A PROJECT REPORT ON

ONLINE BOOKSTORE APPLICATION

Submitted by

Mrs. Amritha S

Mrs. Sharmila R

Ms. Padmavathi K

Ms. Saraji Malathi

Ms. Sandhiya R T

Batch No. 7398

Under the Guidance of

Trainer Mrs. Indrakka Mali

INDEX

Introduction

The project “Online Book store application” is developed using spring boot framework. It is an online web application where the customer can purchase books online. Which is mainly focuses on web browser that customer can search for a book by its title or author. We used Inserting, Deleting, Updating and getting all information in this online book store application itself.

Author Module:

- In the Author module we can perform :
- We can add new records.
- Fetch customer record by author id.
- Fetch all customer records.
- Fetch all book records.
- Fetch all author records.
- Fetch all publisher records.

Books Module:

- In the Books Module we have book Id, Title, ISBN, Genre and Price.
- We can register books by using Book Id.
- Books can easily add, update, delete by using book service.

Customers Module:

- In Customers Module we have Customer Service.
- New customers can get registered using Customer Id.
- In customer details we have first name, last name, customer address and phone number and email id.

Orders Module:

- In orders module we get the orders by checking orderById and date of the Order we ordered.

Publishers Module:

- We get the publication details by PublisherId and PublicationName.

Objectives:

- It provides “better and efficient” service”.
- Faster way to get information about the Authors, Books, Customers, Orders and Publishers.
- Provide facility for proper monitoring and reduce paper work.
- All details will be available on a click.

System Overview:

- The Online Book Store Application will be automated the traditional system.
- There is no need to use paper and pen.
- Checking a student details is very easy.
- Adding new student record is very easy.
- Deleting or updating a record of a particular student is simple.

Requirements:

Software Requirement:

Database: MySQL

API- Spring Data JPA, spring web, spring security

Tools: Postman, IDE-Spring Tool Suit4

Coding language-Java 1.8

Hardware Requirements:

RAM: 2GB

Processor: 64bit

Memory: 512 MB

Disk Space: 100GB

Spring Tool Suit: STS is an Eclipse-based development environment that is customized for the development of spring applications.

It provides a ready-to-use environment to implement, debug, run and deploy your applications**.**

Postman: Postman is a standalone software testing API (Application Programming Interface) platform to build, test, design, modify, and document APIs. It is a simple Graphic User Interface for sending and viewing HTTP requests and responses.

MySQL:

- MySQL is a relational database management system
- MySQL is open-source
- MySQL is free
- MySQL is ideal for both small and large applications
- MySQL is very fast, reliable, scalable, and easy to use
- MySQL is cross-platform

Coding parts

Entity

Authors

```
package com.entity;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.NotBlank;

@Entity

public class Authors {

        @Id

        @GeneratedValue(strategy = GenerationType.IDENTITY)

        private Integer authorId;

        @NotBlank(message = "First name should not be blank")

        private String firstName;
```

```
	@NotBlank(message = "Last name should not be blank")
	private String lastName;
	//no argument constructor
	public Authors() {
	super();
	}
	//getters and setters
	public Integer getAuthorId() {
	return authorId;
	}
	public void setAuthorId(Integer authorId) {
		this.authorId = authorId;
	}
	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;
	}
}
```

Entity

Books

```
package com.entity;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.SequenceGenerator;

import javax.validation.constraints.NotBlank;

import org.hibernate.validator.constraints.Length;

@Entity

@SequenceGenerator(name = "bookseq" , initialValue = 2001)

public class Books {

        @Id

        @GeneratedValue(strategy = GenerationType.SEQUENCE , generator = "bookseq")

        private Integer bookId;

        @NotBlank(message = "Book title should not be blank")

        private String bookTitle;

        @Column(unique = true)
```

```
@NotBlank(message = "ISBN should not be blank")
@Length(min = 13 , max = 18 , message = "ISBN should have 13 digits" )
private String bookISBN;

@NotBlank(message = "Type of the book should not be blank")
private String bookGenre;
private Integer publicationYear;
private Float bookPrice;
@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "author_id" , referencedColumnName = "authorId")
private Authors authors;
@ManyToOne(cascade = CascadeType.ALL)
@JoinColumn(name = "publisher_id" , referencedColumnName = "publisherId")
private Publishers publishers;
//no argument constructor
public Books() {
	super();
}
//getters and setters
public Integer getBookId() {
	return bookId;
}
public void setBookId(Integer bookId) {
	this.bookId = bookId;
}
public String getBookTitle() {
	return bookTitle;
```

```
}
public void setBookTitle(String bookTitle) {
	this.bookTitle = bookTitle;
}
public String getBookISBN() {
	return bookISBN;
}
public void setBookISBN(String bookISBN) {
	this.bookISBN = bookISBN;
}
public String getBookGenre() {
	return bookGenre;
}
public void setBookGenre(String bookGenre) {
	this.bookGenre = bookGenre;
}
public Integer getPublicationYear() {
	return publicationYear;
}
public void setPublicationYear(Integer publicationYear) {
	this.publicationYear = publicationYear;
}
public Float getBookPrice() {
	return bookPrice;
}
public void setBookPrice(Float bookPrice) {
	this.bookPrice = bookPrice;
```

```
	}
	public Authors getAuthors() {
		return authors;
	}
	public void setAuthors(Authors authors) {
		this.authors = authors;
	}
	public void bookAuthor(Authors authors) {
		this.authors = authors;
	}
	public void bookPublisher(Publishers publishers) {
		this.publishers = publishers;
	}
	public Publishers getPublishers() {
		return publishers;
	}
	public void setPublishers(Publishers publishers) {
		this.publishers = publishers;
	}
}
```

Entity

Customers

```
package com.entity;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
```

```
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import org.hibernate.validator.constraints.Length;
@Entity
@SequenceGenerator(name = "custseq" , initialValue = 10001)
public class Customers {
	@Id
	@GeneratedValue(strategy = GenerationType.SEQUENCE , generator = "custseq")
	private Integer customerId;
	@NotBlank(message = "First name should not be blank")
	@Length(max = 30)
	private String firstName;
	@NotBlank(message = "Last name should not be blank")
	@Length(max = 30)
	private String lastName;
	@NotBlank(message = "Communication address should not be blank")
	private String address;
	@NotBlank(message = "Phone number should not be blank")
	@Length(min = 10 , max = 10 , message = "Phone number should have 10 digits")
	private String phoneNumber;
	@Column(unique = true)
	@NotBlank(message = "Email id should not be blank")
	@Email(message = "Enter valid email id")
	private String emailId;
```

```
//no argument constructor
public Customers() {
	super();
}
//getters and setters
public Integer getCustomerId() {
	return customerId;
}
public void setCustomerId(Integer customerId) {
	this.customerId = customerId;
}
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 getAddress() {
	return address;
}
public void setAddress(String address) {
```

```
		this.address = address;
	}
	public String getPhoneNumber() {
		return phoneNumber;
	}
	public void setPhoneNumber(String phoneNumber) {
		this.phoneNumber = phoneNumber;
	}
	public String getEmailId() {
		return emailId;
	}
	public void setEmailId(String emailId) {
		this.emailId = emailId;
	}
}
```

Entity

Orders

```
package com.entity;
import java.time.LocalDate;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
```

```
import javax.persistence.ManyToMany;
import javax.persistence.OneToOne;
import javax.persistence.SequenceGenerator;
import com.fasterxml.jackson.annotation.JsonFormat;
import com.request.OrdersRequest;

@Entity
@SequenceGenerator(name = "orderseq" , initialValue = 50001)
public class Orders {
@Id
@GeneratedValue(strategy = GenerationType.SEQUENCE , generator = "orderseq")
private Integer orderId;
private Integer quantity;
@JsonFormat(pattern = "yyyy-MM-dd")
private LocalDate orderDate;
private float totalPrice;
@ManyToMany
@JoinTable(name = "book_id" , joinColumns = @JoinColumn(name = "order_id"),
inverseJoinColumns = @JoinColumn(name = "book_id"))
private Set<Books> purchasedBooks = new HashSet<>();
@OneToOne
@JoinColumn(name = "customerId")
private Customers customer;
//Default constructor
public Orders() {
super();
}
//getters and setters
public Integer getOrderId() {
return orderId;
}
public void setOrderId(Integer orderId) {
```

```
this.orderId = orderId;
}
public Integer getQuantity() {
return quantity;
}
public void setQuantity(Integer quantity) {
this.quantity = quantity;
}
public LocalDate getOrderDate() {
return orderDate;
}
public void setOrderDate(LocalDate orderDate) {
this.orderDate = orderDate;
}
public float getTotalPrice() {
return totalPrice;
}
public void setTotalPrice(float totalPrice) {
this.totalPrice = totalPrice;
}
public Set<Books> getPurchasedBooks() {
return purchasedBooks;
}
public void setPurchasedBooks(Set<Books> purchasedBooks) {
this.purchasedBooks = purchasedBooks;
}
public void orderedBook(Books book) {
purchasedBooks.add(book);

}
public Customers getCustomer() {
return customer;
```

```
}
public void setCustomer(Customers customer) {
this.customer = customer;
}
public Orders(Integer orderId, Integer quantity, LocalDate orderDate, float totalPrice) {
super();
this.orderId = orderId;
this.quantity = quantity;
this.orderDate = orderDate;
this.totalPrice = totalPrice;
}
@Override
public String toString() {
return "Orders [orderId=" + orderId + ", quantity=" + quantity + ", orderDate=" + orderDate +
", totalPrice="
                + totalPrice + "]";
}
public Orders(OrdersRequest request) {
super();
this.orderId = orderId;
this.quantity = quantity;
this.orderDate = orderDate;
this.totalPrice = totalPrice;
}
```

Entity

Publishers

```
package com.entity;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.validation.constraints.NotBlank;
@Entity
@SequenceGenerator(name = "publicseq" , initialValue = 1001)
public class Publishers {
@Id
@GeneratedValue(strategy = GenerationType.SEQUENCE , generator = "publicseq")
private Integer publisherId;
@NotBlank(message = "Publication name should not be blank")
private String publicationName;
//no argument constructor
public Publishers() {
super();
}
//getters and setters
public Integer getPublisherId() {
return publisherId;
}
public void setPublisherId(Integer publisherId) {
this.publisherId = publisherId;
}
public String getPublicationName() {
return publicationName;
}
public void setPublicationName(String publicationName) {
this.publicationName = publicationName;
}
}
}
```

Controller

Authors Controller

```
package com.entity;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;
import javax.validation.constraints.NotBlank;
@Entity
@SequenceGenerator(name = "publicseq" , initialValue = 1001)
public class Publishers {
@Id
@GeneratedValue(strategy = GenerationType.SEQUENCE , generator = "publicseq")
private Integer publisherId;
@NotBlank(message = "Publication name should not be blank")
private String publicationName;
//no argument constructor
public Publishers() {
super();
}
//getters and setters
public Integer getPublisherId() {
return publisherId;
}
public void setPublisherId(Integer publisherId) {
this.publisherId = publisherId;
}
public String getPublicationName() {
return publicationName;
}
public void setPublicationName(String publicationName) {
this.publicationName = publicationName;
```

```
}
}
```

Books Controller

```
package com.controller;
import java.util.List;
import javax.validation.Valid;
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.RestController;
import com.entity.Books;
import com.service.BooksService;
@RestController
public class BooksController {
@Autowired
private BooksService booksService;
@GetMapping("/getAllBooks")
public List<Books> getAllBooks()
{
return booksService.getAllBooks();
}
@PostMapping("/registerBook")
```

```
public ResponseEntity<Books> registerBook(@Valid @RequestBody Books books)
{
return new ResponseEntity<Books>(booksService.registerBook(books),
HttpStatus.CREATED);
}

@PutMapping("/updateBookById/{bookid}")
public ResponseEntity<Books> updateBook(@PathVariable("bookid") Integer bookid,
@RequestBody Books book)
{
return new ResponseEntity<Books>(booksService.updateBook(bookid,book),HttpStatus.OK);
}
@DeleteMapping("/deleteBookById/{bookid}")
public ResponseEntity<String> deleteBookById(@PathVariable("bookid") Integer bookid)
{
booksService.deleteBookById(bookid);
return new ResponseEntity<String>("Book record is deleted",HttpStatus.OK);
}
@PutMapping("/book/{bookid}/author/{authorid}")
public ResponseEntity<Books> updateAuthorToBook(@PathVariable Integer bookid ,
@PathVariable Integer authorid)
{
return new ResponseEntity<Books>(booksService.updateAuthorToBook(bookid,authorid),
HttpStatus.OK);
}
@PutMapping("/book/{bookid}/publisher/{publisherid}")
public ResponseEntity<Books> updatePublisherToBook(@PathVariable Integer bookid ,
@PathVariable Integer publisherid)
{
return new ResponseEntity<Books>(booksService.updatePublisherToBook(bookid,
publisherid) , HttpStatus.OK);
}
}
```

Customers Controller

```
package com.controller;
import java.util.List;
import javax.validation.Valid;
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.RestController;
import com.entity.Customers;
import com.service.CustomersService;
@RestController
public class CustomerController {
@Autowired
private CustomersService customersService;
@GetMapping("/getAllCustomers")
public List<Customers> getAllCustomers()
{
return customersService.getAllCustomers();
}
@PostMapping("/registerCustomer")
public ResponseEntity<Customers> registerCustomer(@Valid @RequestBody Customers
customers)
{
return new ResponseEntity<Customers>(customersService.registerCustomer(customers) ,
HttpStatus.CREATED);
}
```

```
@PutMapping("/updateCustomerById/{customerid}")
public ResponseEntity<Customers> updateCustomer(@PathVariable("customerid") Integer customerid , @RequestBody Customers customer)
{
return new ResponseEntity<Customers>(customersService.updateCustomer(customerid,customer), HttpStatus.OK);
}
@DeleteMapping("/deleteCustomerById/{customerid}")
public ResponseEntity<String> deleteCustomerById(@PathVariable("customerid") Integer customerid)
{
customersService.deleteCustomerById(customerid);
return new ResponseEntity<String>("Customer details deleted" , HttpStatus.OK);
}
}
```

Orders Controller

```
package com.controller;
import java.util.List;
import javax.validation.Valid;
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.RestController;
import com.entity.Customers;
import com.entity.Orders;
```

```
import com.repository.CustomersRepository;
import com.repository.OrdersRepository;
import com.request.OrdersRequest;
import com.service.OrdersService;

@RestController
public class OrdersController {
@Autowired
private OrdersRepository ordersRepository;
@Autowired
private CustomersRepository customersRepository;
@Autowired
private OrdersService ordersService;
@GetMapping("/getAllOrders")
public List<Orders> getAllOrders()
{
return ordersService.getAllOrders();
}
@PutMapping("/updateOrderById/{orderid}")
public ResponseEntity<Orders> updateOrderDetails(@PathVariable("orderid") Integer
orderid, @RequestBody Orders order)
{
return new ResponseEntity<Orders>(ordersService.updateOrderDetails(orderid,order) ,
HttpStatus.OK);
}
@DeleteMapping("/deleteOrderById/{orderid}")
public ResponseEntity<String> deleteOrderById(@PathVariable("orderid") Integer orderid)
{
ordersService.deleteOrderById(orderid);
return new ResponseEntity<String>("Your Order is cancelled successfully!" , HttpStatus.OK);
}
@PutMapping("/order/{orderid}/book/{bookid}")
```

```
public ResponseEntity<Orders> addBookToOrderList(@PathVariable Integer orderid ,
@PathVariable Integer bookid)
{
return new ResponseEntity<Orders>(ordersService.addBookToOrderList(orderid,bookid) ,
HttpStatus.OK);
}
@PostMapping("/placeNewOrder")
public ResponseEntity<Orders> addCustomerToOrder(@Valid @RequestBody OrdersRequest
request)
{
Customers customer = new Customers();
customer.setFirstName(request.getFirstName());
customer.setLastName(request.getLastName());
customer.setPhoneNumber(request.getPhoneNumber());
customer.setEmailId(request.getEmailId());
customer.setAddress(request.getAddress());
customer= customersRepository.save(customer);
Orders order = new Orders(request);
order.setCustomer(customer);
order= ordersRepository.save(order);
return new ResponseEntity<Orders>(order , HttpStatus.OK);
}
}
```

Publishers Controller

```
package com.controller;
import java.util.List;
import javax.validation.Valid;
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.RestController;
import com.entity.Publishers;
import com.service.PublishersService;
@RestController
public class PublishersController {
@Autowired
private PublishersService publishersService;
@GetMapping("/getAllPublishers")
public List<Publishers> getAllPublishers()
{
return publishersService.getAllPublishers();
}
@PostMapping("/registerPublisher")
public ResponseEntity<Publishers> registerPublisher(@Valid @RequestBody Publishers
publishers)
{
return new ResponseEntity<Publishers>(publishersService.registerPublisher(publishers),
HttpStatus.CREATED);
}
@PutMapping("/updatePublisherById/{publisherid}")
public ResponseEntity<Publishers> updatePublisherById(@PathVariable("publisherid")
Integer publisherid , @RequestBody Publishers publisher)
{
return new ResponseEntity<Publishers>(publishersService.updatePublisherById(publisherid,
publisher) , HttpStatus.OK);
}
@DeleteMapping("/deletePublisherById/{publisherid}")
public ResponseEntity<String> deletePublisherById(@PathVariable("publisherid") Integer
publisherid)
```

```
{
publishersService.deletePublisherById(publisherid);
return new ResponseEntity<String>("Publisher record is deleted",HttpStatus.OK);
}}
```

Repository

Authors Repository

```
package com.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.entity.Authors;
@Repository
public interface AuthorsRepository extends JpaRepository<Authors, Integer> {
}
```

Books Repository

```
package com.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.entity.Books;
@Repository
public interface BooksRepository extends JpaRepository<Books, Integer> {
}
```

Customers Repository

```
package com.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.entity.Customers;
@Repository
public interface CustomersRepository extends JpaRepository<Customers, Integer>{
}
```

Orders Repository

```
package com.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.entity.Orders;
@Repository
public interface OrdersRepository extends JpaRepository<Orders, Integer>{
}
```

Publishers Repository

```
package com.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.entity.Publishers;
@Repository
public interface PublishersRepository extends JpaRepository<Publishers, Integer> {
}
```

Exception Handling

Global Exception Handler

```
package com.exception;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.context.request.WebRequest;
```

```
import
org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;
//Global exception handling
@ControllerAdvice
public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {
@Override
protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
                    HttpHeaders headers, HttpStatus status, WebRequest request) {
          Map<String,Object> body=new LinkedHashMap<>();
   body.put("timestamp", System.currentTimeMillis());
   body.put("status", status.value());
 //get all the errors
 List<String> errors=ex.getBindingResult()
                                          .getFieldErrors()
                                          .stream()
                                          .map(x->x.getDefaultMessage())
                                          .collect(Collectors.toList());
                    body.put("errors", errors);
                    return new ResponseEntity<Object>(body,status);
  }
}
```

ResourceNotFoundException

```
package com.exception;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
//Customexception
@ResponseStatus(value=HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException {
private String resourceName;
private String fieldName;
```

```
private Object fieldvalue;
private static final long serialVersionUID = 1L;
public ResourceNotFoundException(String resourceName, String fieldName, Integer bookid) {
super(String.format("%s not found with %s:'%s'",resourceName,fieldName,bookid));
this.resourceName = resourceName;
this.fieldName = fieldName;
this.fieldvalue = bookid;
}
//getterMethods
public String getResourceName() {
return resourceName;
}
public String getFieldName() {
return fieldName;
}
public Object getFieldvalue() {
return fieldvalue;
}}
```

OrdersRequest

```
package com.request;
import java.time.LocalDate;
public class OrdersRequest {
private Integer quantity;
private LocalDate orderDate;
private float totalPrice;
private String firstName;
private String lastName;
private String address;
private String phoneNumber;
private String emailId;
//Default Constructor
```

```
public OrdersRequest() {
super();
}
//getters and setters
public Integer getQuantity() {
return quantity;
}
public void setQuantity(Integer quantity) {
this.quantity = quantity;
}
public LocalDate getOrderDate() {
return orderDate;
}
public void setOrderDate(LocalDate orderDate) {
this.orderDate = orderDate;
}
public float getTotalPrice() {
return totalPrice;
}
public void setTotalPrice(float totalPrice) {
this.totalPrice = totalPrice;
}
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 getAddress() {
return address;
}
public void setAddress(String address) {
this.address = address;
}
public String getPhoneNumber() {
return phoneNumber;
}
public void setPhoneNumber(String phoneNumber) {
this.phoneNumber = phoneNumber;
}
public String getEmailId() {
return emailId;
}
public void setEmailId(String emailId) {
this.emailId = emailId;
}
@Override
public String toString() {
return "OrdersRequest [quantity=" + quantity + ", orderDate=" + orderDate
                + ", totalPrice=" + totalPrice + ", firstName=" + firstName
                + ", lastName=" + lastName + ", address=" + address + ", phoneNumber=" +
phoneNumber + ", emailId="
                + emailId + "]";
}
}
```

Service

Customer Service

```
package com.service;
```

```
import java.util.List;
import com.entity.Customers;
public interface CustomersService {
List<Customers> getAllCustomers();
Customers registerCustomer(Customers customers);
Customers updateCustomer(Integer customerid, Customers customer2);
void deleteCustomerById(Integer customerid);
}
```

Customer Service Impl

```
package com.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.entity.Customers;
import com.exception.ResourceNotFoundException;
import com.repository.CustomersRepository;
@Service
public class CustomersServiceImpl implements CustomersService {@Autowired
private CustomersRepository customersRepository;
@Override
public List<Customers> getAllCustomers() {
return customersRepository.findAll();
}
@Override
public Customers registerCustomer(Customers customers) {
return customersRepository.save(customers);
}
@Override
public Customers updateCustomer(Integer customerid, Customers customer) {
Customers customer1 = customersRepository.findById(customerid).get();
Customers customerdb= null;
if(customer1 != null)
```

```
{
        customerdb = customersRepository.findById(customerid).get();
        customerdb.setFirstName(customer.getFirstName());
        customerdb.setLastName(customer.getLastName());
        customerdb.setAddress(customer.getAddress());
        customerdb.setPhoneNumber(customer.getPhoneNumber());
        customerdb.setEmailId(customer.getEmailId());
        System.out.println(customerdb);
        return customersRepository.save(customerdb);
}
else
{
        throw new ResourceNotFoundException("Customer", "customerid", customerid);
}
}
@Override
public void deleteCustomerById(Integer customerid) {
Customers customer = customersRepository.findById(customerid).get();
if(customer != null)
{
        customersRepository.deleteById(customerid);
}
else
{
        throw new ResourceNotFoundException("Customer", "customerid" , customerid);
}
}
}
```

Order Service

```
package com.service;
import java.util.List;
import com.entity.Orders;
public interface OrdersService {
List<Orders> getAllOrders();
Orders placeNewOrder(Orders orders);
Orders updateOrderDetails(Integer orderid, Orders order);
void deleteOrderById(Integer orderid);
Orders addBookToOrderList(Integer orderid, Integer bookid);
}
```

Order Service Impl

```
package com.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.entity.Books;
import com.entity.Orders;
import com.exception.ResourceNotFoundException;
import com.repository.BooksRepository;
import com.repository.OrdersRepository;

@Service
public class OrdersServiceImpl implements OrdersService {
@Autowired
private OrdersRepository ordersRepository;

@Autowired
private BooksRepository booksRepository;
@Override
public List<Orders> getAllOrders() {
return ordersRepository.findAll();
```

```
}
@Override
public Orders placeNewOrder(Orders orders) {
return ordersRepository.save(orders);
}
@Override
public Orders updateOrderDetails(Integer orderid, Orders order) {
Orders orders = ordersRepository.findById(orderid).get();
Orders orderdb = null;
if(orders != null)
{
	orderdb = ordersRepository.findById(orderid).get();

	orderdb.setQuantity(order.getQuantity());
	orderdb.setOrderDate(order.getOrderDate());
	orderdb.setTotalPrice(order.getTotalPrice());
	System.out.println(orderdb);
	return ordersRepository.save(orderdb);
}
else
{
	throw new ResourceNotFoundException("Orders", "Orderid", orderid);
}
}
@Override
public void deleteOrderById(Integer orderid) {
Orders orders = ordersRepository.findById(orderid).get();
if(orders != null)
{
	ordersRepository.deleteById(orderid);
}
else
```

```
{
        throw new ResourceNotFoundException("Orders", "Orderid", orderid);
}
}
@Override
public Orders addBookToOrderList(Integer orderid, Integer bookid) {
Orders order = ordersRepository.findById(orderid).get();
Books book = booksRepository.findById(bookid).get();
if(order != null && book != null)
{
        order.orderedBook(book);
        return ordersRepository.save(order);
}
else
{
        throw new ResourceNotFoundException("Orders", "Orderid", orderid);
}
}
}
```

Book Service

```
package com.service;
import java.util.List;
import com.entity.Books;
public interface BooksService {
List<Books> getAllBooks();
Books registerBook(Books books);
Books updateAuthorToBook(Integer bookid, Integer authorid);
Books updatePublisherToBook(Integer bookid, Integer publisherid);
void deleteBookById(Integer bookid);
Books updateBook(Integer bookid, Books book);
}
```

Book Service Impl

```
package com.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.entity.Authors;
import com.entity.Books;
import com.entity.Publishers;
import com.exception.ResourceNotFoundException;
import com.repository.AuthorsRepository;
import com.repository.BooksRepository;
import com.repository.PublishersRepository;

@Service
public class BooksServiceImpl implements BooksService {
@Autowired
	private BooksRepository booksRepository;
@Autowired
	private AuthorsRepository authorsRepository;
@Autowired
	private PublishersRepository publishersRepository;
@Override
	public List<Books> getAllBooks() {
	return booksRepository.findAll();
	}
@Override
	public Books registerBook(Books books) {
	return booksRepository.save(books);
	}
@Override
	public Books updateAuthorToBook(Integer bookid, Integer authorid) {
		Books books = booksRepository.findById(bookid).get();
```

```
		Authors authors = authorsRepository.findById(authorid).get();
		if(books != null && authors != null)
		{
		books.bookAuthor(authors);
		return booksRepository.save(books);
		}
		else
		{
			throw new ResourceNotFoundException("Books", "bookid", bookid);
		}
	}
@Override
	public Books updatePublisherToBook(Integer bookid, Integer publisherid) {
		Books books = booksRepository.findById(bookid).get();
		Publishers publishers = publishersRepository.findById(publisherid).get();
		if(books != null && publishers != null)
		{
		books.bookPublisher(publishers);
		return booksRepository.save(books);
		}
		else
		{
			throw new ResourceNotFoundException("Books", "bookid", bookid);
		}
	}
	@Override
	public void deleteBookById(Integer bookid) {
		Books books = booksRepository.findById(bookid).get();
		if(books != null)
			booksRepository.deleteById(bookid);
		else
			throw new ResourceNotFoundException("Books", "bookid", bookid);
```

```
	}
@Override
	public Books updateBook(Integer bookid, Books book) {
		Books book1 = booksRepository.findById(bookid).get();
		Books bookdb = null;
		if(book1 != null)
		{
		bookdb = booksRepository.findById(bookid).get();
		bookdb.setBookTitle(book.getBookTitle());
		bookdb.setBookISBN(book.getBookISBN());
		bookdb.setBookGenre(book.getBookGenre());
		bookdb.setPublicationYear(book.getPublicationYear());
		bookdb.setBookPrice(book.getBookPrice());
		System.out.println(bookdb);
		return booksRepository.save(bookdb);
		}
		else
		{
		throw new ResourceNotFoundException("Books", "bookid", bookid);
		}
	}
}
```

Publishers Service

```
package com.service;
import java.util.List;
import com.entity.Publishers;
public interface PublishersService {
List<Publishers> getAllPublishers();
Publishers registerPublisher(Publishers publishers);
void deletePublisherById(Integer publisherid);
Publishers updatePublisherById(Integer publisherid, Publishers publisher);
}
```

Publishers Service Impl

```
package com.service;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.entity.Publishers;
import com.exception.ResourceNotFoundException;
import com.repository.PublishersRepository;
@Service
public class PublishersServiceImpl implements PublishersService{
@Autowired
private PublishersRepository publishersRepository;
@Override
public List<Publishers> getAllPublishers() {
return publishersRepository.findAll();
}
@Override
public Publishers registerPublisher(Publishers publishers) {
return publishersRepository.save(publishers);
}
@Override
public void deletePublisherById(Integer publisherid) {
Publishers publishers = publishersRepository.findById(publisherid).get();
if(publishers != null)
{
	publishersRepository.deleteById(publisherid);
}
else
{
	throw new ResourceNotFoundException("Publisher", "publisherid", publisherid);
}
```

```
}
@Override
public Publishers updatePublisherById(Integer publisherid, Publishers publisher) {
Publishers publishers = publishersRepository.findById(publisherid).get();
Publishers publisherdb= null;
if(publishers != null)
{
        publisherdb = publishersRepository.findById(publisherid).get();
        publisherdb.setPublicationName(publisher.getPublicationName());
        System.out.println(publisherdb);
        return publishersRepository.save(publisherdb);
}
else
{
        throw new ResourceNotFoundException("Publisher", "publisherid", publisherid);
}
}
}
```

OnlineBookStoreAppProjectApplication

```
package com;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
public class OnlineBookStoreAppProjectApplication {
public static void main(String[] args) {
SpringApplication.run(OnlineBookStoreAppProjectApplication.class, args);
}
}
```

OnlineBookStoreAppProjectApplicationTest

```
package com;
import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
```

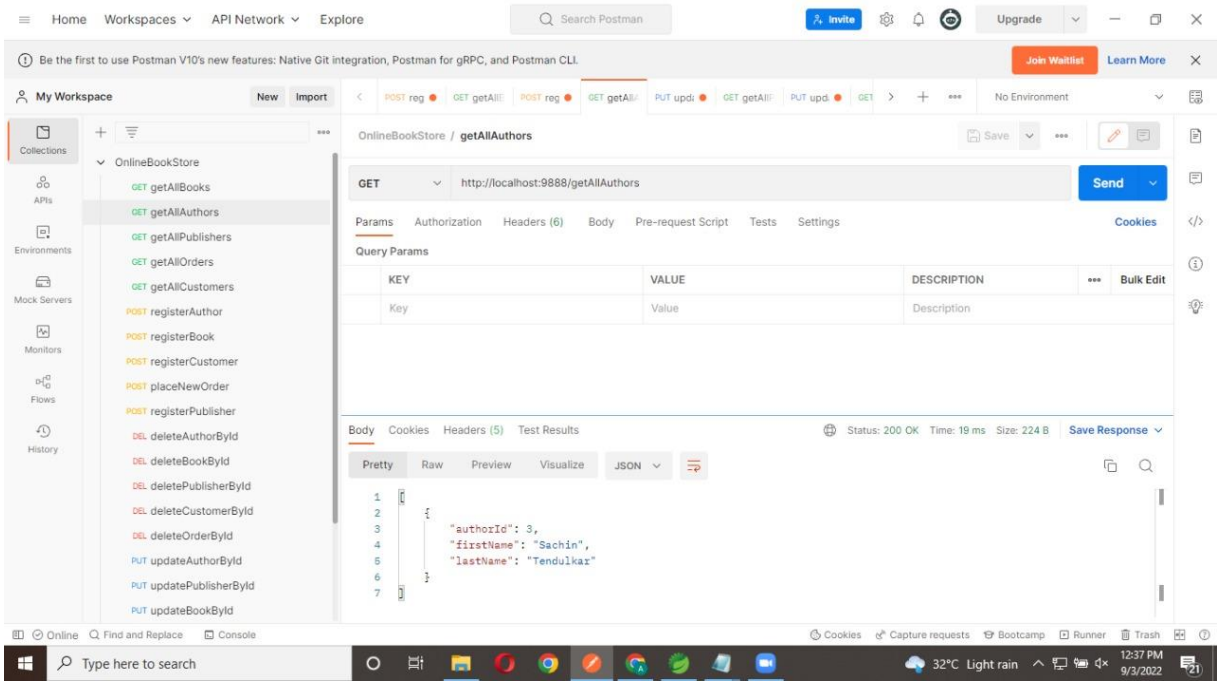
```
@SpringBootTest
class OnlineBookStoreAppProjectApplicationTests {
@Test
void contextLoads() {
}}
```

Screenshot

Get

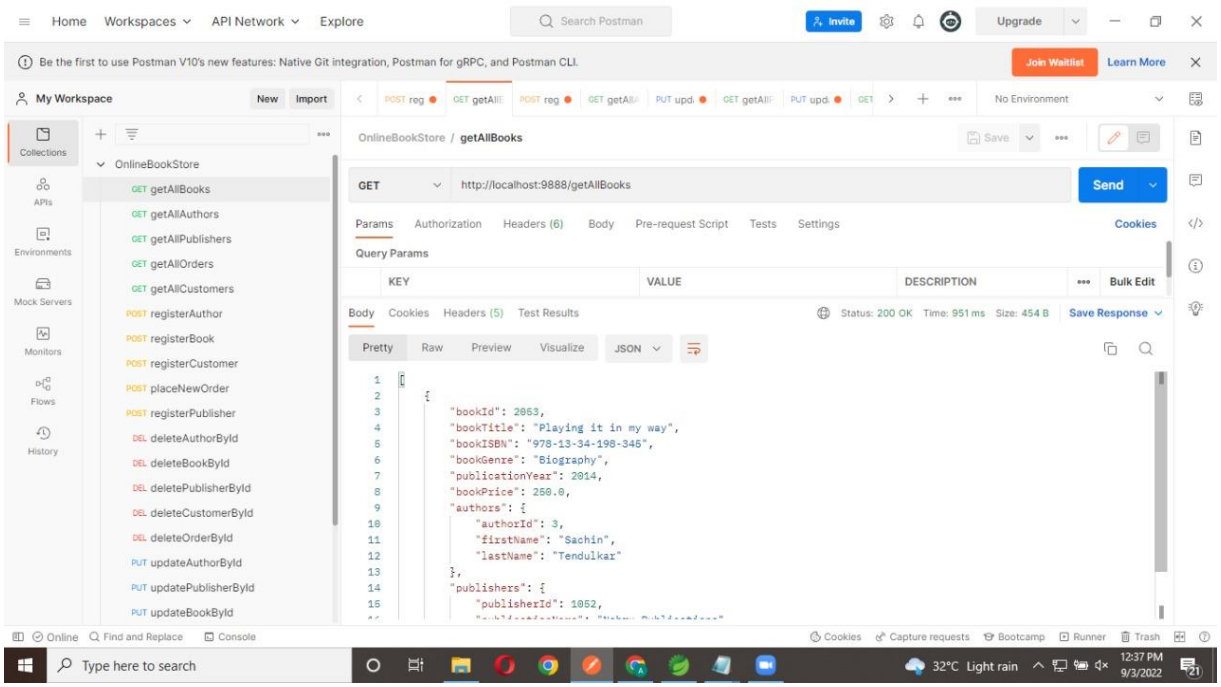
Step 1: We can get all the author details by using author Id.

URL: http://localhost:9888/getAllAuthors

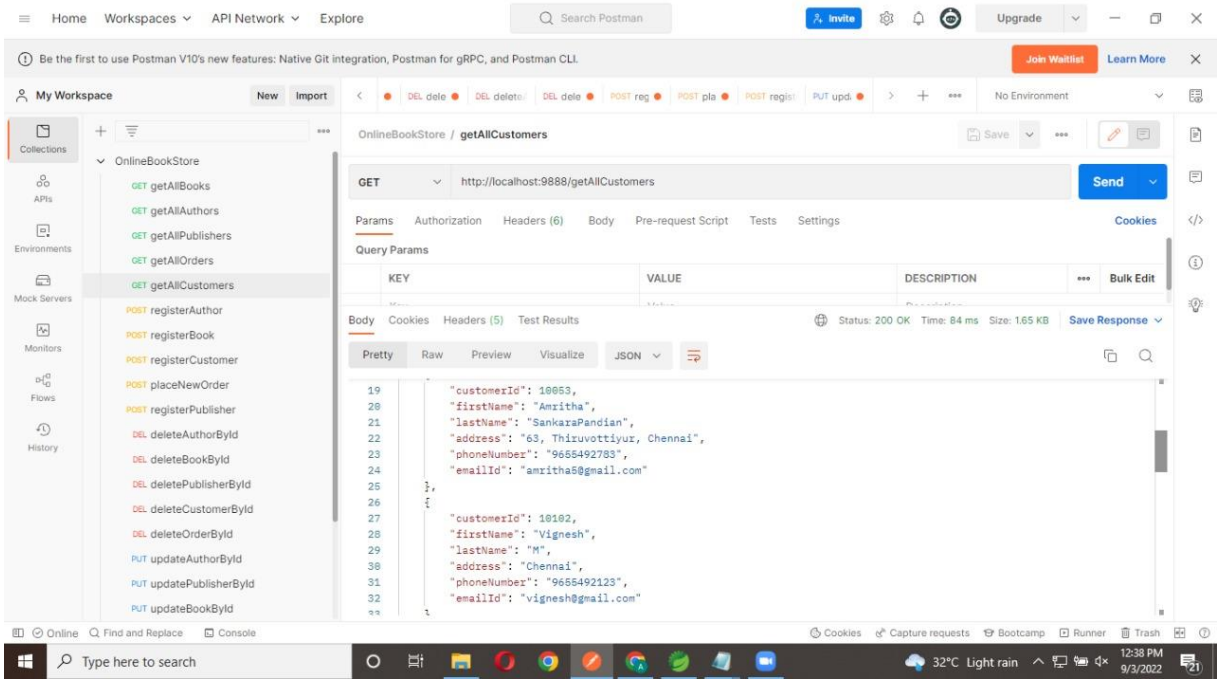


Step 2: We can get all the books by using book Id.

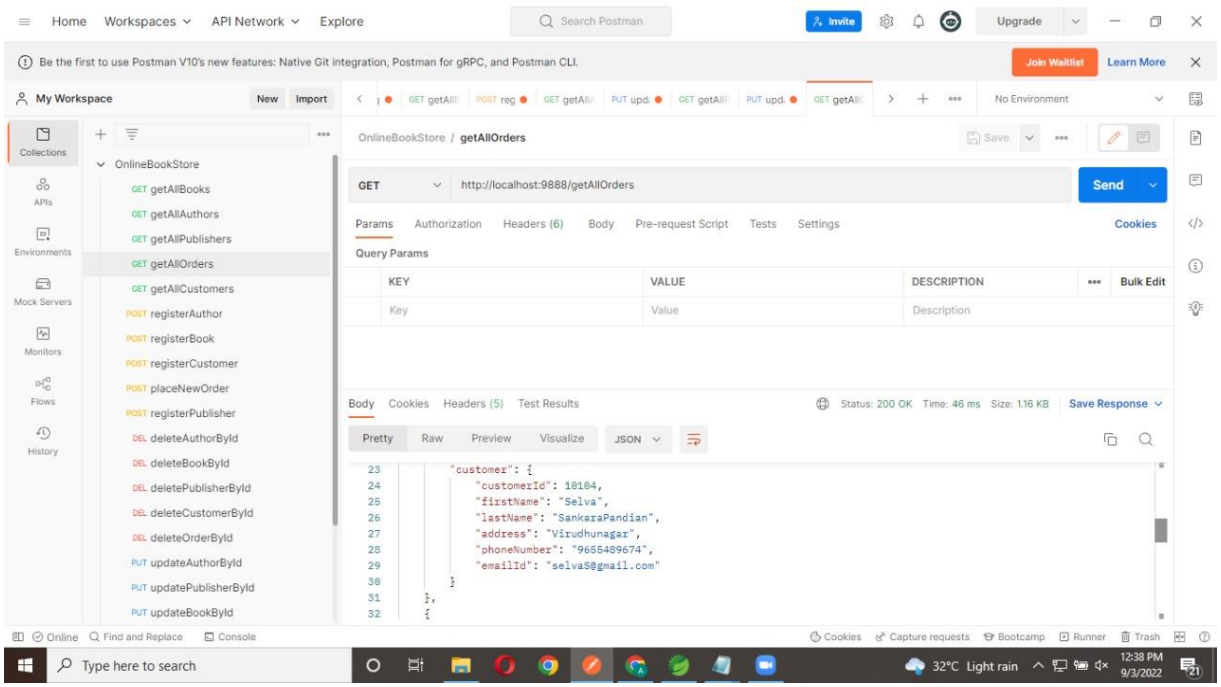
URL: http://localhost:9888/getAllBooks



Step 3: We can get all the customer details by using Customer Id.
URL: http://localhost:9888//getAllCustomers

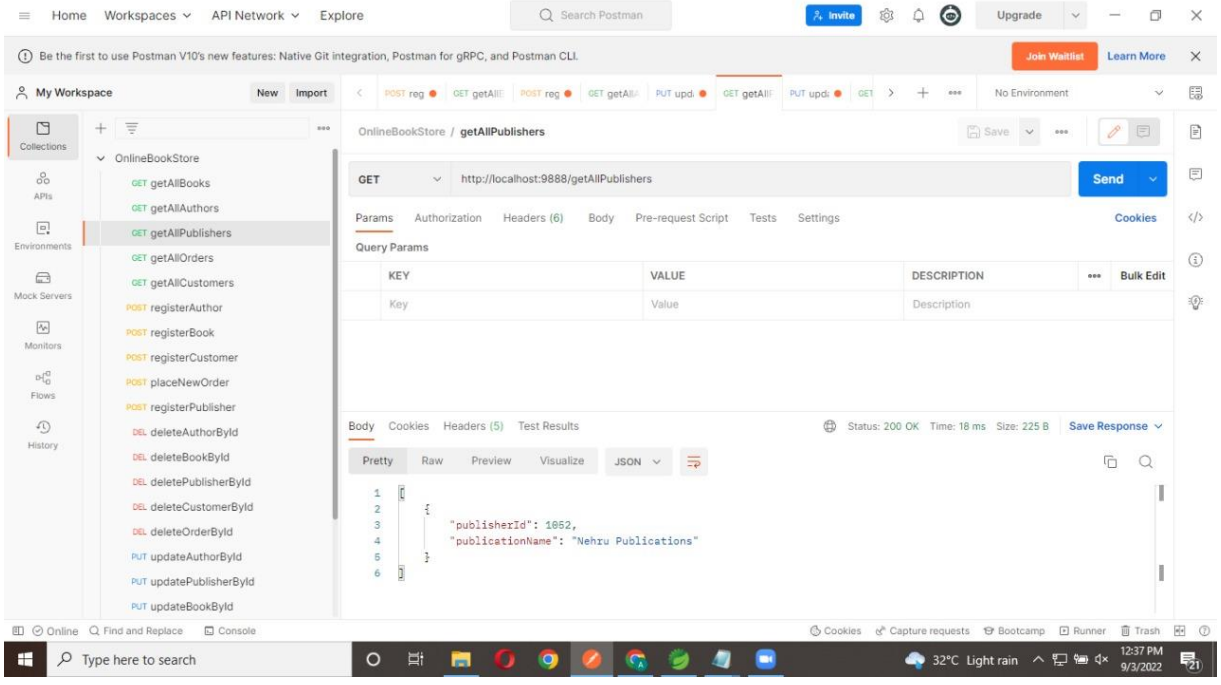


Step 4: We can get all the orders using customer Id.
URL: http://localhost:9888//getAllOrders



Step 5: We get the publication details using publisher Id.

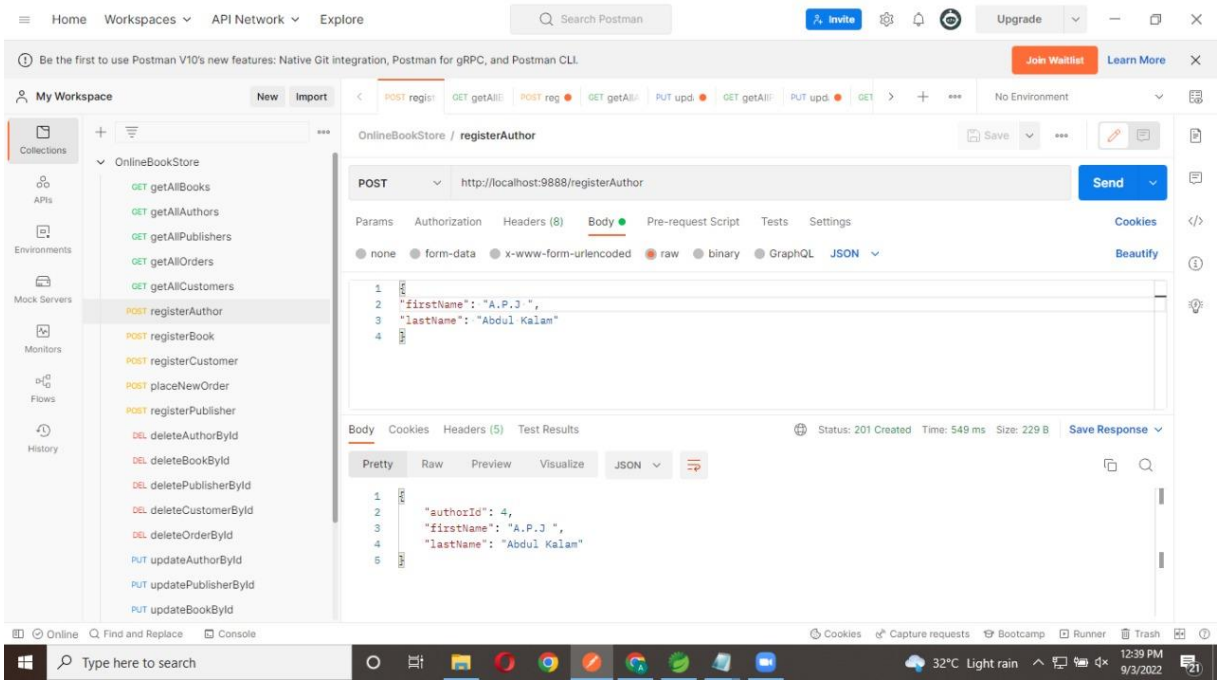
URL: http://localhost:9888//getAllPublishers



Post

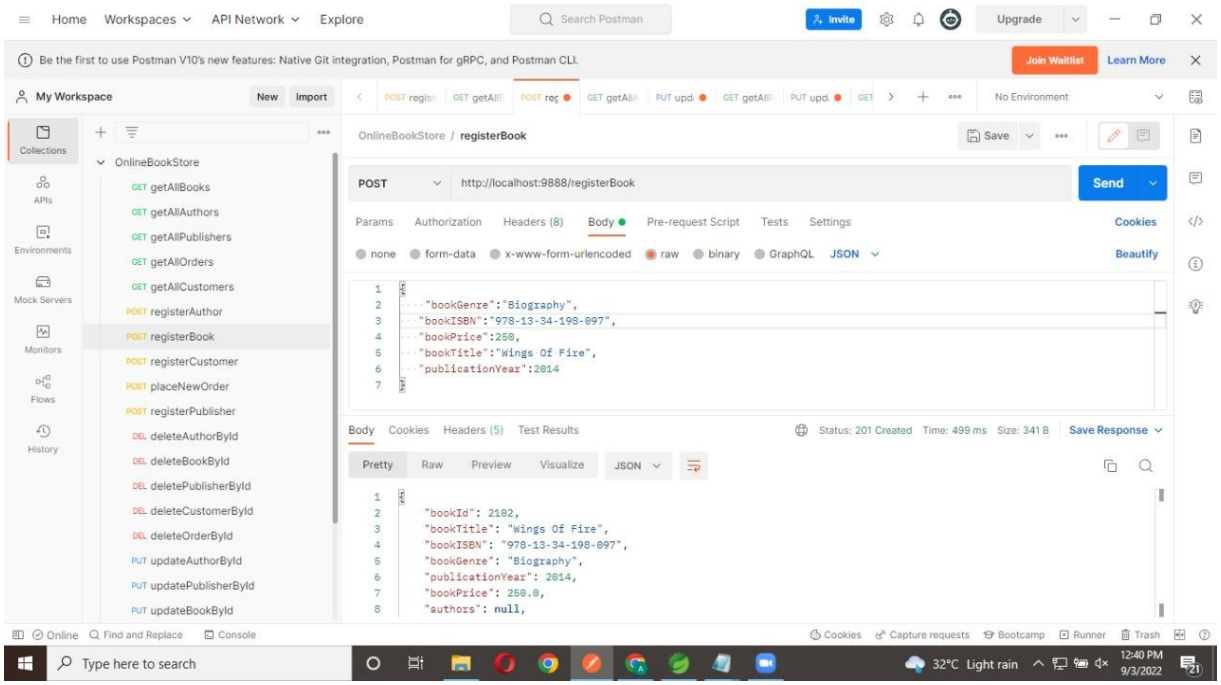
Step 6: Author can register using author Id.

URL: http://localhost:9888//registerAuthor



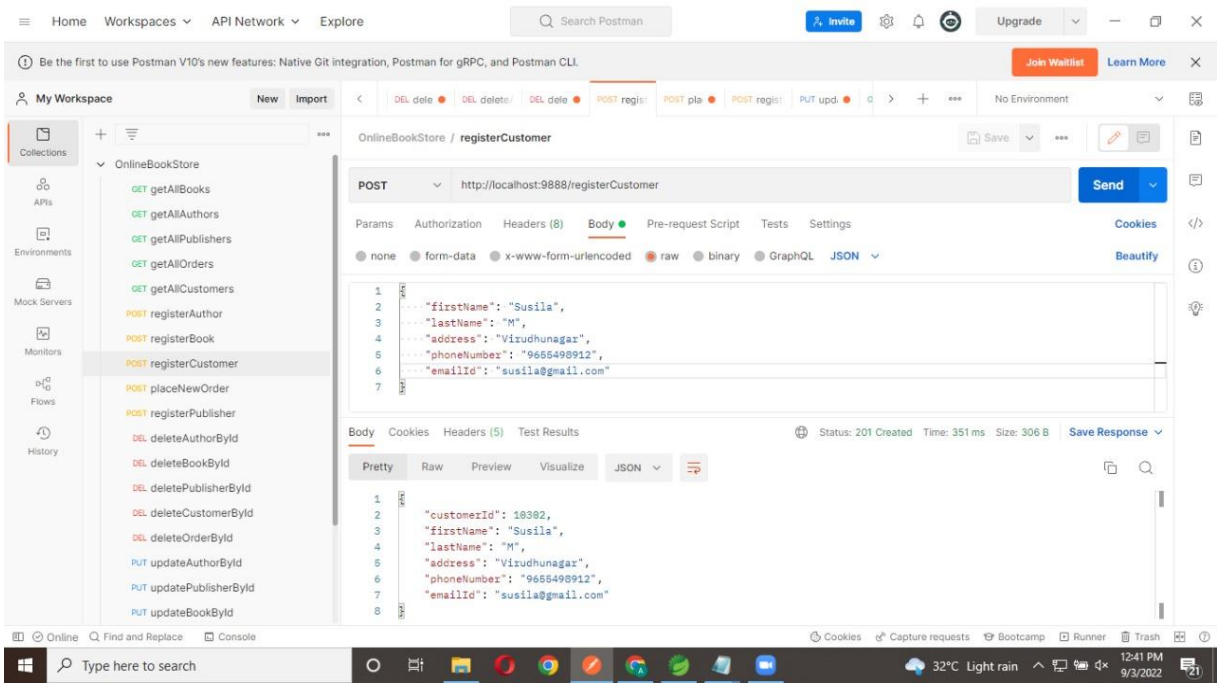
Step 7: We register the books by using book Id.

URL: http://localhost:9888//registerBook



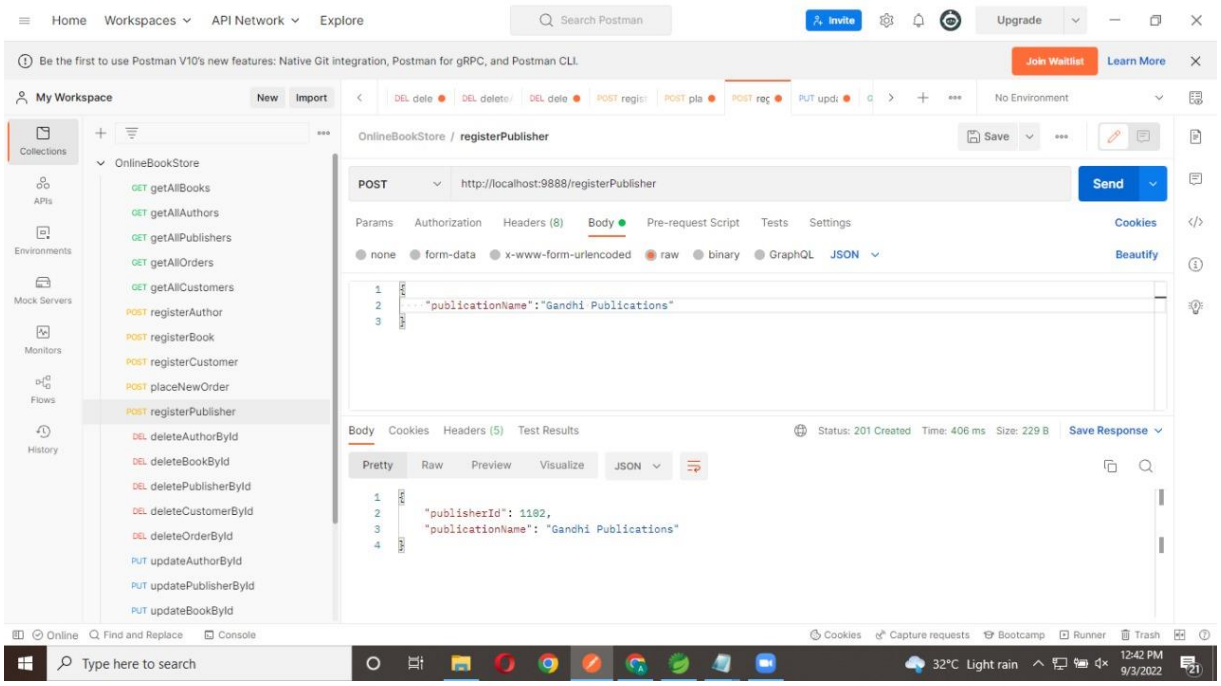
Step 8: Customer can register with customer register Id.

URL: http://localhost:9888//registerCustomer



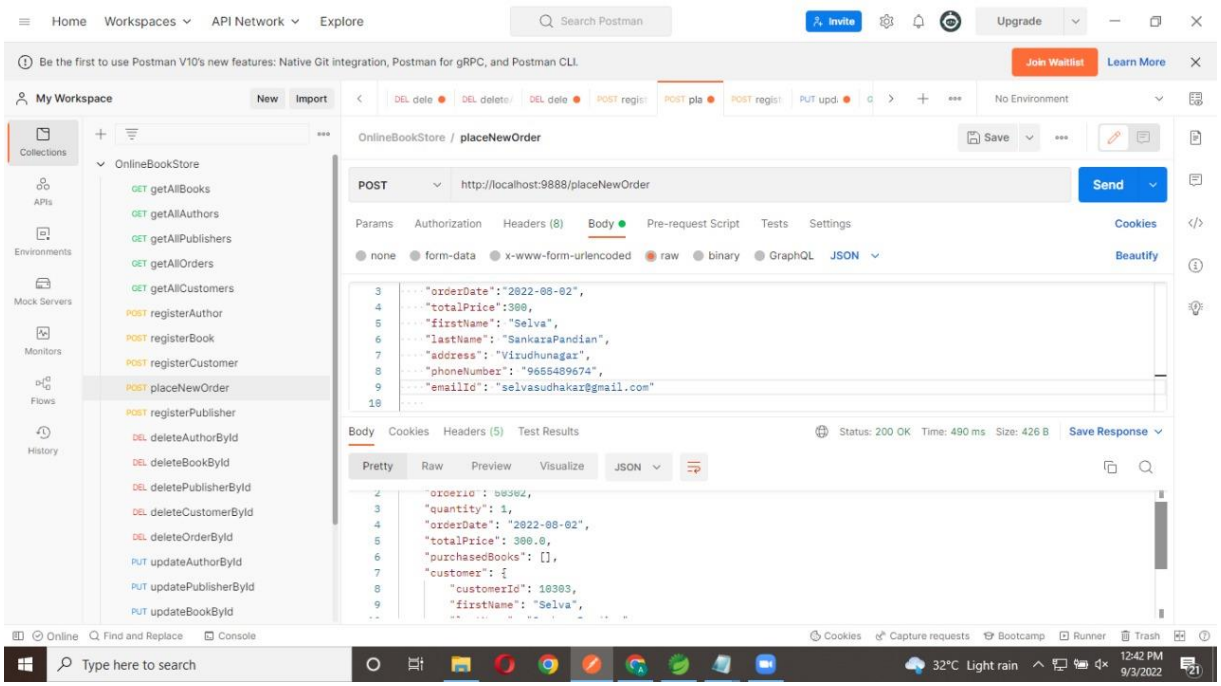
Step 9: Get publisher by using publisher Id.

URL: http://localhost:9888//registerPublisher



Step 10: We can place the new orders by using order Id.

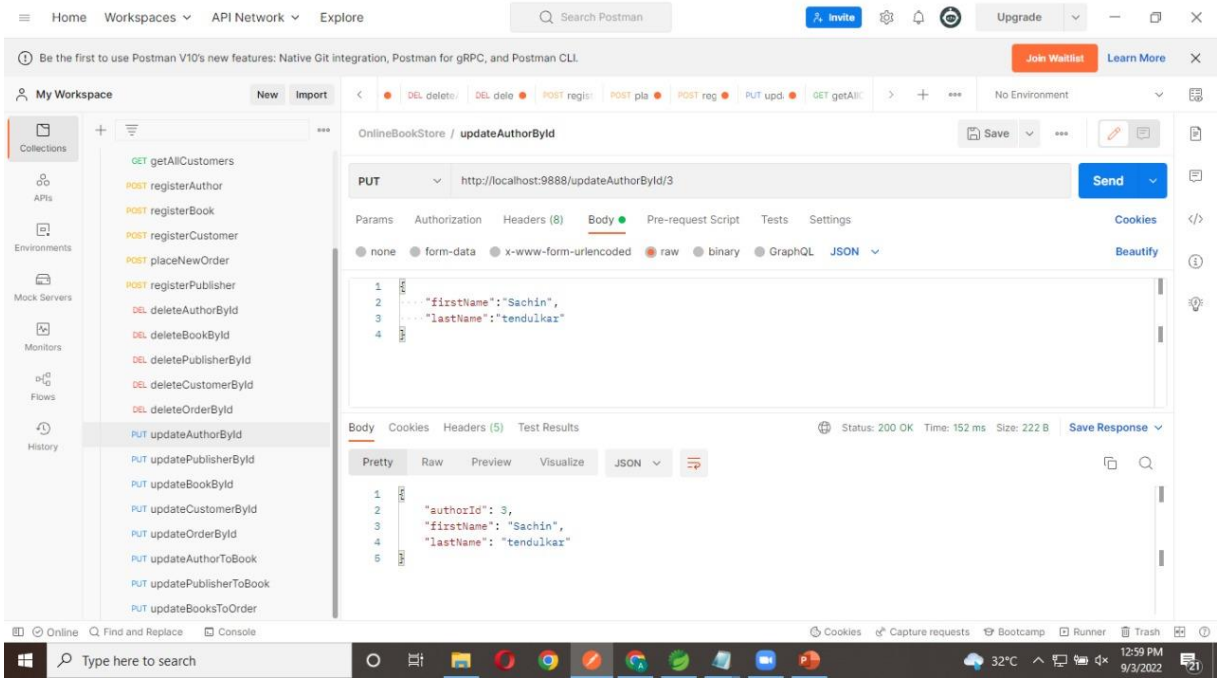
URL: http://localhost:9888//placeNewOrder



Put

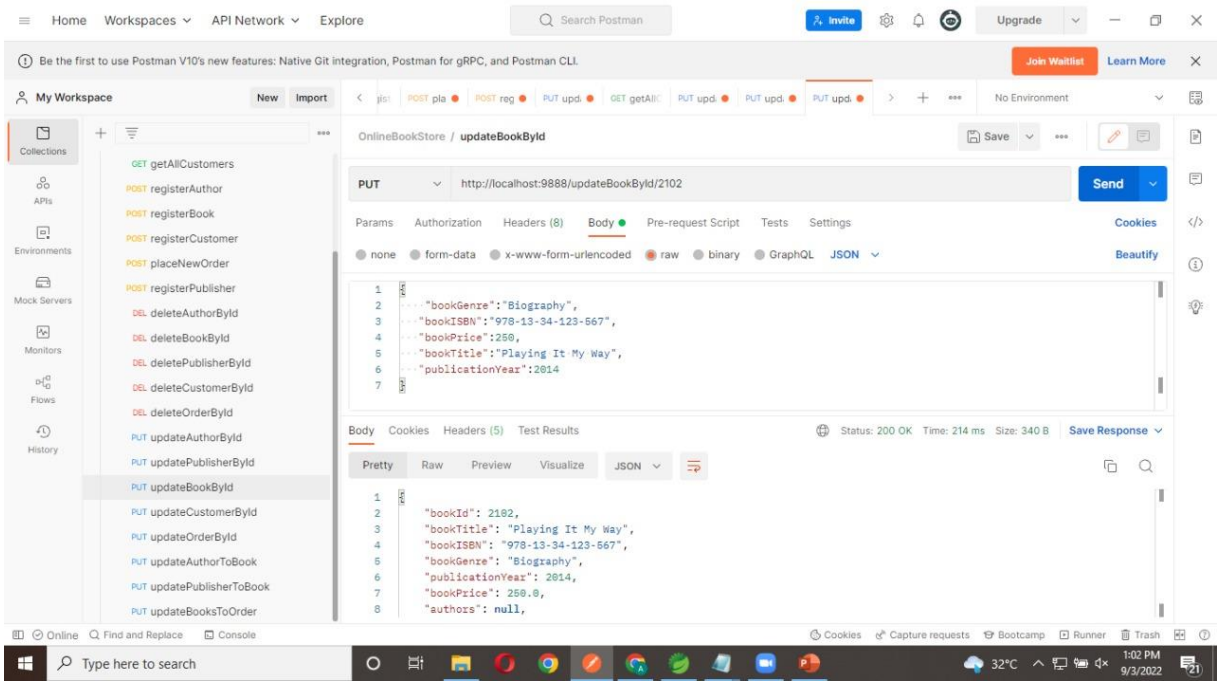
Step 11: By giving first name and last name we can update the author using author id.

URL: http://localhost:9888//updateAuthorById/3



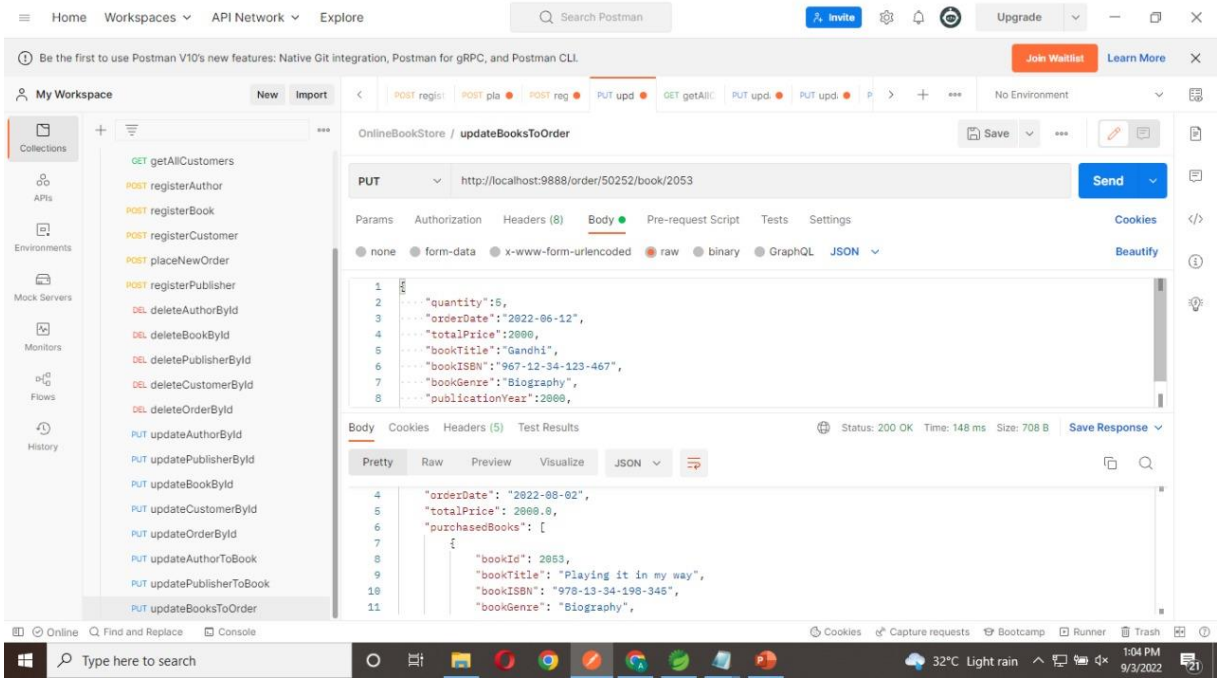
Step 12: To update book we have book id using updateBookById it starts from 2102.

URL: http://localhost:9888//updateBookById/2102



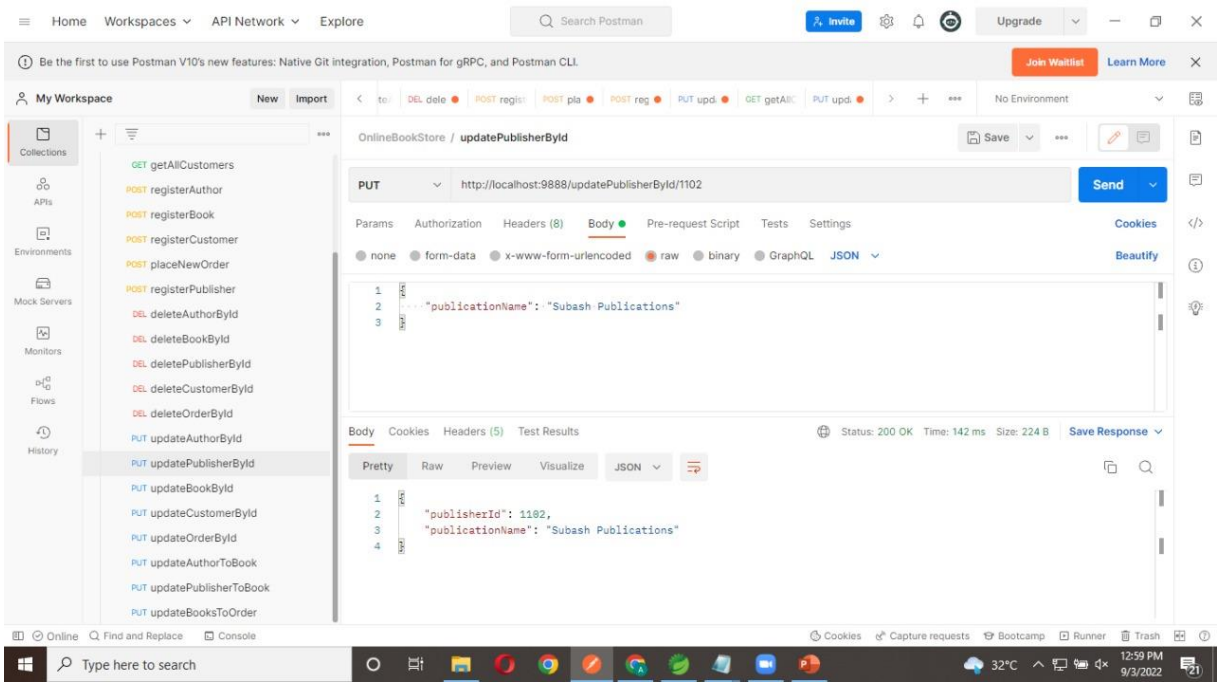
Step 13: Updating the orders we can see the order date and total price of the book.

URL: http://localhost:9888//updateOrderById/50152



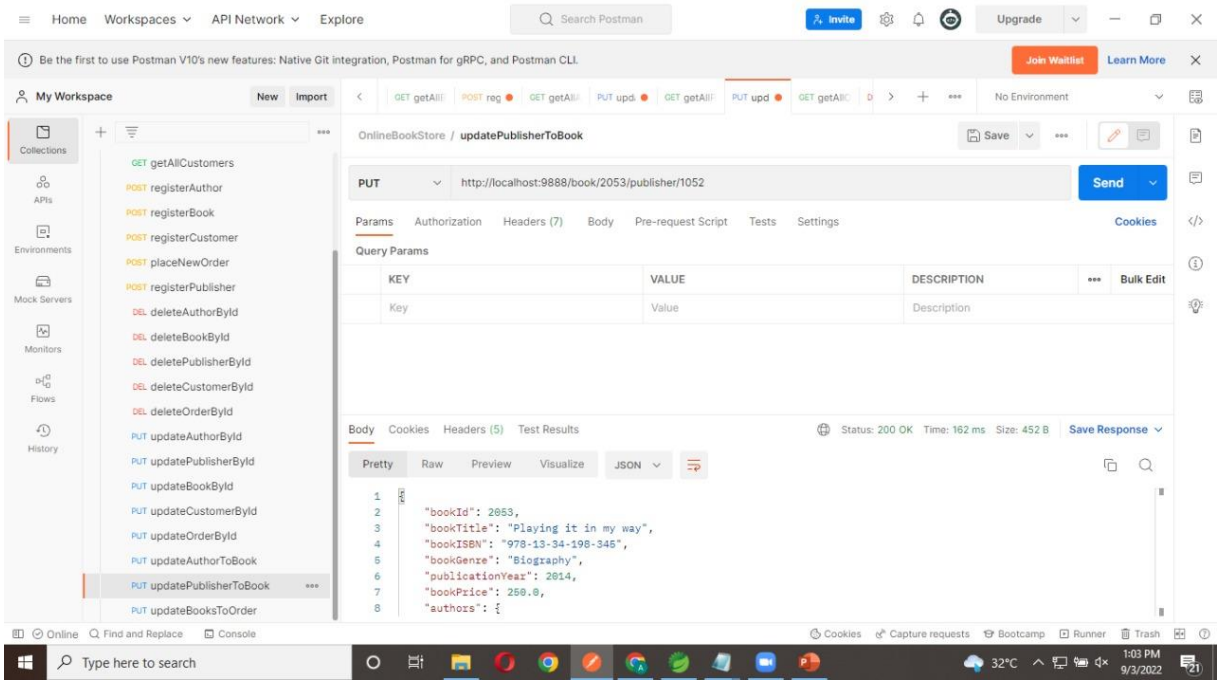
Step 14: We update the publication name using publisher id.

URL: http://localhost:9888//updatePublisherById/1102



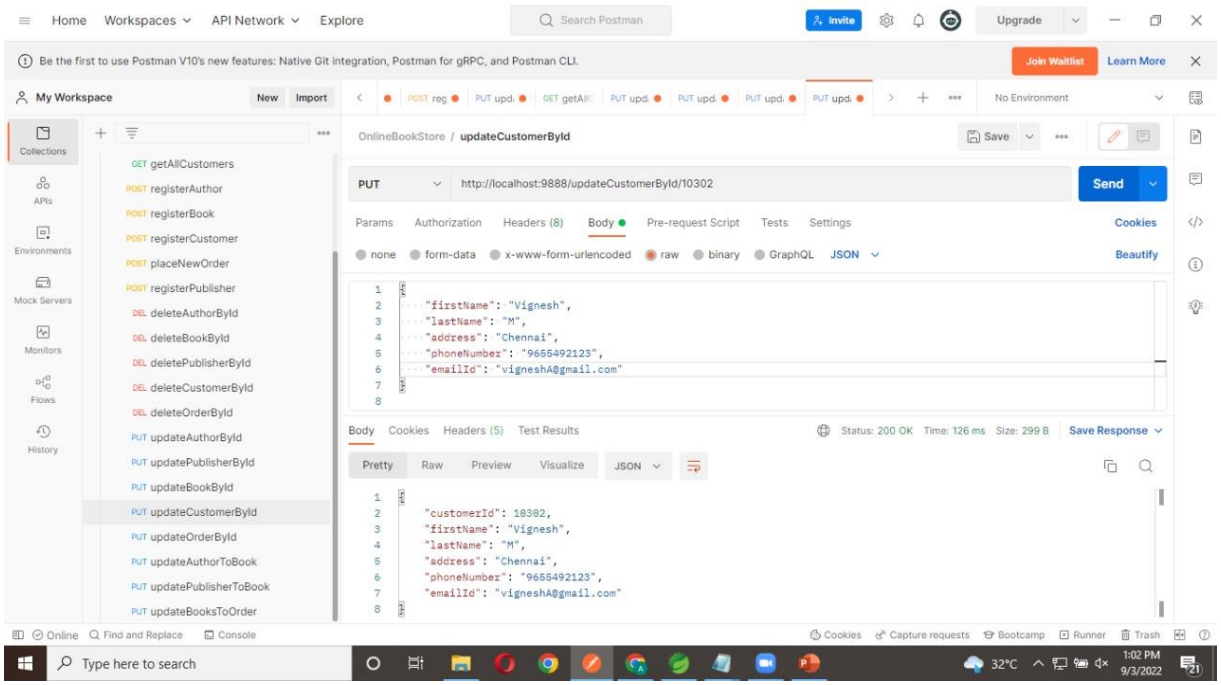
Step 15: Updating the publisher through books we can see the book title, ISBN it should be 13 digits and Book Genre.

URL: http://localhost:9888/book/2053/publisher/1052



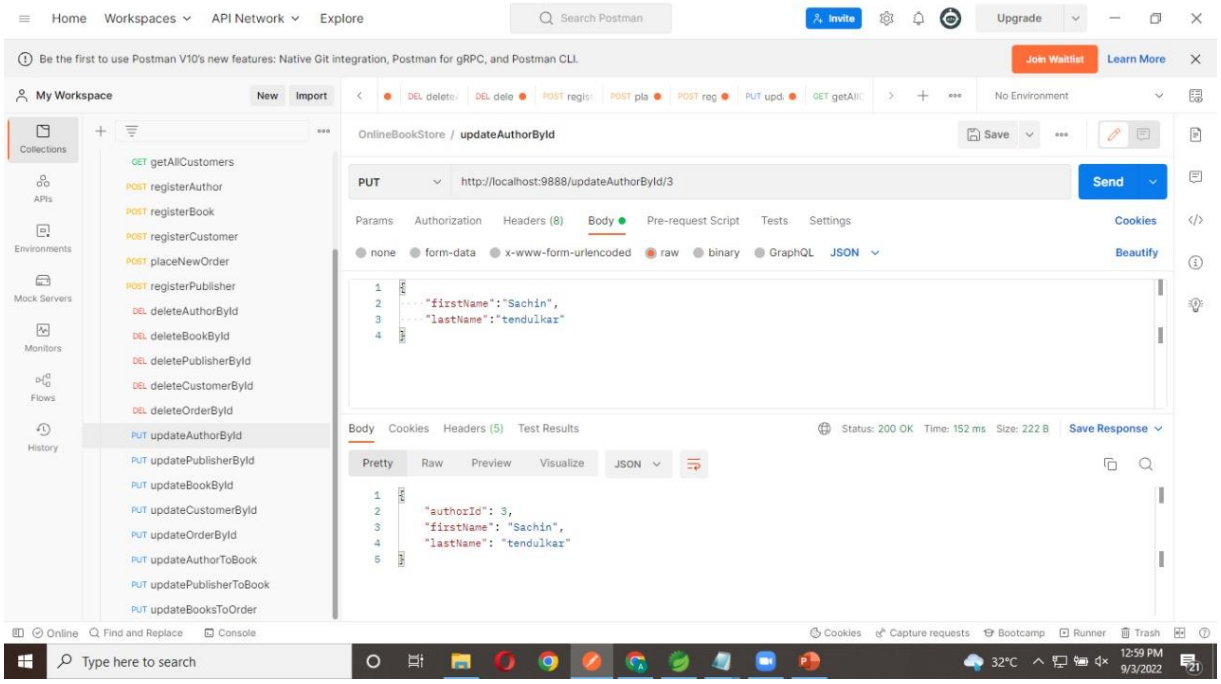
Step 16: To put the customer details by fetching customer id, name, address, phone number and email id.

URL: http://localhost:9888/updateCustomerById/10302

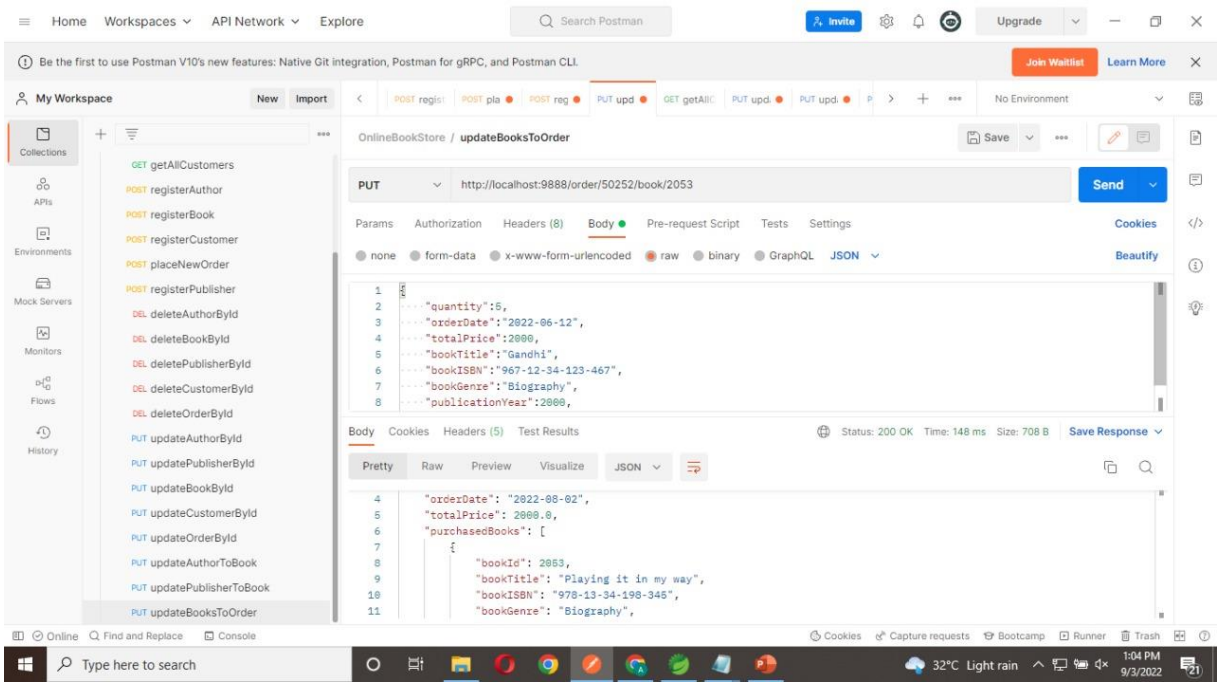


Step 17: To put the author details we use author id and name.

URL: http://localhost:9888/2053/author/3

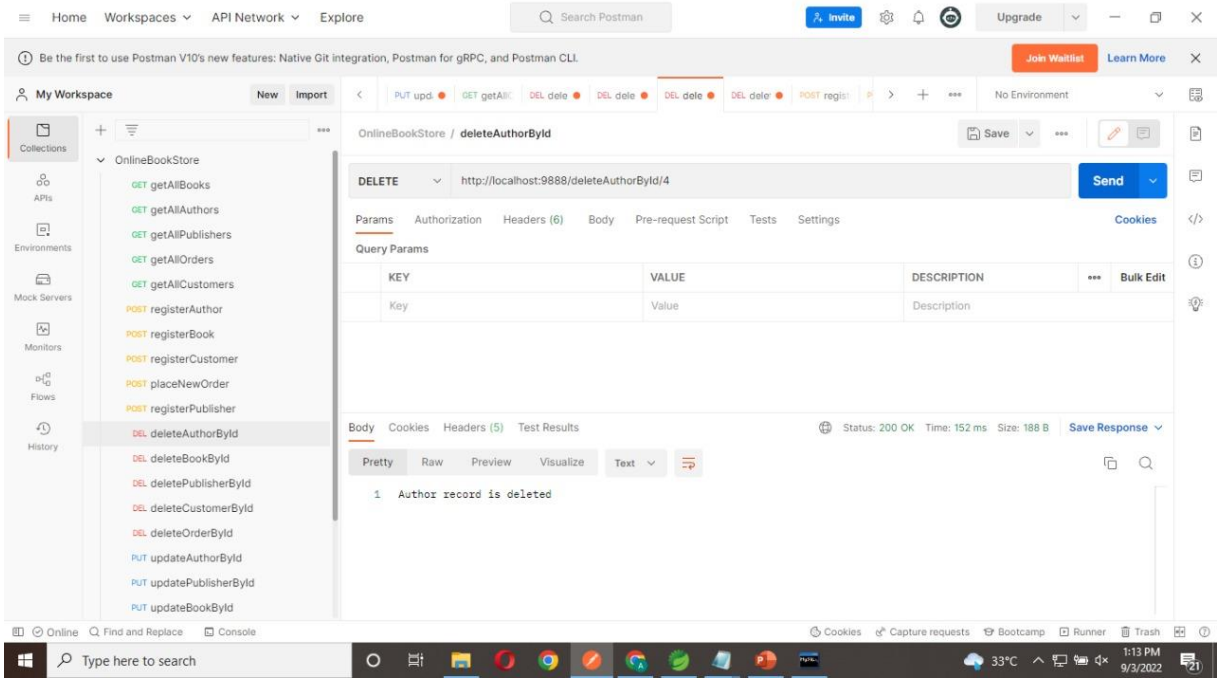


Step: 18 We put order we can get the details of the order date and prices.
URL: http://localhost:9888/order/50252/book/2053

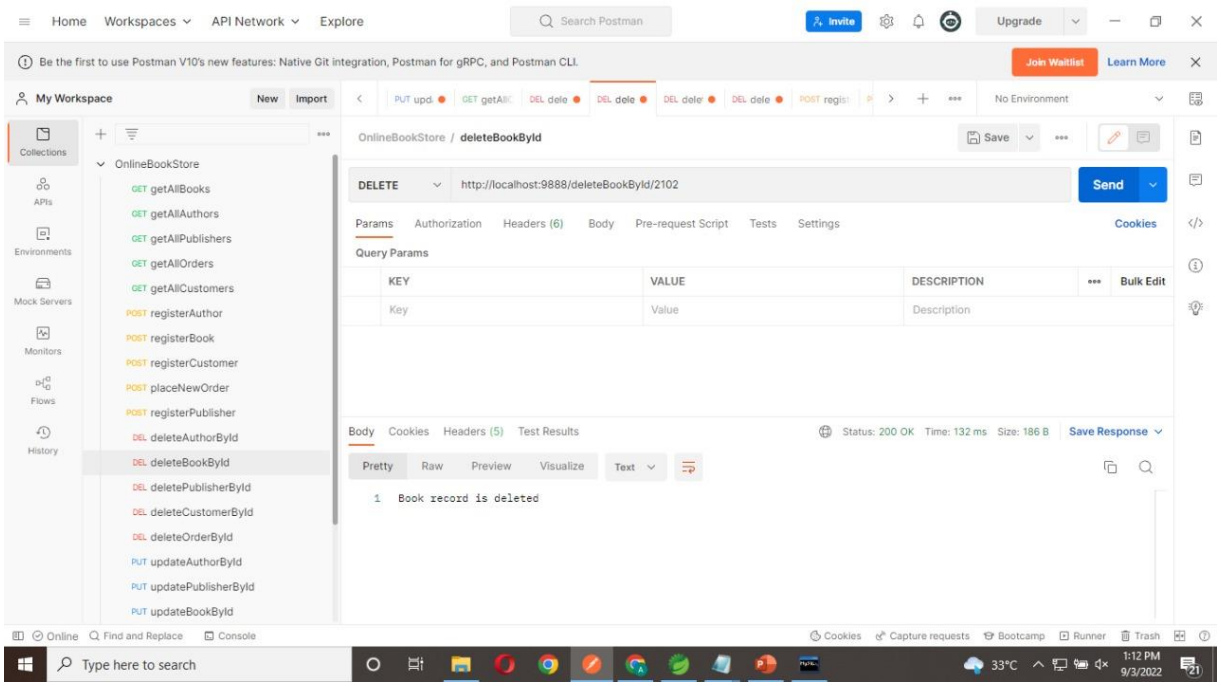


Delete

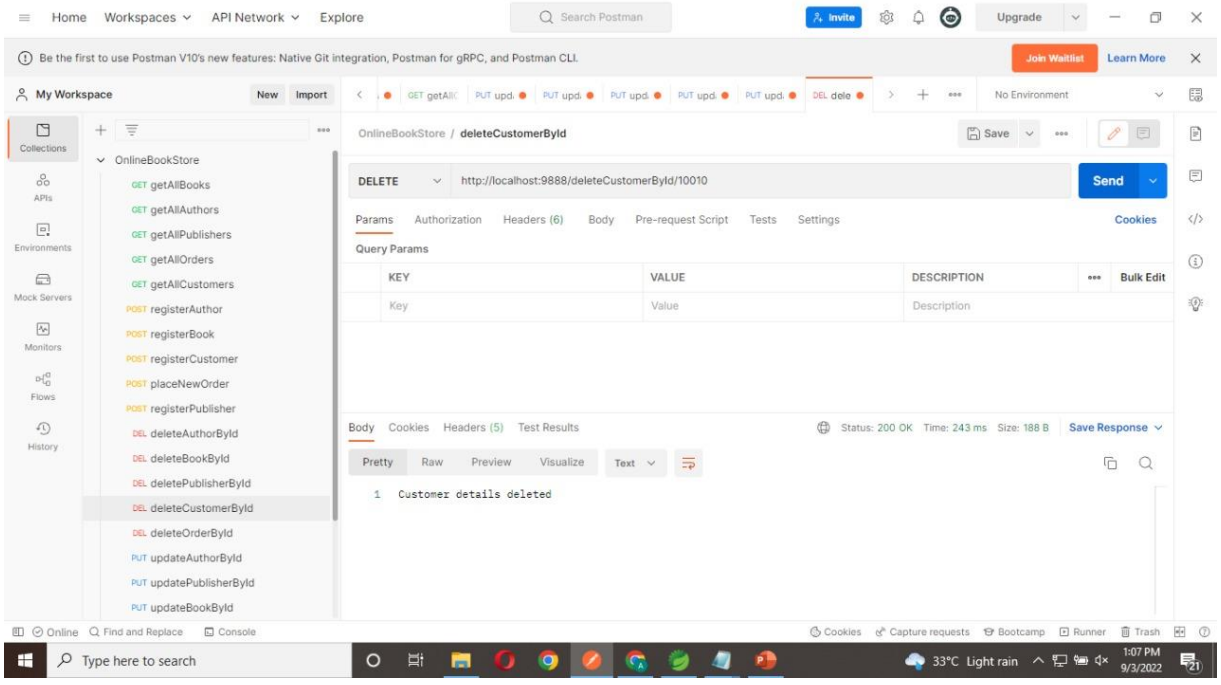
Step 19: We can delete the author by using author by id.
URL: http://localhost:9888//deleteAuthorById



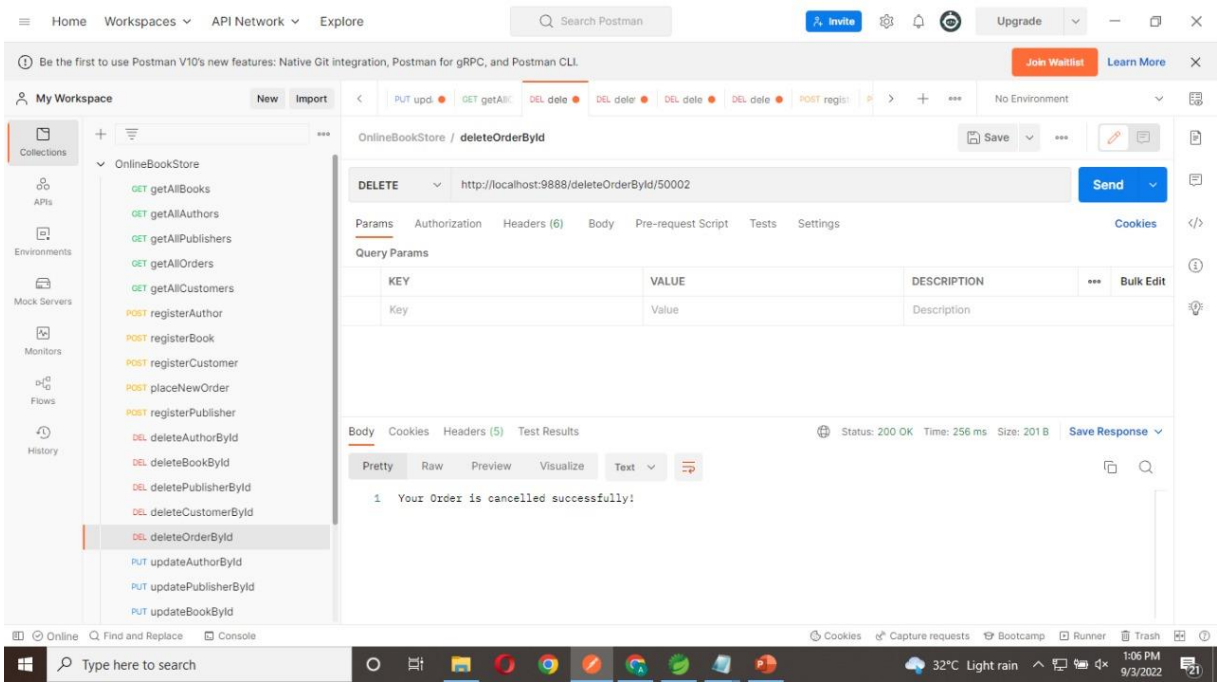
Step 20: We can delete the book by using delete book by id.
URL: http://localhost:9888//deleteBookById



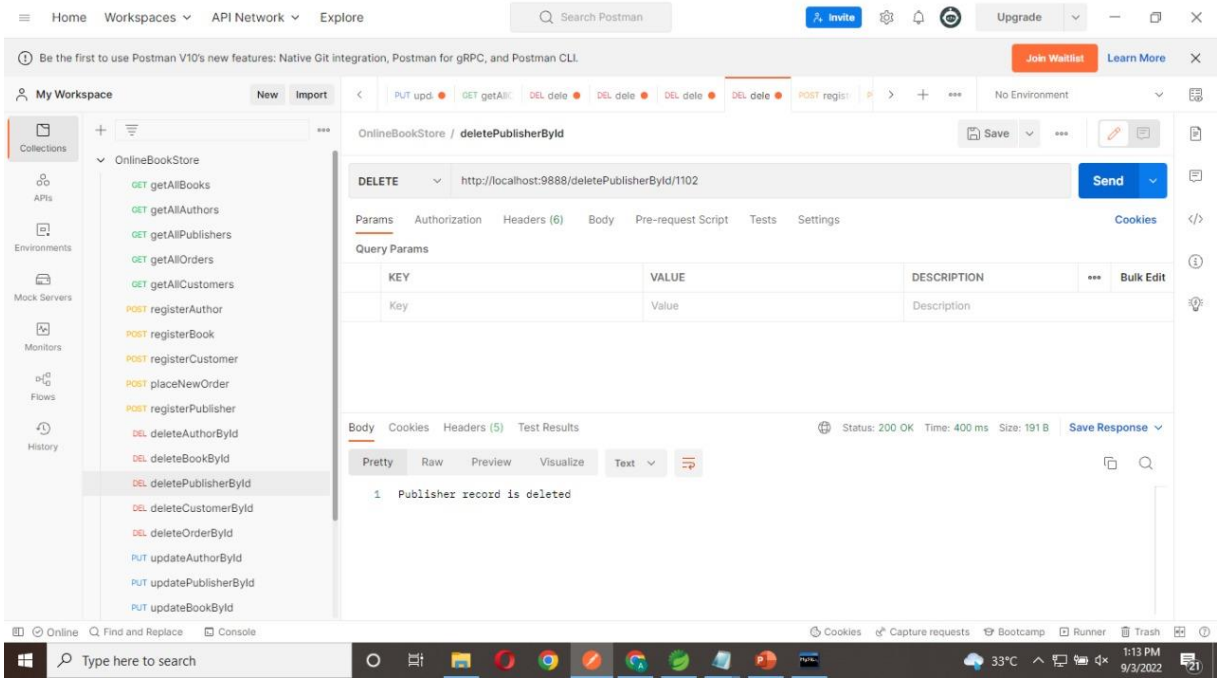
Step 21: We can delete the customer details using customer by id.
URL: http://localhost:9888//deleteCustomerById



Step 22: We can delete the order by using order id.
URL: http://localhost:9888//deleteOrderById



Step 23: We can delete the publication using publisher by id.
URL: http://localhost:9888//deletePublisherById



Annotations:

1.@Service:

We mark beans with @Service to indicate that they're holding the business logic. Besides being used in the service layer, there isn't any other special use for this annotation.

2. @Repository:

The @Repository's job is to catch persistence-specific exceptions and re-throw them as one of spring's unified unchecked exceptions.

3. @RestController:

The @RestController annotation is a convenience annotation that is itself annotated with @Controller and @ResponseBody.It is used to create RESTful web services using Spring MVC. Spring Rest Controller takes care of mapping request data to the defined request handler method.

4. @Autowired:

The spring framework enables automatic dependency injection. In other words, by declaring all the bean dependencies in a spring configuration file, Spring container can autowire relationships between collaborating beans. This is called spring bean autowiring.

5. @GetMapping:

The @GetMapping annotation is a specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping (method = RequestMethod.GET).

6. @PostMapping:

The @PostMapping is specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping (method = RequestMethod.POST). The @PostMapping annotated methods in the @Controller annotated classes handle the HTTP POST requests matched with given URI expression.

7. @PutMapping:

The @PutMapping is used to update/modify the resource so the @PutMapping annotation is used for mapping HTTP PUT requests onto specific handler methods. Specifically, @PutMapping is a composed annotation that acts as a shortcut for @RequestMapping (method = RequestMethod.PUT).

8.@DeleteMapping:

The @DeleteMapping annotation maps HTTP DELETE requests onto specific handler methods. It is a composed annotation that acts as a shortcut for @RequestMapping (method = RequestMethod.DELETE).

9.@OneToMany:

A one-to-many relationship between two entities is defined by using the @OneToMany annotation in Spring Data JPA. It declares the mappedBy element to indicate the entity that owns the bidirectional relationship. Usually, the child entity is onethat owns the relationship and the parent entity contains the @OneToMany annotation.

10.@ManyToOne:

The @many-to-one mapping means that one parent record can have multiple child records. In other words, multiple records of a table can associate themselves with a common record in another table.

11.@generatedValue:

Marking a field with the @GeneratedValue annotation specifies that a value will be automatically generated for that field. This is primarily intended for primary key fields but Object DB also supports this annotation for non-key numeric persistent fields as well.

12.@entity:

The @Entity annotation specifies that the class is an entity and is mapped to a database table. The @Table annotation specifies the name of the database table to be used for mapping. Database Table

13.@one to one:

One-to-One relationship in JPA, each entity instance is related to a single instance of another entity. It means each row of one entity is referred to one and only one row of another entity.

14. @Many to Many:

A relationship is a connection between two types of entities. In the case of a many-to-many relationship, both sides can relate to multiple instances of the other side.

Database Table Design:

Authors Table

```
mysql> desc authors;
+------------+--------------+------+-----+---------+----------------+
| Field      | Type         | Null | Key | Default | Extra          |
+------------+--------------+------+-----+---------+----------------+
| author_id  | int          | NO   | PRI | NULL    | auto_increment |
| first_name | varchar(255) | YES  |     | NULL    |                |
| last_name  | varchar(255) | YES  |     | NULL    |                |
+------------+--------------+------+-----+---------+----------------+
```

Books Table

```
mysql> desc books;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| book_id          | int          | NO   | PRI | NULL    |       |
| book_genre       | varchar(255) | YES  |     | NULL    |       |
| bookisbn         | varchar(18)  | YES  | UNI | NULL    |       |
| book_price       | float        | YES  |     | NULL    |       |
| book_title       | varchar(255) | YES  |     | NULL    |       |
| publication_year | int          | YES  |     | NULL    |       |
| author_id        | int          | YES  | MUL | NULL    |       |
| publisher_id     | int          | YES  | MUL | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
8 rows in set (0.04 sec)
```

Customers Table

```
mysql> desc customers;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| customer_id  | int          | NO   | PRI | NULL    |       |
| address      | varchar(255) | YES  |     | NULL    |       |
| email_id     | varchar(255) | YES  | UNI | NULL    |       |
| first_name   | varchar(30)  | YES  |     | NULL    |       |
| last_name    | varchar(30)  | YES  |     | NULL    |       |
| phone_number | varchar(10)  | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
6 rows in set (0.03 sec)
```

Orders Table

```
mysql> desc orders;
+-------------+-------+------+-----+---------+-------+
| Field       | Type  | Null | Key | Default | Extra |
+-------------+-------+------+-----+---------+-------+
| order_id    | int   | NO   | PRI | NULL    |       |
| order_date  | date  | YES  |     | NULL    |       |
| quantity    | int   | YES  |     | NULL    |       |
| total_price | float | NO   |     | NULL    |       |
| customer_id | int   | YES  | MUL | NULL    |       |
+-------------+-------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Publishers Table

```
mysql> desc publishers;
+------------------+--------------+------+-----+---------+-------+
| Field            | Type         | Null | Key | Default | Extra |
+------------------+--------------+------+-----+---------+-------+
| publisher_id     | int          | NO   | PRI | NULL    |       |
| publication_name | varchar(255) | YES  |     | NULL    |       |
+------------------+--------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Conclusion: Online Bookstore Application makes customer jobs more accessible by giving them an easy place to find and sort information.

THANK YOU!!