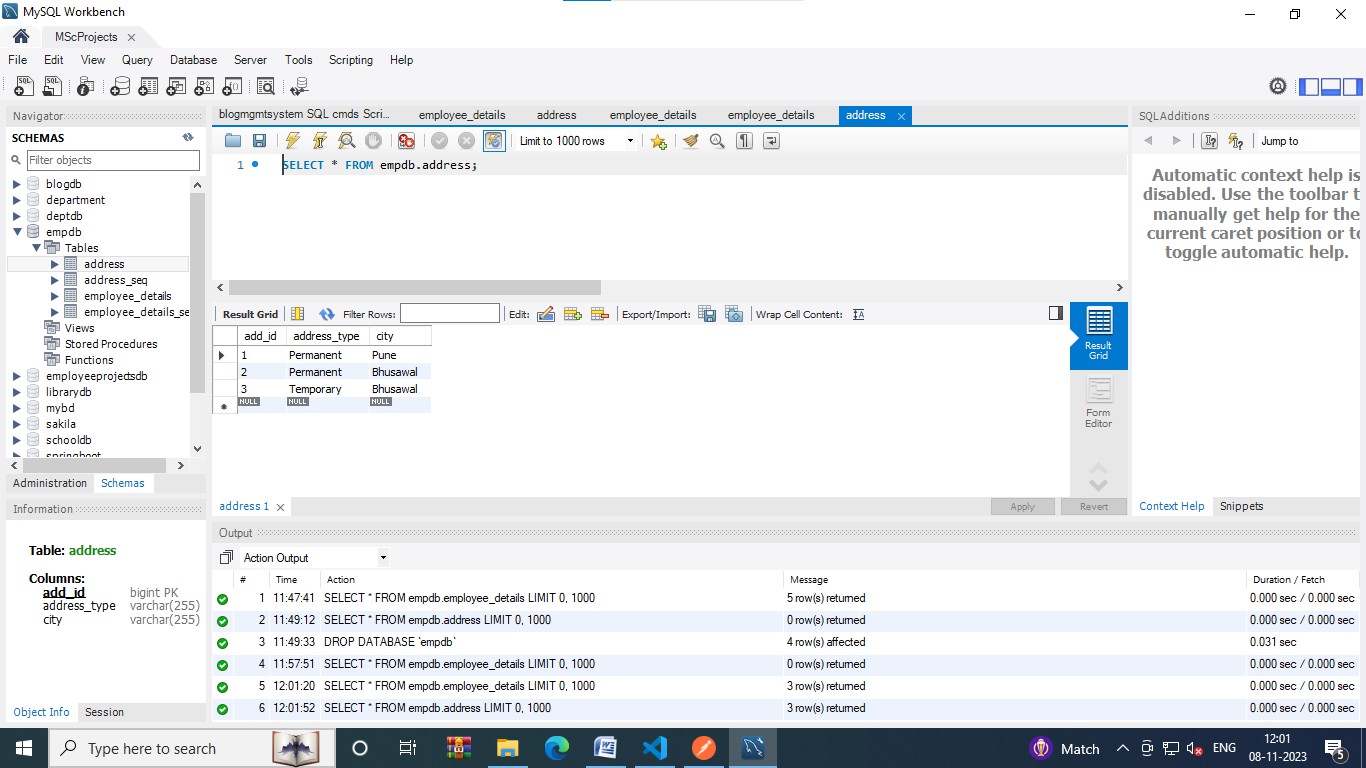
**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**//Auto-updating MYSQL database (Backend)**

**// Table: empdb.employee\_details**



**// Table: empdb.address**



# /\*\* Practical 5: Develop an employee entity where the user can create employee data. Use Hibernate to map the address entity and its relationship (one-to-many) with employee entity. \*\*/

**//Employee.java**

package com.example.employeeprojects.entity;

import jakarta.persistence.CascadeType; import jakarta.persistence.Column; import jakarta.persistence.Entity;

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

import jakarta.persistence.JoinColumn; import jakarta.persistence.OneToMany; import jakarta.persistence.OneToOne; import jakarta.persistence.Table;

import jakarta.transaction.Transactional; import lombok.Data;

import lombok.NoArgsConstructor; import java.util.List;

@Entity @Transactional @Data @NoArgsConstructor

@Table(name = "employee\_details") public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.AUTO) @Column(name = "emp\_id")

private Long empId; private String empName; private Integer empAge;

@OneToMany(cascade = CascadeType.ALL)

@JoinColumn(name = "fk\_emp\_id", referencedColumnName = "emp\_id") private List<Address> address;

}

**//Address.java**

package com.example.employeeprojects.entity;

import jakarta.persistence.Column; import jakarta.persistence.Entity;

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

//import jakarta.persistence.OneToOne; import jakarta.persistence.Table;

import jakarta.transaction.Transactional; import lombok.Data;

import lombok.NoArgsConstructor;

@Entity @Transactional @Data @NoArgsConstructor

@Table(name = "address") public class Address {

@Id

@GeneratedValue(strategy = GenerationType.AUTO) @Column(name = "add\_id")

private Long addressid; private String city;

private String addressType;

}

**//EmployeeRepository.java**

package com.example.employeeprojects.repository;

import org.springframework.data.jpa.repository.JpaRepository; import com.example.employeeprojects.entity.Employee;

public interface EmployeeRepository extends JpaRepository<Employee,Long>{

}

**//AddressRepository.java**

package com.example.employeeprojects.repository;

import org. springframework.data.jpa.repository.JpaRepository; import com.example.employeeprojects.entity.Address;

public interface AddressRepository extends JpaRepository<Address,Long>{

}

**//EmployeeController.java**

package com.example.employeeprojects.controller;

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

import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestBody; import org.springframework.web.bind.annotation.RestController; import java.util.List;

import com.example.employeeprojects.entity.Employee;

import com.example.employeeprojects.repository.EmployeeRepository;

@RestController

public class EmployeeController { @Autowired

private EmployeeRepository employeeRepository;

@PostMapping("/saveEmployees")

public ResponseEntity<String> saveEmployees(@RequestBody List<Employee> empData){ employeeRepository.saveAll(empData);

return ResponseEntity.ok("Data Saved");

}

}

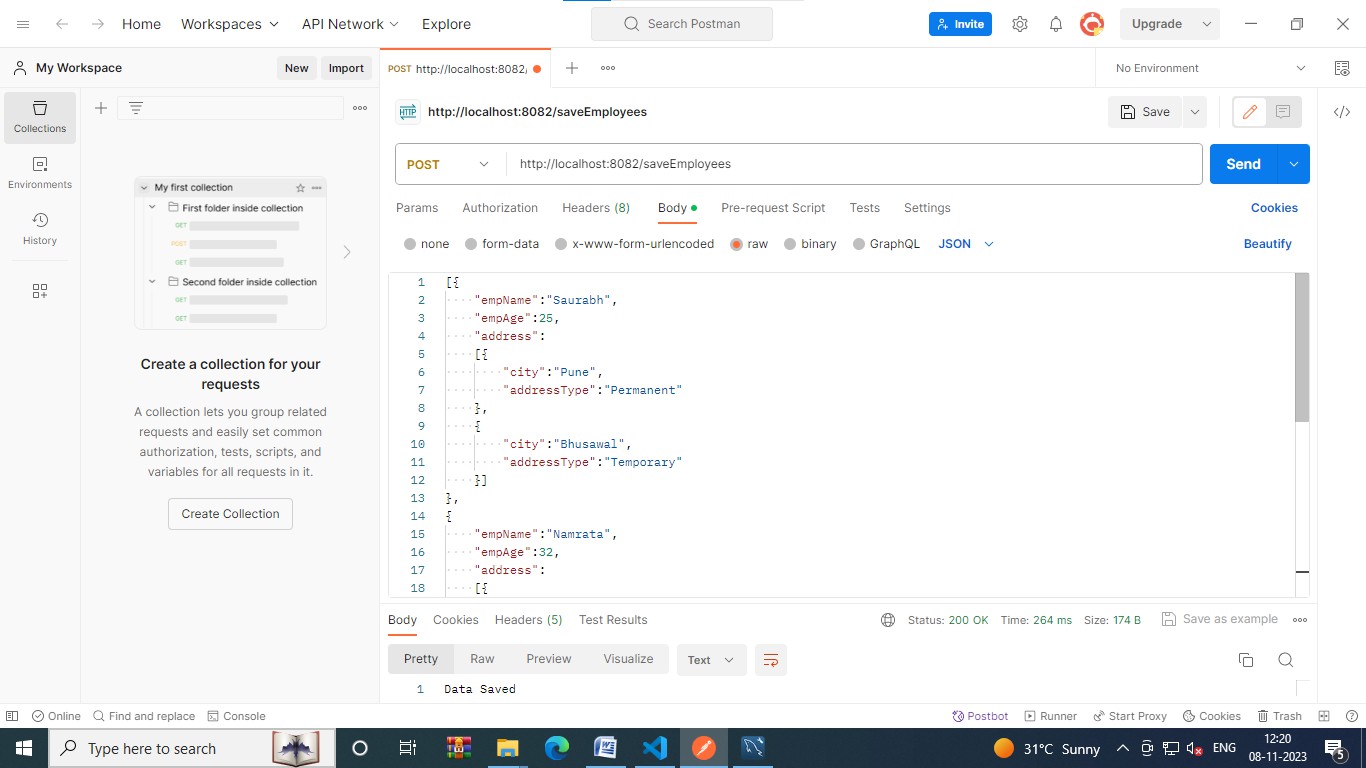
**//application.properties**

server.port=8082

# Configuration for MySQL Database spring.jpa.hibernate.ddl-auto=update

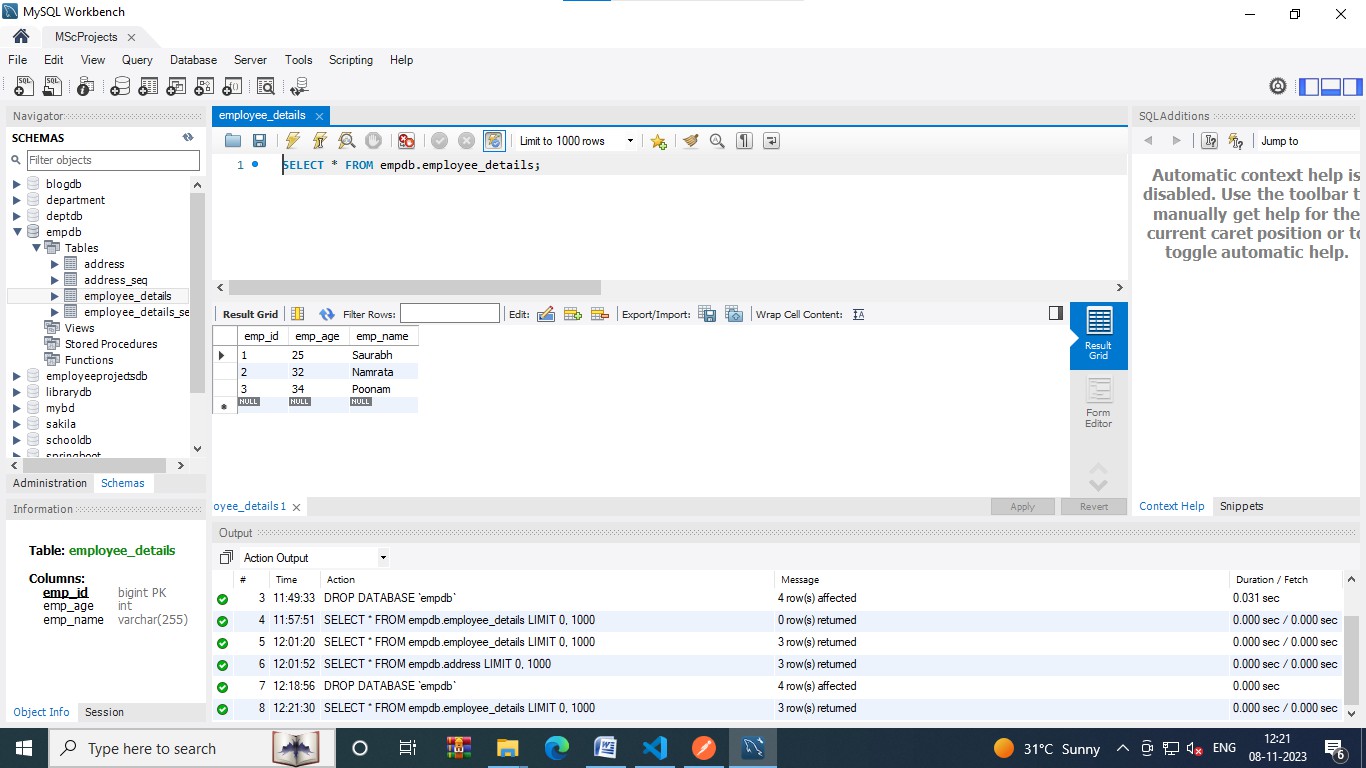
spring.datasource.url=jdbc:mysql://localhost:3306/empdb?createDatabaseIfNotExist=true spring.datasource.username=root

spring.datasource.password=msc@1234 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.jpa.show-sql:true

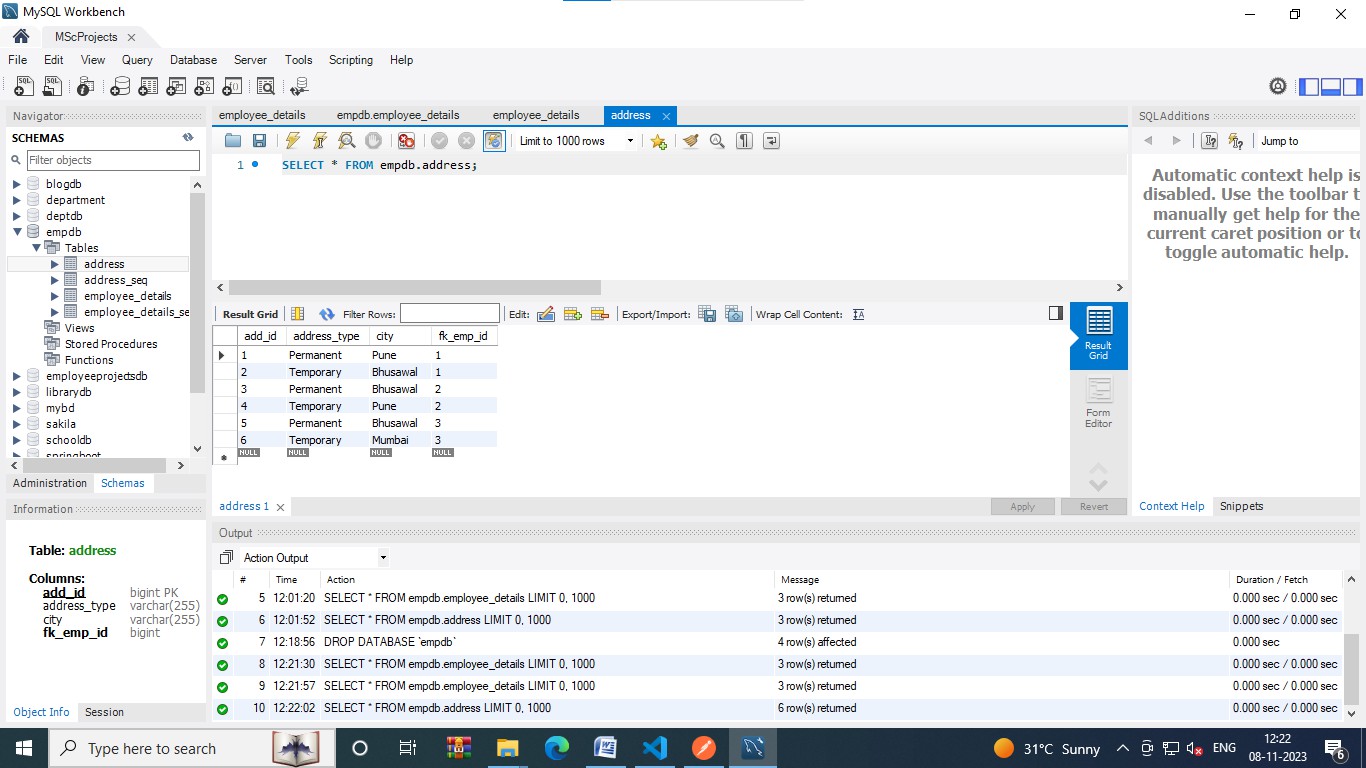
**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**//Auto-updating MYSQL database (Backend)**

**// Table: empdb.employee\_details**



**// Table: empdb.address**



# /\*\*Practical 6: Develop the micro service for employee management implement endpoint for retrieve employee details. Utilize spring boot and spring JPA to store and retrieve employee data. \*\*/

**// Employee.java**

package com.example.employeemicroserviceproject.entity;

import jakarta.persistence.\*; @Entity

@Table(name = "employee") public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY) @Column(name = "id")

private int id;

@Column(name = "name") private String name;

@Column(name = "email") private String email;

@Column(name = "age") private String 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;

}

public String getEmail() { return email;

}

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

}

public String getAge() { return age;

}

public void setAge(String age) { this.age = age;

}

}

**//EmployeeRepository.java**

package com.example.employeemicroserviceproject.repository; import org.springframework.data.jpa.repository.JpaRepository; import com.example.employeemicroserviceproject.entity.Employee;

public interface EmployeeRepository extends JpaRepository<Employee,Long>{

}

**//EmployeeService.java**

package com.example.employeemicroserviceproject.service;

import com.example.employeemicroserviceproject.entity.Employee;

import com.example.employeemicroserviceproject.repository.EmployeeRepository; import com.example.employeemicroserviceproject.response.EmployeeResponse; import org.modelmapper.ModelMapper;

import org.springframework.beans.factory.annotation.Autowired; import java.util.Optional;

public class EmployeeService { @Autowired

private EmployeeRepository employeeRepo;

@Autowired

private ModelMapper mapper;

public EmployeeResponse getEmployeeById(long id) { Optional<Employee> employee = employeeRepo.findById(id); EmployeeResponse employeeResponse = mapper.map(employee,

EmployeeResponse.class); return employeeResponse;

}

}

**//EmployeeController.java**

package com.example.employeemicroserviceproject.controller;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RestController;

import com.example.employeemicroserviceproject.response.EmployeeResponse; import com.example.employeemicroserviceproject.service.EmployeeService;

@RestController

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping("/employees/{id}")

private ResponseEntity<EmployeeResponse> getEmployeeDetails(@PathVariable("id") int id) {

EmployeeResponse employee = employeeService.getEmployeeById(id); return ResponseEntity.status(HttpStatus.OK).body(employee);

}

}

**//EmployeeResponse.java**

package com.example.employeemicroserviceproject.response; public class EmployeeResponse {

private int id; private String name; private String email; private String 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;

}

public String getEmail() { return email;

}

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

}

public String getAge() { return age;

}

public void setAge(String age) { this.age = age;

}

}

**//EmployeeConfig.java**

package com.example.employeemicroserviceproject.configuration;

import org.modelmapper.ModelMapper;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import com.example.employeemicroserviceproject.service.EmployeeService;

@Configuration

public class EmployeeConfig {

@Bean

public EmployeeService employeeBean() { return new EmployeeService();

}

@Bean

public ModelMapper modelMapperBean() { return new ModelMapper();

}

}

**//application.properties**

server.port=8086

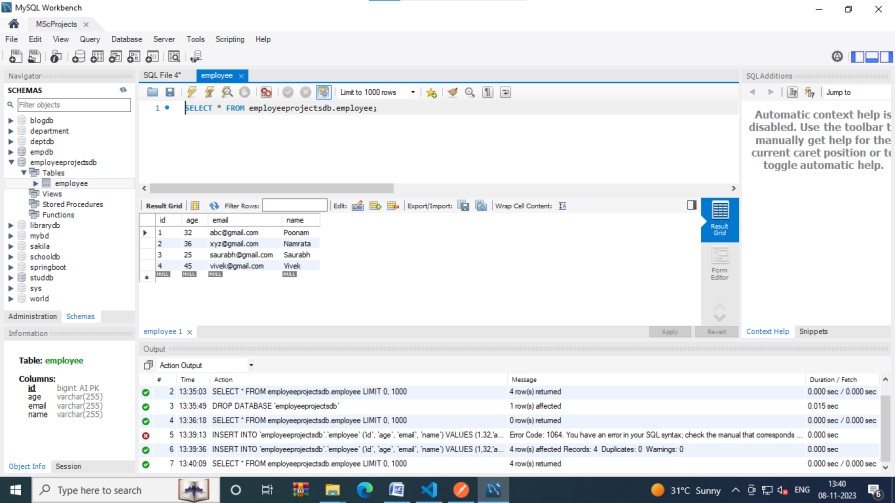
# Configuration for MySQL Database spring.jpa.hibernate.ddl-auto=update

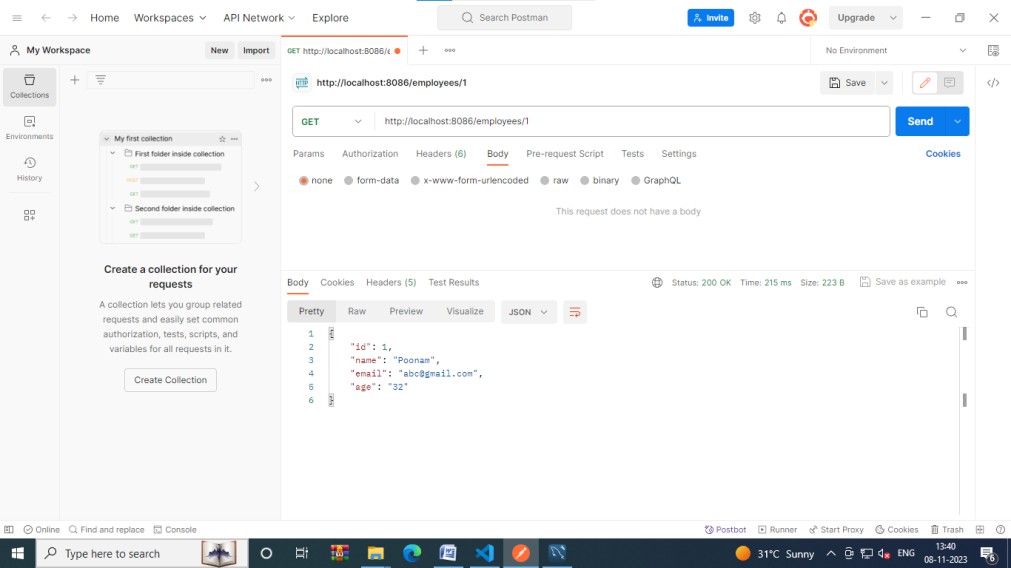
spring.datasource.url=jdbc:mysql://localhost:3306/employeeprojectsdb?createDatabaseIfNotExis t=true

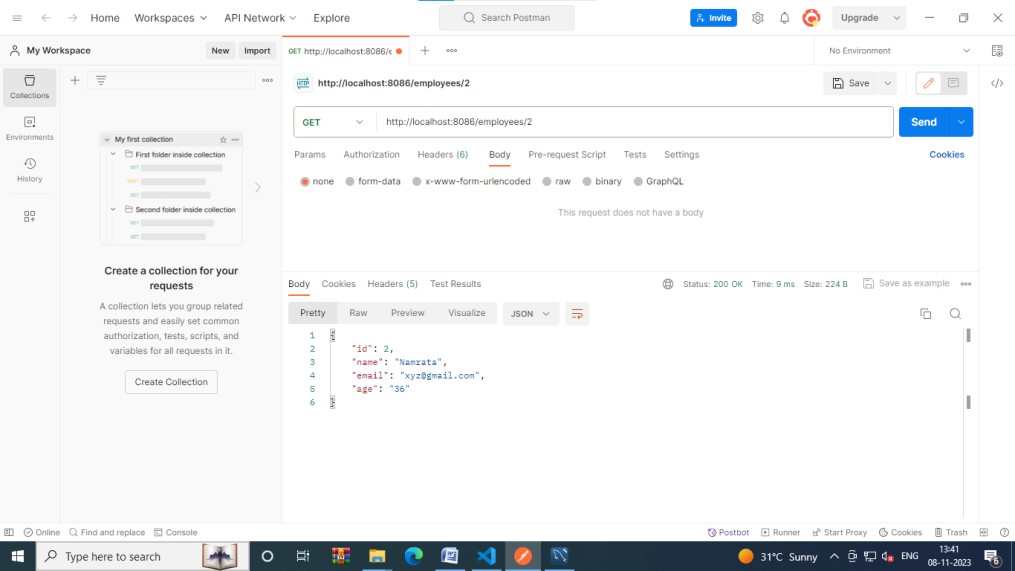
spring.datasource.username=root spring.datasource.password=msc@1234 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.jpa.show-sql:true

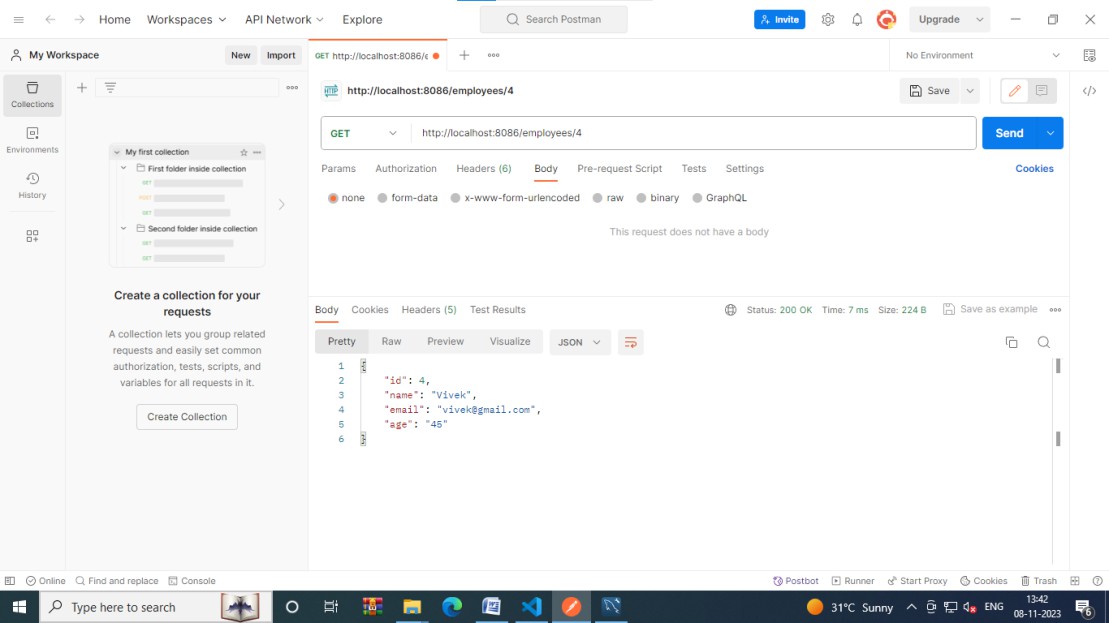
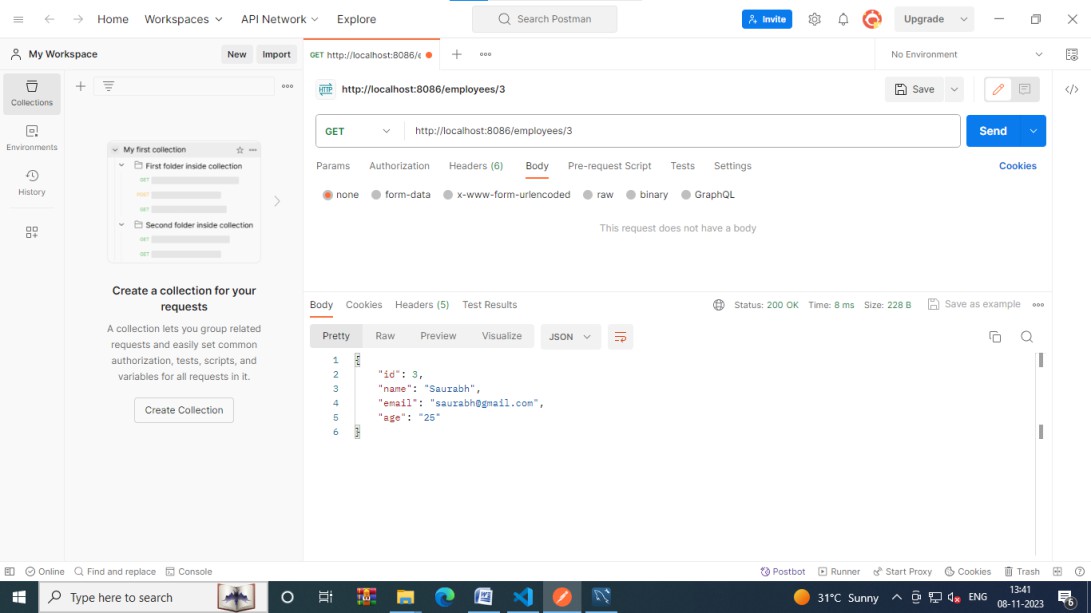
**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**// Data Stored at MYSQL Table**

**// Table: employee**

**// Get Operation at Postman**





# /\*\*Practical 7: Develop a library management system where users can borrow and return books. Use Hibernate to map the "Book" and "User" entities and their relationship as a many-to-many association. Implement CRUD operations to manage books and user records, as well as handling book borrowing and returning operations. \*\*/

**//Book.java**

package com.example.librarymgmtsystem.entity;

import java.util.\*;

import jakarta.persistence.\*;

@Entity

public class Book { @Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Long id;

public Book(Long id, String title, Set<User> borrowers) { this.id = id;

this.title = title; this.borrowers = borrowers;

}

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

}

public Long getId() { return id;

}

private String title; public String getTitle() {

return title;

}

public void setTitle(String title) { this.title = title;

}

@ManyToMany(mappedBy = "borrowedBooks") private Set<User> borrowers = new HashSet<>(); public Book() {

}

}

**//User.java**

package com.example.librarymgmtsystem.entity; import java.util.\*;

import jakarta.persistence.\*;

@Entity

public class User { @Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Long id;

public Long getId() { return id;

}

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

}

public User(Long id, String username, Set<Book> borrowedBooks) { this.id = id;

this.username = username; this.borrowedBooks = borrowedBooks;

}

private String username;

public String getUsername() { return username;

}

public void setUsername(String username) { this.username = username;

}

@ManyToMany @JoinTable(name = "user\_books",

joinColumns = @JoinColumn(name = "user\_id"), inverseJoinColumns = @JoinColumn(name = "book\_id"))

private Set<Book> borrowedBooks = new HashSet<>();

public User() {

}

public void setBorrowedBooks(Set<Book> borrowedBooks) { this.borrowedBooks = borrowedBooks;

}

public Set<Book> getBorrowedBooks() { return borrowedBooks;

}

}

**//BookRepository.java**

package com.example.librarymgmtsystem.repository;

import org.springframework.data.jpa.repository.JpaRepository; import com.example.librarymgmtsystem.entity.Book;

public interface BookRepository extends JpaRepository<Book,Long>{

}

**//UserRepository.java**

package com.example.librarymgmtsystem.repository;

import org.springframework.data.jpa.repository.JpaRepository; import com.example.librarymgmtsystem.entity.User;

public interface UserRepository extends JpaRepository<User,Long>{

}

**// BookService.java**

package com.example.librarymgmtsystem.service;

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

import com.example.librarymgmtsystem.entity.Book;

import com.example.librarymgmtsystem.exception.BookNotFoundException; import com.example.librarymgmtsystem.repository.BookRepository;

@Service

public class BookService {

private BookRepository bookRepository;

@Autowired

public BookService(BookRepository bookRepository) { this.bookRepository = bookRepository;

}

public Book createBook(Book book) { return bookRepository.save(book);

}

public Book getBookById(Long bookId) {

return bookRepository.findById(bookId).orElseThrow(() -> new BookNotFoundException("Book with ID " + bookId + " not found."));

}

public Book updateBook(Long bookId, Book updatedBook) { Book existingBook = getBookById(bookId); existingBook.setTitle(updatedBook.getTitle());

// Update other book properties as needed

return bookRepository.save(existingBook);

}

public void deleteBook(Long bookId) { Book book = getBookById(bookId); bookRepository.delete(book);

}

}

**// UserService.java**

package com.example.librarymgmtsystem.service;

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

import com.example.librarymgmtsystem.entity.Book; import com.example.librarymgmtsystem.entity.User;

import com.example.librarymgmtsystem.exception.BookNotFoundException; import com.example.librarymgmtsystem.exception.UserNotFoundException;

import com.example.librarymgmtsystem.repository.BookRepository; import com.example.librarymgmtsystem.repository.UserRepository;

import java.util.HashSet; import java.util.Set;

@Service

public class UserService {

private UserRepository userRepository; private BookRepository bookRepository;

@Autowired

public UserService(UserRepository userRepository, BookRepository bookRepository) { this.userRepository = userRepository;

this.bookRepository = bookRepository;

}

public User createUser(User user) { return userRepository.save(user);

}

public User getUserById(Long userId) {

return userRepository.findById(userId).orElseThrow(() -> new UserNotFoundException("User with ID " + userId + " not found."));

}

public Set<Book> getBorrowedBooks(Long userId) { User user = getUserById(userId);

return user.getBorrowedBooks();

}

public void borrowBook(Long userId, Long bookId) { User user = getUserById(userId);

Book book = bookRepository.findById(bookId).orElseThrow(() -> new BookNotFoundException("Book with ID " + bookId + " not found."));

Set<Book> borrowedBooks = user.getBorrowedBooks(); borrowedBooks.add(book); user.setBorrowedBooks(borrowedBooks);

userRepository.save(user);

}

public void returnBook(Long userId, Long bookId) { User user = getUserById(userId);

Book book = bookRepository.findById(bookId).orElseThrow(() -> new BookNotFoundException("Book with ID " + bookId + " not found."));

Set<Book> borrowedBooks = user.getBorrowedBooks(); borrowedBooks.remove(book); user.setBorrowedBooks(borrowedBooks); userRepository.save(user);

}

}

**// LibraryService.java**

package com.example.librarymgmtsystem.service;

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

import com.example.librarymgmtsystem.entity.Book; import com.example.librarymgmtsystem.entity.User;

import com.example.librarymgmtsystem.exception.BookNotFoundException; import com.example.librarymgmtsystem.exception.UserNotFoundException; import com.example.librarymgmtsystem.repository.BookRepository;

import com.example.librarymgmtsystem.repository.UserRepository;

@Service

public class LibraryService { @Autowired

private BookRepository bookRepository;

@Autowired

private UserRepository userRepository;

public void borrowBook(Long userId, Long bookId) {

User user = userRepository.findById(userId).orElseThrow(UserNotFoundException::new); Book book =

bookRepository.findById(bookId).orElseThrow(BookNotFoundException::new); user.getBorrowedBooks().add(book);

userRepository.save(user);

}

public void returnBook(Long userId, Long bookId) {

User user = userRepository.findById(userId).orElseThrow(UserNotFoundException::new); Book book =

bookRepository.findById(bookId).orElseThrow(BookNotFoundException::new); user.getBorrowedBooks().remove(book);

userRepository.save(user);

}

}

**// BookNotFoundException.java**

package com.example.librarymgmtsystem.exception;

public class BookNotFoundException extends RuntimeException { public BookNotFoundException() {

super("Book not found.");

}

public BookNotFoundException(String message) { super(message);

}

public BookNotFoundException(String message, Throwable cause) { super(message, cause);

}

}

**// UserNotFoundException.java**

package com.example.librarymgmtsystem.exception;

public class UserNotFoundException extends RuntimeException { public UserNotFoundException() {

super("User not found.");

}

public UserNotFoundException(String message) { super(message);

}

public UserNotFoundException(String message, Throwable cause) {

super(message, cause);

}

}

**// LibraryController.java**

package com.example.librarymgmtsystem.controller;

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

import org.springframework.web.bind.annotation.\*;

import com.example.librarymgmtsystem.entity.Book; import com.example.librarymgmtsystem.entity.User;

import com.example.librarymgmtsystem.service.BookService; import com.example.librarymgmtsystem.service.UserService;

import java.util.Set;

@RestController @RequestMapping("/library") public class LibraryController {

@Autowired

private UserService userService;

@Autowired

private BookService bookService;

// Create a new user @PostMapping("/users")

public ResponseEntity<String> createUser(@RequestBody User user) { userService.createUser(user);

return ResponseEntity.ok("User Saved");

}

// Get a user by ID @GetMapping("/users/{userId}")

public User getUser(@PathVariable Long userId) { return userService.getUserById(userId);

}

// Get borrowed books by user @GetMapping("/users/{userId}/borrowed-books")

public Set<Book> getBorrowedBooks(@PathVariable Long userId) { return userService.getBorrowedBooks(userId);

}

// Create a new book @PostMapping("/books")

public ResponseEntity<String> createBook(@RequestBody Book book) { bookService.createBook(book);

return ResponseEntity.ok("Book Saved");

}

// Get a book by ID @GetMapping("/books/{bookId}")

public Book getBook(@PathVariable Long bookId) { return bookService.getBookById(bookId);

}

// Borrow a book @PostMapping("/users/{userId}/borrow/{bookId}")

public ResponseEntity<String> borrowBook(@PathVariable Long userId, @PathVariable Long bookId) {

userService.borrowBook(userId, bookId); return ResponseEntity.ok("Book borrowed");

}

// Return a book @PostMapping("/users/{userId}/return/{bookId}")

public ResponseEntity<String> returnBook(@PathVariable Long userId, @PathVariable Long bookId) {

userService.returnBook(userId, bookId); return ResponseEntity.ok("Book returned");

}

}

**// application.properties**

server.port=8082

# Configuration for MySQL Database spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/librarydb?createDatabaseIfNotExist=true spring.datasource.username=root

spring.datasource.password=msc@1234 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.jpa.show-sql:true

**// pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="[http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation=["http://maven.apa](http://maven.apache.org/POM/4.0.0)c[he.org/POM/4.0.0](http://maven.apache.org/POM/4.0.0) https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>librarymgmtsystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>librarymgmtsystem</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

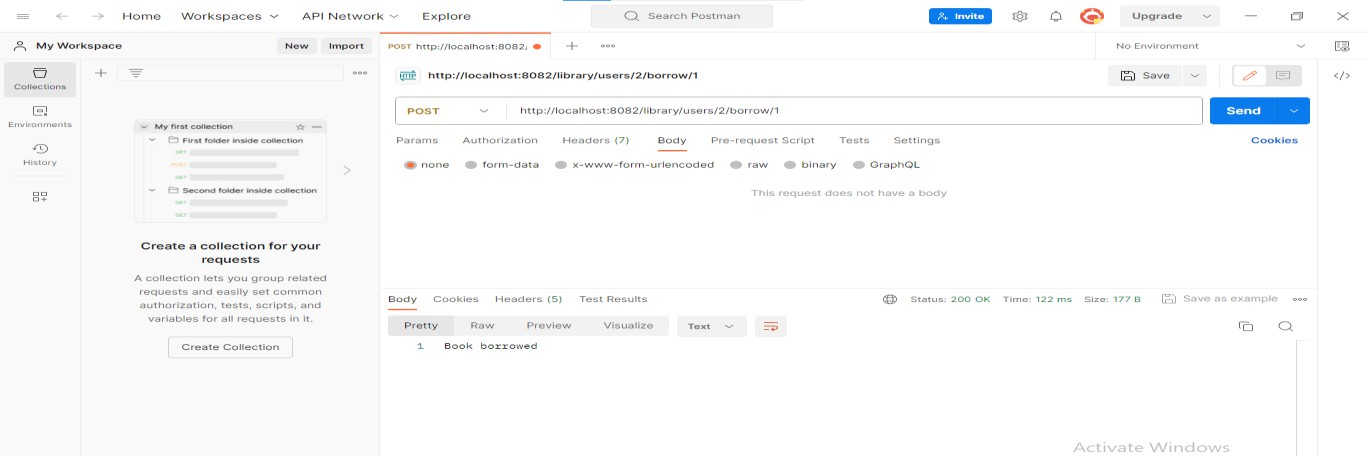
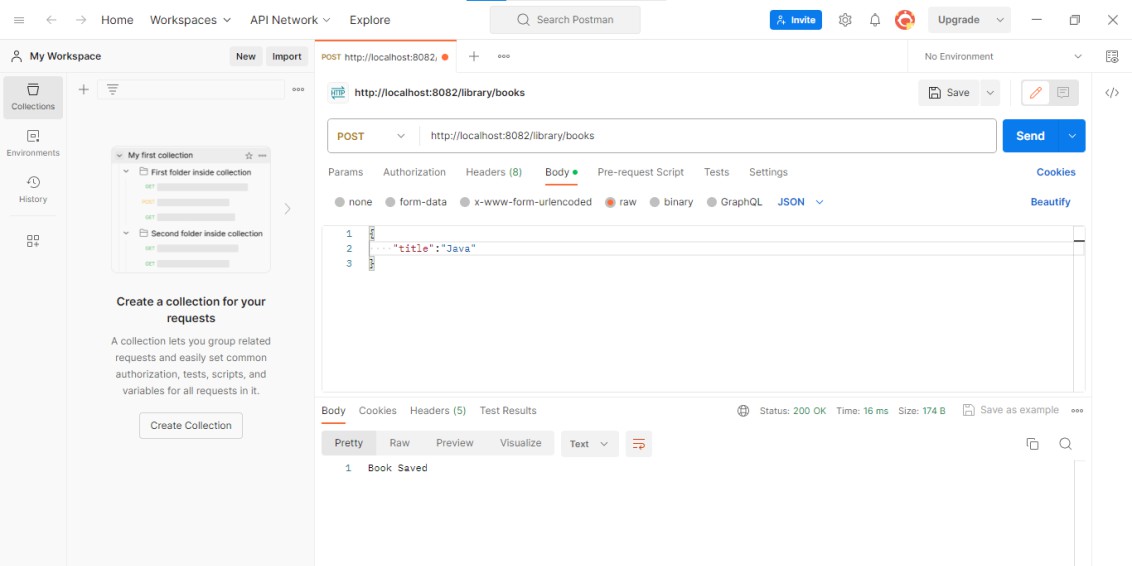
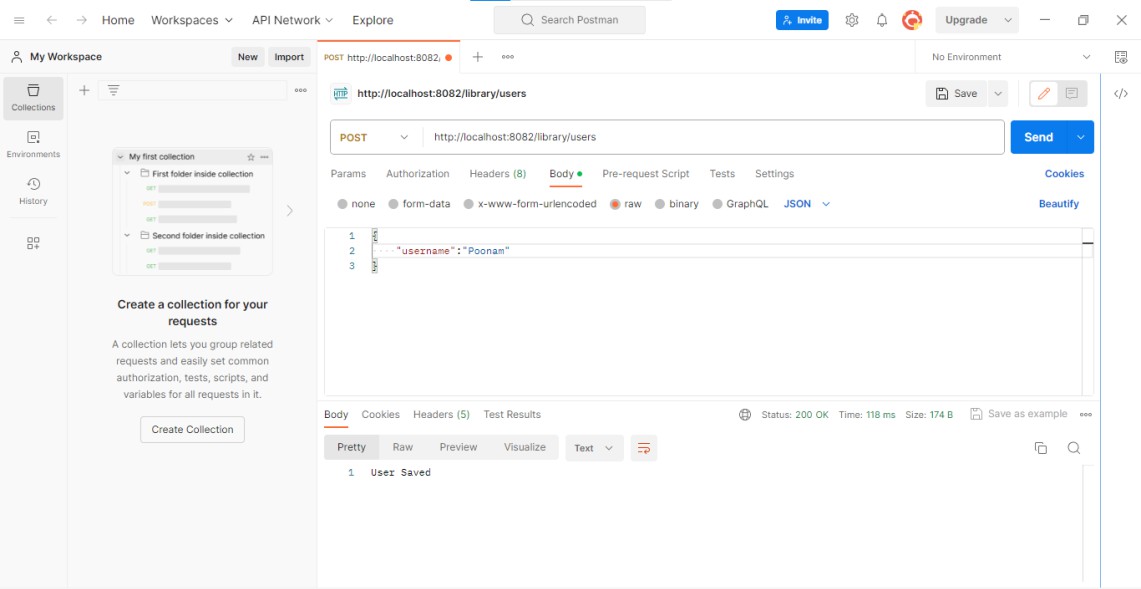
</plugins>

</build>

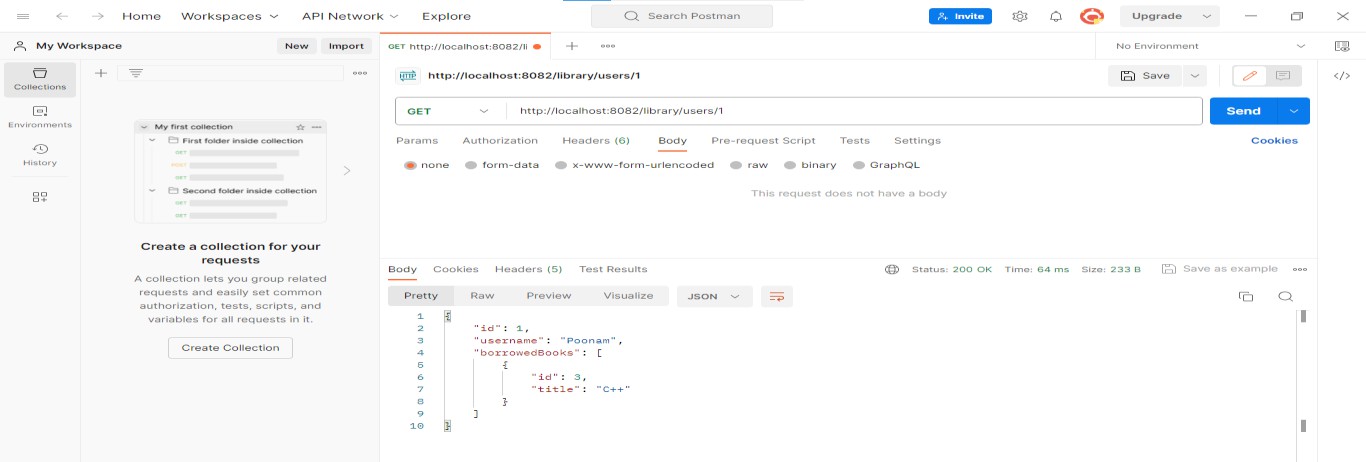
</project>

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**// Create Operation**

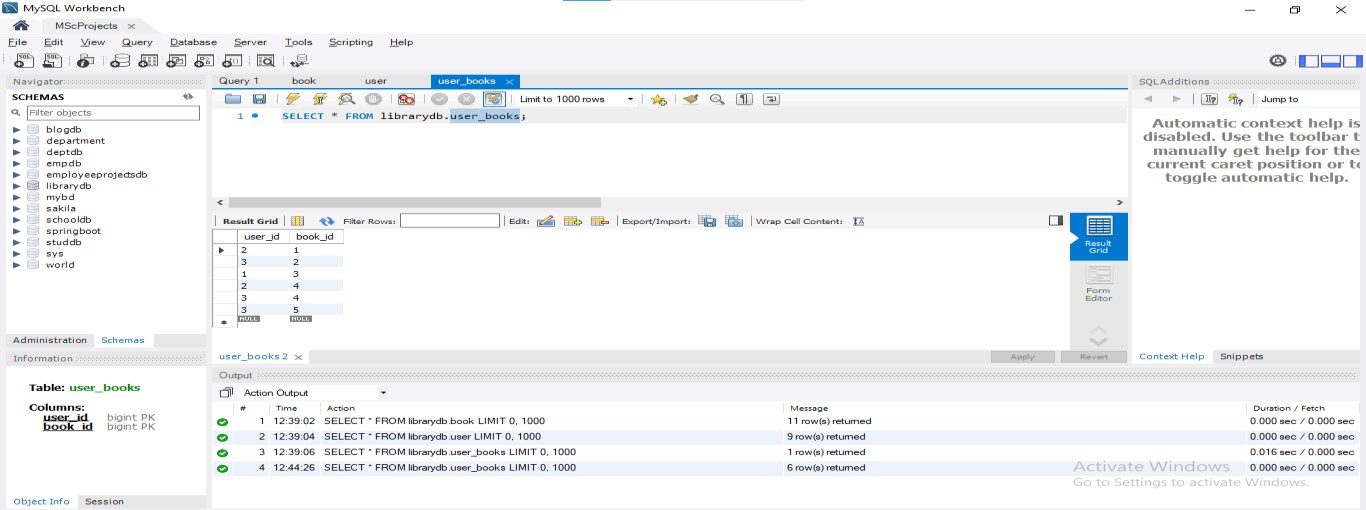


**//Retrieve Operation**



**//Auto-updating MYSQL database (Backend)**

**// Table: user\_books**



# /\*\*Practical 8: Build a blog management system where users can create, read, update, and delete blog posts. Use Hibernate to map the "Blog" entity and its relationships with other entities such as "User" and "Comment." Implement CRUD operations to manage blog posts.

**//Blog.java**

package com.example.blogmgmtsystem.entity; import java.util.\*;

import jakarta.persistence.\*;

@Entity

public class Blog { @Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Long id;

public Long getId() { return id;

}

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

}

private String title; public String getTitle() {

return title;

}

public void setTitle(String title) { this.title = title;

}

private String content;

// Other blog properties

public String getContent() { return content;

}

public void setContent(String content) { this.content = content;

}

@ManyToOne @JoinColumn(name = "user\_id") private User author;

@OneToMany(mappedBy = "blog")

private List<Comment> comments = new ArrayList<>();

// Other annotations, constructors, getters, and setters public Blog() {

}

public List<Comment> getComments() { return comments;

}

public void setAuthor(User author) { this.author = author;

}

}

**//Comment.java**

package com.example.blogmgmtsystem.entity; import jakarta.persistence.\*;

@Entity

public class Comment { @Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Long id;

public Long getId() { return id;

}

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

}

private String content;

// Other comment properties

public String getContent() { return content;

}

public void setContent(String content) { this.content = content;

}

@ManyToOne @JoinColumn(name = "blog\_id") private Blog blog;

// Other annotations, constructors, getters, and setters public Comment() {

}

public void setBlog(Blog blog) { this.blog = blog;

}

}

**//User.java**

package com.example.blogmgmtsystem.entity; import java.util.\*;

import jakarta.persistence.\*;

@Entity

public class User { @Id

@GeneratedValue(strategy = GenerationType.IDENTITY) private Long id;

public Long getId() { return id;

}

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

}

private String username;

// Other user properties

public String getUsername() { return username;

}

public void setUsername(String username) { this.username = username;

}

@OneToMany(mappedBy = "author")

private List<Blog> blogs = new ArrayList<>(); public List<Blog> getBlogs() {

return blogs;

}

public void setBlogs(List<Blog> blogs) { this.blogs = blogs;

}

public User(Long id, String username, List<Blog> blogs) { this.id = id;

this.username = username; this.blogs = blogs;

}

// Other annotations, constructors, getters, and setters public User() {

}

}

**//BlogRepository.java**

package com.example.blogmgmtsystem.repository;

import org.springframework.data.jpa.repository.JpaRepository; import com.example.blogmgmtsystem.entity.Blog;

public interface BlogRepository extends JpaRepository<Blog, Long>{

}

**//CommentRepository.java**

package com.example.blogmgmtsystem.repository; import java.util.List;

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

import com.example.blogmgmtsystem.entity.Blog; import com.example.blogmgmtsystem.entity.Comment;

public interface CommentRepository extends JpaRepository<Comment, Long>{ List<Comment> findByBlog(Blog blog);

}

**//UserRepository.java**

package com.example.blogmgmtsystem.repository;

import org.springframework.data.jpa.repository.JpaRepository; import com.example.blogmgmtsystem.entity.User;

public interface UserRepository extends JpaRepository<User, Long> {

}

**//BlogService.java**

package com.example.blogmgmtsystem.service;

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

import com.example.blogmgmtsystem.entity.Blog; import com.example.blogmgmtsystem.entity.Comment;

import com.example.blogmgmtsystem.exception.BlogNotFoundException; import com.example.blogmgmtsystem.repository.BlogRepository;

import com.example.blogmgmtsystem.repository.CommentRepository;

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

@Service

public class BlogService {

private final BlogRepository blogRepository; private CommentRepository commentRepository;

@Autowired

public BlogService(BlogRepository blogRepository, CommentRepository commentRepository) {

this.blogRepository = blogRepository; this.commentRepository = commentRepository;

}

public Blog createBlog(Blog blog) { return blogRepository.save(blog);

}

public List<Blog> getAllBlogs() {

return blogRepository.findAll();

}

public Blog getBlogById(Long id) {

Optional<Blog> optionalBlog = blogRepository.findById(id); return optionalBlog.orElse(null);

}

public Blog updateBlog(Long id, Blog updatedBlog) { Optional<Blog> optionalBlog = blogRepository.findById(id); if (optionalBlog.isPresent()) {

Blog existingBlog = optionalBlog.get(); existingBlog.setTitle(updatedBlog.getTitle()); existingBlog.setContent(updatedBlog.getContent());

// Update other properties as needed return blogRepository.save(existingBlog);

} else {

return null; // Blog not found

}

}

public boolean deleteBlog(Long id) {

Optional<Blog> optionalBlog = blogRepository.findById(id); if (optionalBlog.isPresent()) {

blogRepository.deleteById(id);

return true; // Blog deleted successfully

} else {

return false; // Blog not found

}

}

public List<Comment> getCommentsForBlog(Long blogId) {

Blog blog = blogRepository.findById(blogId).orElseThrow(() -> new BlogNotFoundException("Blog with ID " + blogId + " not found."));

return commentRepository.findByBlog(blog);

}

public Comment addCommentToBlog(Long blogId, Comment comment) { Blog blog = blogRepository.findById(blogId).orElseThrow(() -> new

BlogNotFoundException("Blog with ID " + blogId + " not found."));

comment.setBlog(blog); // Associate the comment with the blog return commentRepository.save(comment);

}

}

**//CommentService.java**

package com.example.blogmgmtsystem.service; import java.util.List;

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

import com.example.blogmgmtsystem.entity.Blog; import com.example.blogmgmtsystem.entity.Comment;

import com.example.blogmgmtsystem.exception.BlogNotFoundException; import com.example.blogmgmtsystem.exception.CommentNotFoundException; import com.example.blogmgmtsystem.repository.BlogRepository;

import com.example.blogmgmtsystem.repository.CommentRepository;

@Service

public class CommentService {

private CommentRepository commentRepository; private BlogRepository blogRepository;

@Autowired

public CommentService(CommentRepository commentRepository, BlogRepository blogRepository) {

this.commentRepository = commentRepository; this.blogRepository = blogRepository;

}

public Comment createComment(Comment comment, Long blogId) { Blog blog = blogRepository.findById(blogId).orElseThrow(() -> new

BlogNotFoundException("Blog with ID " + blogId + " not found.")); comment.setBlog(blog);

return commentRepository.save(comment);

}

public Comment getCommentById(Long commentId) {

return commentRepository.findById(commentId).orElseThrow(() -> new CommentNotFoundException("Comment with ID " + commentId + " not found."));

}

public List<Comment> getCommentsByBlogId(Long blogId) {

Blog blog = blogRepository.findById(blogId).orElseThrow(() -> new BlogNotFoundException("Blog with ID " + blogId + " not found."));

return blog.getComments();

}

public Comment updateComment(Long commentId, Comment updatedComment) { Comment existingComment = getCommentById(commentId); existingComment.setContent(updatedComment.getContent());

// Update other comment properties as needed return commentRepository.save(existingComment);

}

public void deleteComment(Long commentId) { Comment comment = getCommentById(commentId); commentRepository.delete(comment);

}

}

**//UserService.java**

package com.example.blogmgmtsystem.service;

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

import com.example.blogmgmtsystem.entity.User;

import com.example.blogmgmtsystem.exception.UserNotFoundException; import com.example.blogmgmtsystem.repository.UserRepository;

import java.util.List; @Service

public class UserService {

private UserRepository userRepository; @Autowired

public UserService(UserRepository userRepository) { this.userRepository = userRepository;

}

public User createUser(User user) { return userRepository.save(user);

}

public User getUserById(Long userId) {

return userRepository.findById(userId).orElseThrow(() -> new UserNotFoundException("User with ID " + userId + " not found."));

}

public List<User> getAllUsers() { return userRepository.findAll();

}

public User updateUser(Long userId, User updatedUser) { User existingUser = getUserById(userId); existingUser.setUsername(updatedUser.getUsername());

// Update other user properties as needed return userRepository.save(existingUser);

}

public void deleteUser(Long userId) { User user = getUserById(userId); userRepository.delete(user);

}

}

**//BlogNotFoundException.java**

package com.example.blogmgmtsystem.exception;

public class BlogNotFoundException extends RuntimeException { public BlogNotFoundException() {

super("Blog not found.");

}

public BlogNotFoundException(String message) { super(message);

}

public BlogNotFoundException(String message, Throwable cause) { super(message, cause);

}

}

**//CommentNotFoundException.java**

package com.example.blogmgmtsystem.exception;

public class CommentNotFoundException extends RuntimeException { public CommentNotFoundException() {

super("Comment not found.");

}

public CommentNotFoundException(String message) { super(message);

}

public CommentNotFoundException(String message, Throwable cause) { super(message, cause);

}

}

**//UserNotFoundException.java**

package com.example.blogmgmtsystem.exception;

public class UserNotFoundException extends RuntimeException { public UserNotFoundException() {

super("User not found.");

}

public UserNotFoundException(String message) { super(message);

}

public UserNotFoundException(String message, Throwable cause) { super(message, cause);

}

}

**//BlogController.java**

package com.example.blogmgmtsystem.controller; import java.util.List;

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

import org.springframework.web.bind.annotation.\*;

import com.example.blogmgmtsystem.service.\*; import com.example.blogmgmtsystem.entity.\*;

@RestController @RequestMapping("/blogs") public class BlogController {

@Autowired

private BlogService blogService;

@Autowired

private UserService userService;

@Autowired

private CommentService commentService;

@PostMapping("/createUser")

public User createUser(@RequestBody User user) { return userService.createUser(user);

}

// Create a new blog post @PostMapping({"/{userId}"})

public ResponseEntity<String> createBlog(@RequestBody Blog blog, @RequestParam Long userId) {

User user = userService.getUserById(userId); // Get the user by ID blog.setAuthor(user); // Associate the blog with the user blogService.createBlog(blog);

return ResponseEntity.ok("Blog Post Created");

}

// Add a comment to a blog post @PostMapping("/{blogId}/comments")

public ResponseEntity<String> addCommentToBlog(@PathVariable Long blogId, @RequestBody Comment comment) {

blogService.addCommentToBlog(blogId, comment); return ResponseEntity.ok("Comment Posted");

}

// Get a blog post by ID @GetMapping("/{blogId}")

public Blog getBlog(@PathVariable Long blogId) { return blogService.getBlogById(blogId);

}

@GetMapping("/users/{userId}")

public User getUserById(@PathVariable Long userId) { return userService.getUserById(userId);

}

// Get all blog posts @GetMapping

public List<Blog> getAllBlogs() {

return blogService.getAllBlogs();

}

// Update a blog post @PutMapping("/{blogId}")

public ResponseEntity<String> updateBlog(@PathVariable Long blogId, @RequestBody Blog updatedBlog) {

blogService.updateBlog(blogId, updatedBlog); return ResponseEntity.ok("Blog Post Updated");

}

// Delete a blog post @DeleteMapping("/{blogId}")

public ResponseEntity<String> deleteBlog(@PathVariable Long blogId) { blogService.deleteBlog(blogId);

return ResponseEntity.ok("Blog Post Deleted");

}

// Get comments for a specific blog post @GetMapping("/{blogId}/comments")

public List<Comment> getCommentsForBlog(@PathVariable Long blogId) { return blogService.getCommentsForBlog(blogId);

}

//Delete a comment @DeleteMapping("/comments/{commentId}")

public ResponseEntity<String> deleteComment(@PathVariable Long commentId) { commentService.deleteComment(commentId);

return ResponseEntity.ok("Comment Deleted");

}

}

**//application.properties**

server.port=8084

# Configuration for MySQL Database spring.jpa.hibernate.ddl-auto=update

spring.datasource.url=jdbc:mysql://localhost:3306/blogdb?createDatabaseIfNotExist=true spring.datasource.username=root

spring.datasource.password=msc@1234 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver spring.jpa.show-sql:true

**//pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="[http://maven.apache.org/POM/4.0.0"](http://maven.apache.org/POM/4.0.0) xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation=["http://maven.apa](http://maven.apache.org/POM/4.0.0)c[he.org/POM/4.0.0](http://maven.apache.org/POM/4.0.0) https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.1.5</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example</groupId>

<artifactId>blogmgmtsystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>blogmgmtsystem</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

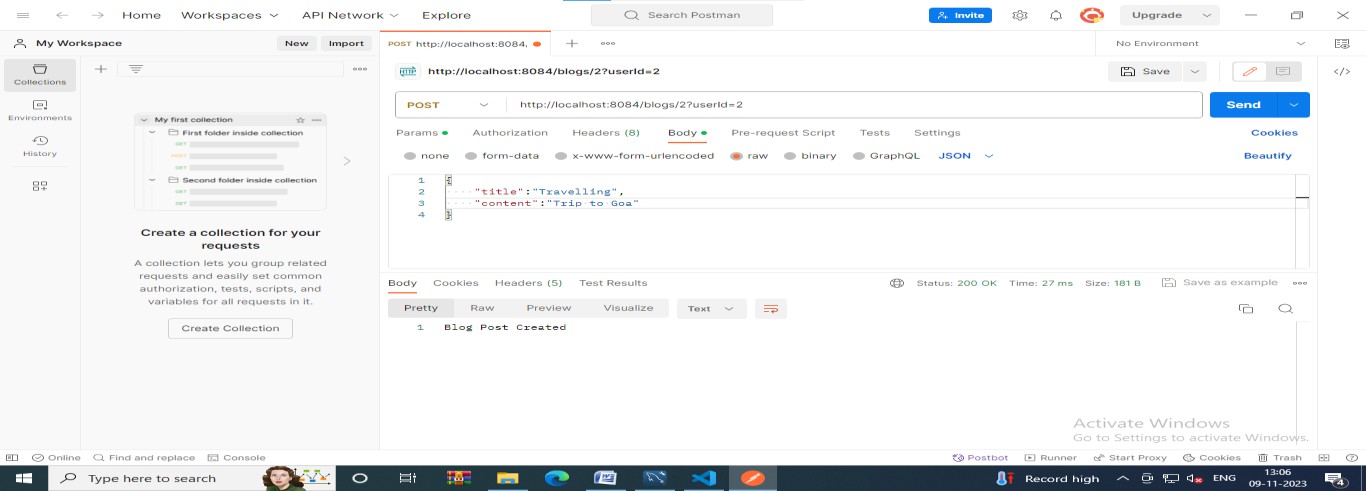
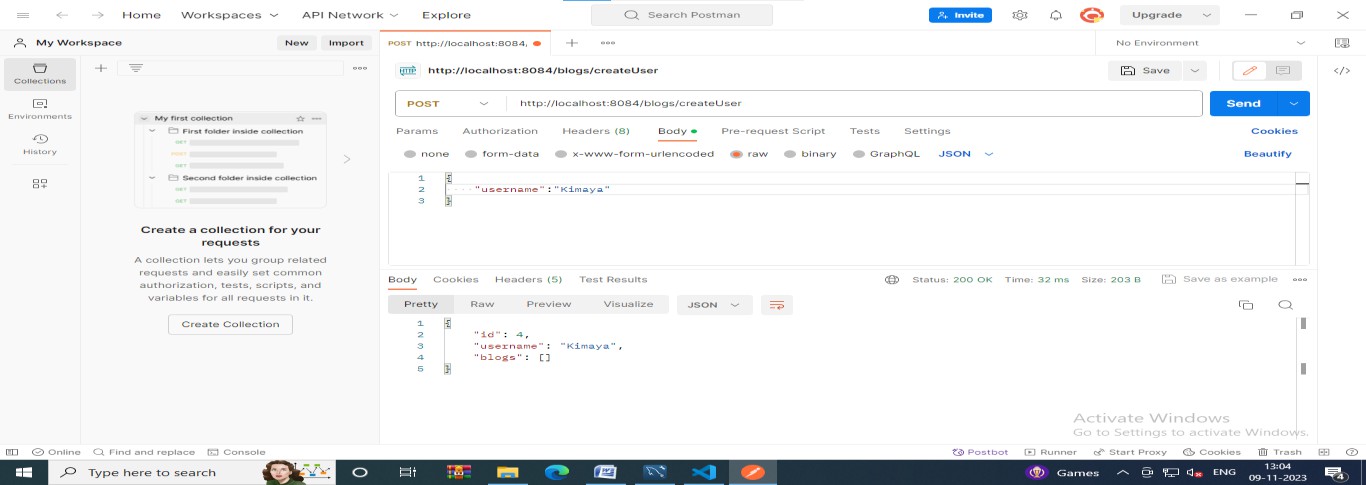
</plugins>

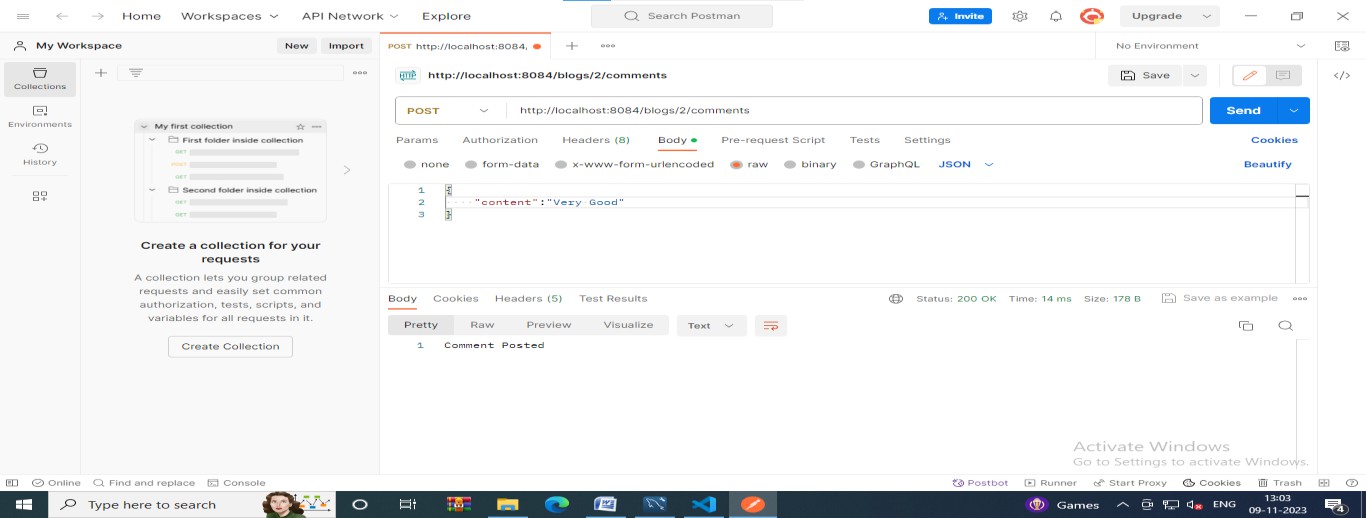
</build>

</project>

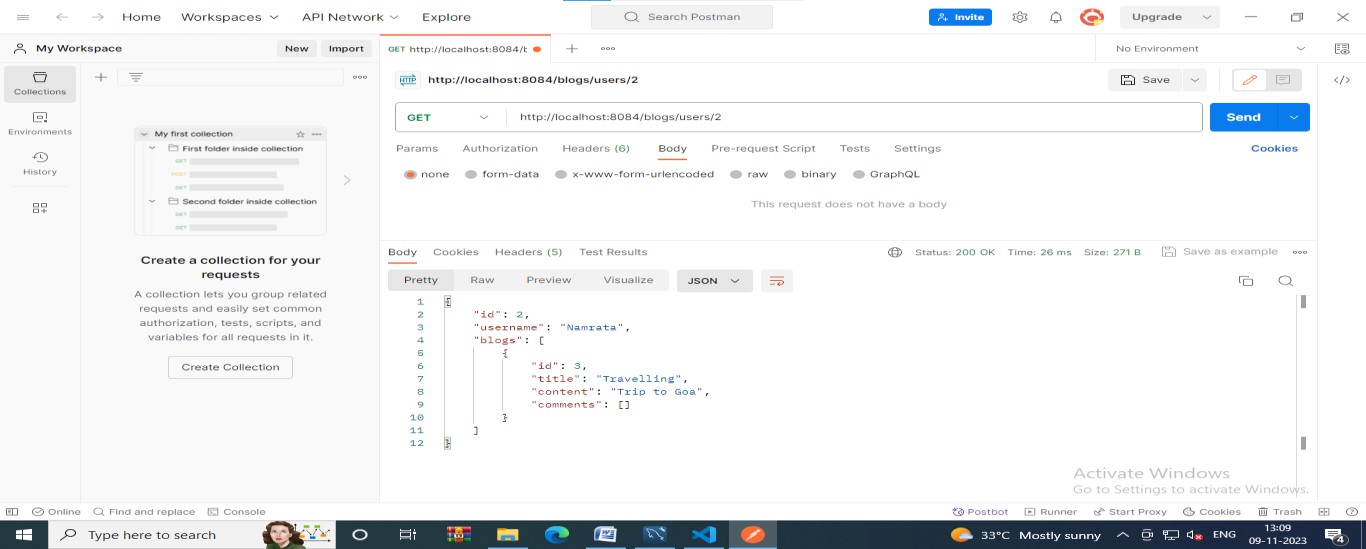
**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

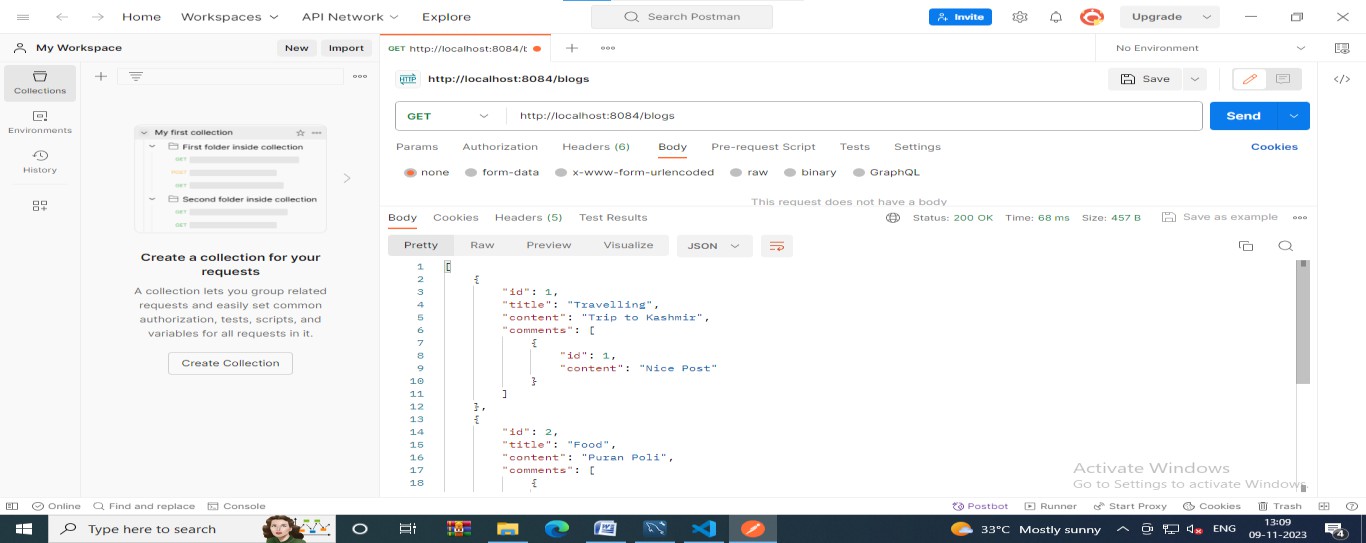
**//Create User, Blog and Comment**

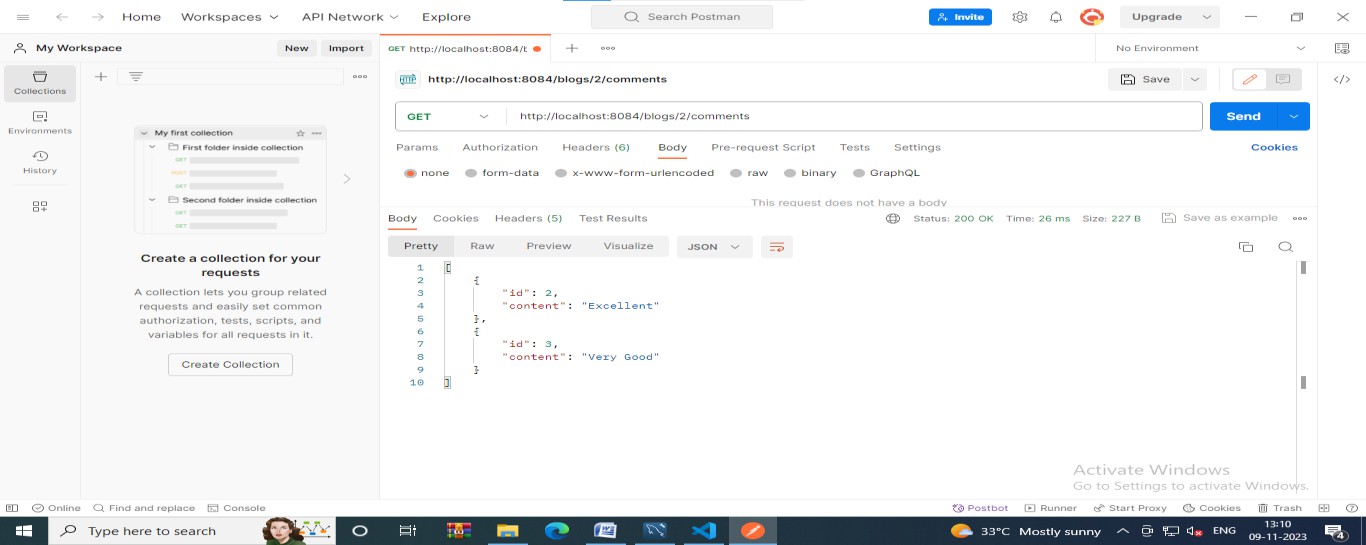




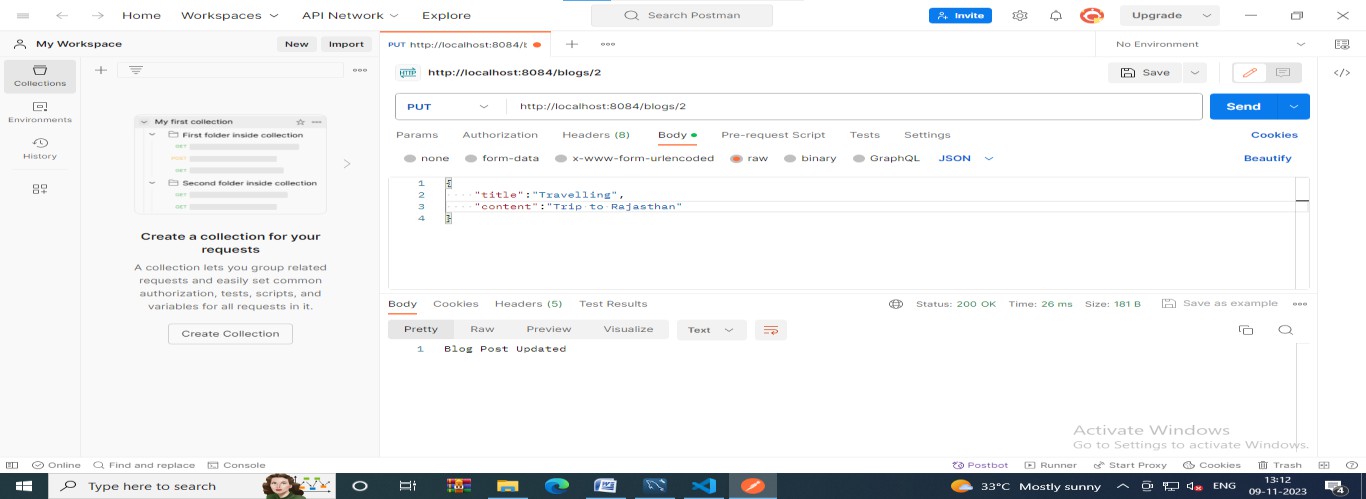
**// Retrieve Blogs, Users and Comments**



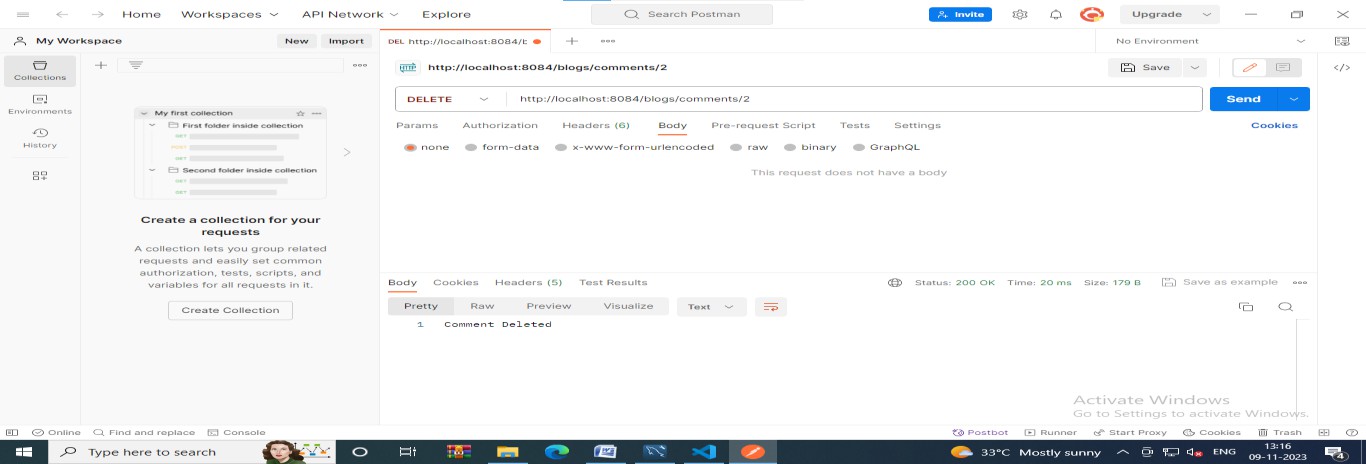




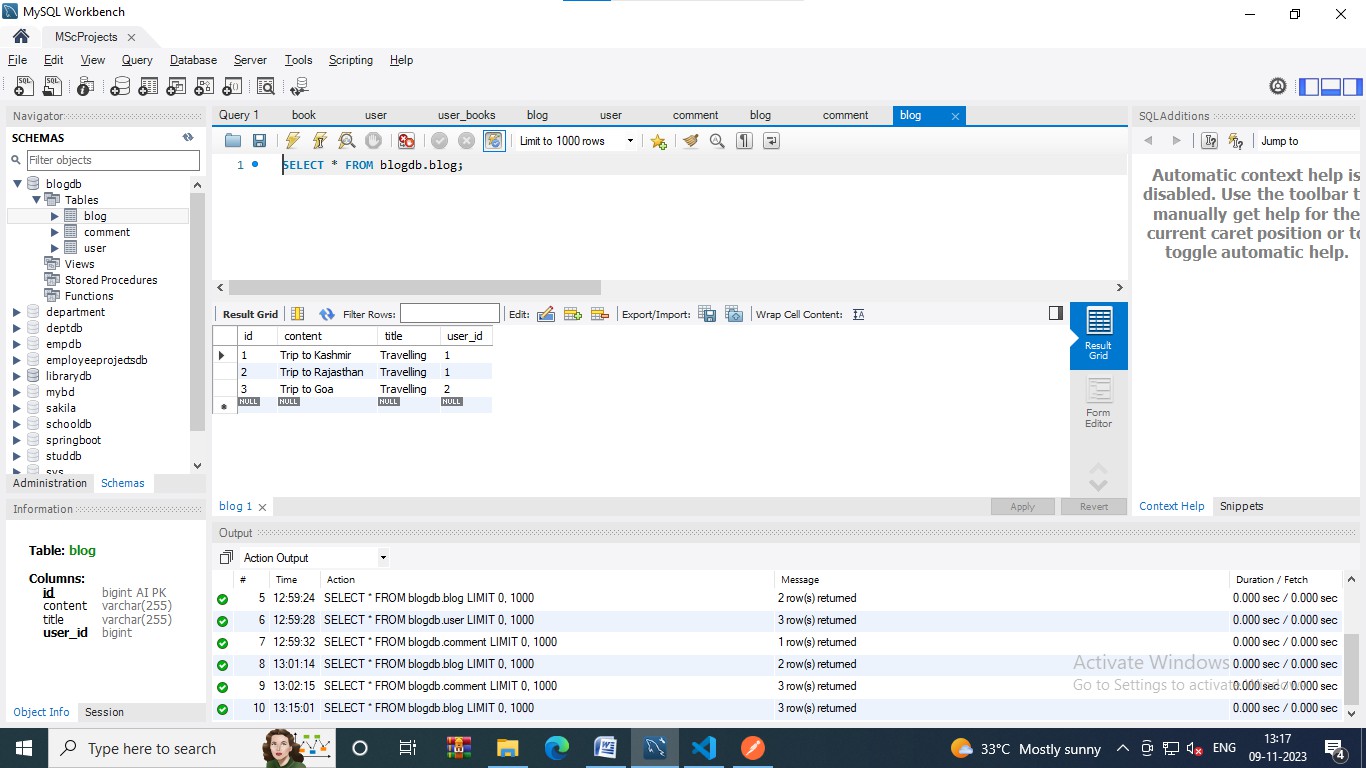
**//Update Blog**



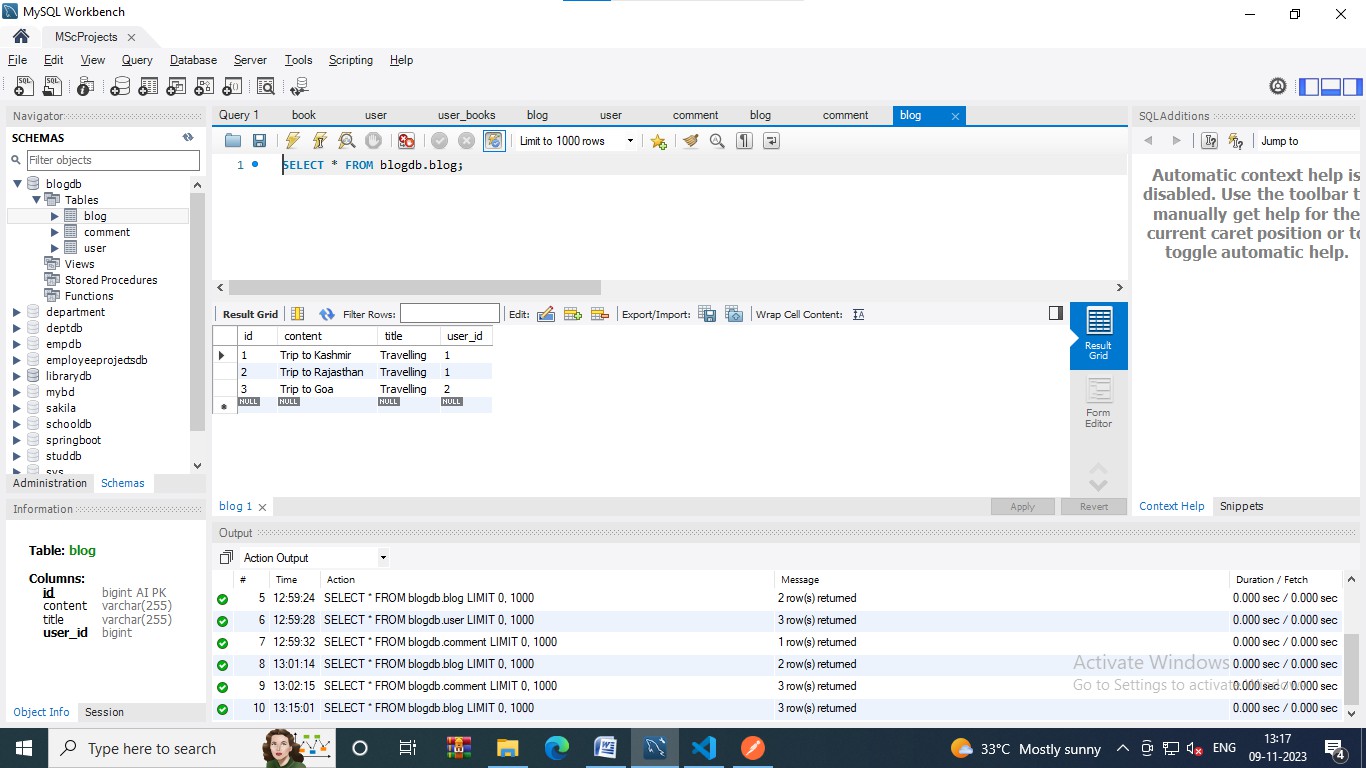
**//Delete a comment**



**//Auto-updating MYSQL database (Backend)**



**//comment**



**//user**

