PROJECT REPORT

On

Book Store Management System Using Rest API

By

(**Batch: 2022-7469**)

Jambhale Dipali- EBEON0522602992

Pratiksha Vaidya- EBEON0522601614

Tupake Pallavi- EBEON0522601621

Nehe Shreya- EBTSOC0522598195

Arkhade Prajkta- EBEON0522601543

Under the Supervision of

Pooja Mehta Mam

EduBridge

India's leading Workforce Development Platform

Table of Content

1. Introduction

Book Store Management System is the web application to automate all kinds of operations in the book shore. The purpose of this software is to manage the books in the book store. Generally, it includes the Order Processing, Stock Management. We developed this software to maintain records of sales, purchase and Author. This project developed using Java Spring Boot and SQL Server as Back end. Here we are try to develop such type system which is provide the automation on the any type of the bookshop.

2. Objective

It is designed to make the existing manual system automatic with the help of computerised equipment and full-edged computer software, fulfilling their requirements, so that their valuable data and information can be stored for a longer period with easy access and manipulation of the same. The required software is easily available and easy to work with.

3. Technology Used

➢ Spring Tool Suite

Spring Boot is a java framework used for develop standalone application. Need strong knowledge in OOPS & Java concept then only working with spring framework. Mostly all applications are developed by spring boot. Because it was very secure no one hack the information. Spring Boot is an open source Java-based framework used to create a micro Service. It is developed by Pivotal Team and is used to build stand-alone and production ready spring applications.

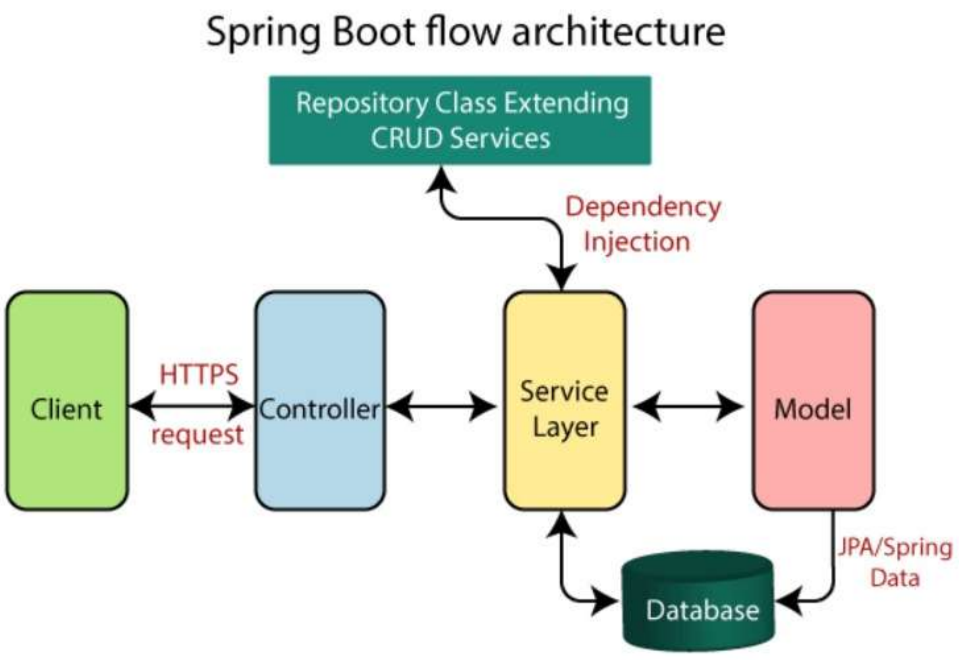


Fig. Spring Boot flow architecture.

- **MySQL**

  MySQL creates a database for storing and manipulating data, defining the relationship of each table. Clients can make requests by typing specific SQL statements on MySQL. The server application will respond with the requested information and it will appear on the clients' side. MySQL works with an operating system to implement a relational database in a computer's storage system, manages users, allows for network access and facilitates testing database integrity and creation of backups. MySQL is ideal for both small and large applications.

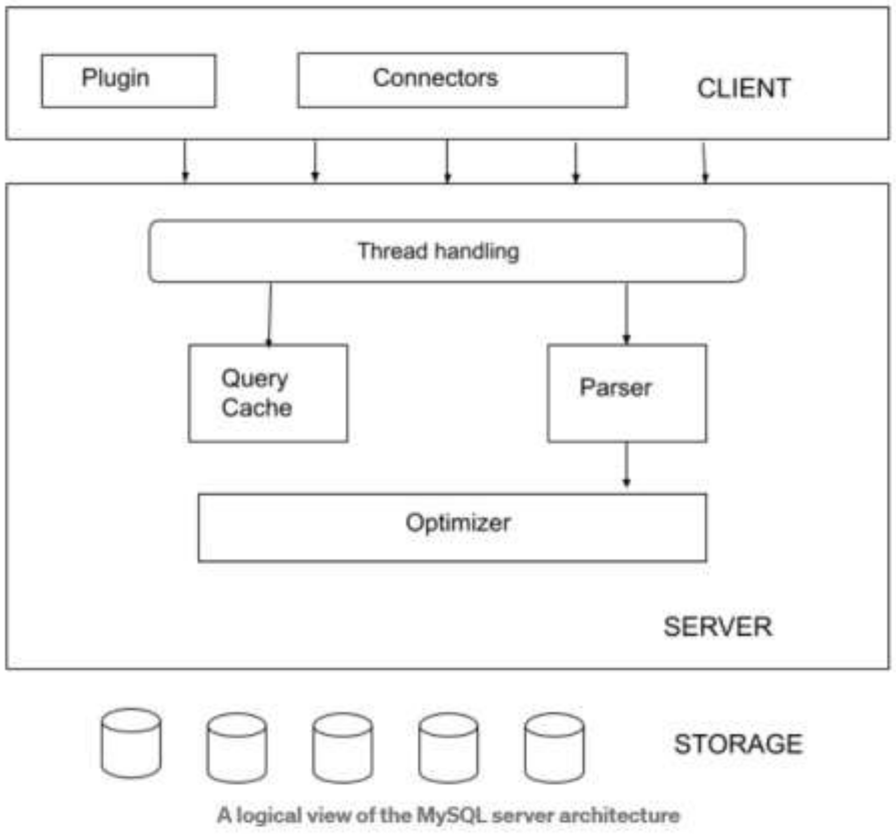


Fig. MySQL Server architecture.

- **Postman**

  Postman is an API client that makes it easy for developers to create, share, test and document APIs. With this open-source solution, users can create and save simple and complex HTTP/s requests, as well as read their responses. When you are signed into your account, you are able to access your files. You can execute Postman API tests anytime, anywhere.

  Postman is very convenient when it comes to executing APIs. Once you’ve entered and saved them, you can simply use them over and over again, without having to remember the exact endpoint, headers, or API keys.

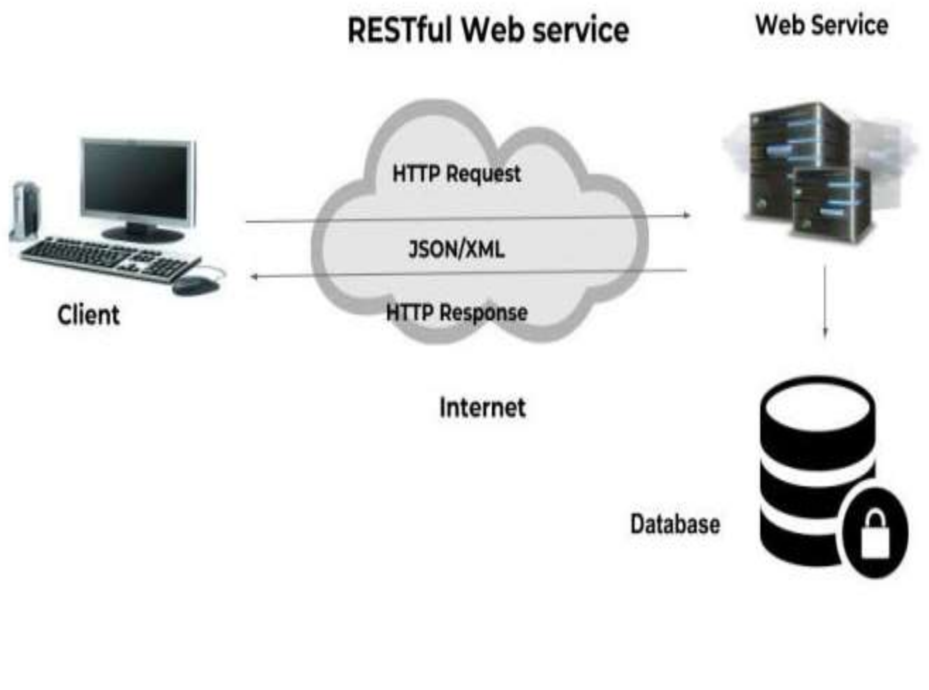


Fig. Working of Postman

4. Module Used

- Entity

  The entities are the persistence objects stores as a record in the database. Persistence Unit: It defines a set of all entity classes. In an application, Entity Manager Instances manage it. The set of entity classes represents the data contained within a single data store.

- Repository

  Spring boot framework provides us repository which is responsible to perform various operations on the object. Repository classes are auto detected by spring framework through class path scanning.

- Service

  Spring boot service component is defined as a class file that includes the @Service annotation and allows developers to add business functionalities. The annotation is used with the classes that provide these business functionalities.

- Controller

  In Spring Boot, the controller class is responsible for processing incoming REST API requests, preparing a model, and returning the view to be rendered as a response. The controller classes in spring are annotated either by the @Controller or the @RestController annotation.

5. Annotation

1. @Autowired:

   Spring provides annotation-based auto-wiring by providing @Autowired annotation. It is used to autowire spring bean on setter methods, instance variable, and constructor. When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.

2. @RequestMapping:

   It is used to map the web requests. It has many optional elements like consumes, header, method, name, params, path, produces, and value.

3. @RestController:

   It is a convenience annotation for creating Restful controllers. It is a specialization of @Component and is autodetected through classpath scanning.

4. @GetMapping:

   It maps the HTTP GET requests on the specific handler method. It is used to create a web service endpoint that fetches It is used instead of using: @RequestMapping(method = RequestMethod.GET)

5. @PostMapping:
   It maps the HTTP POST requests on the specific handler method. It is used to create a web service endpoint that creates It is used instead of using: @RequestMapping(method = RequestMethod.POST)

6. @PutMapping:
   It maps the HTTP PUT requests on the specific handler method. It is used to create a web service endpoint that creates or updates It is used instead of using: @RequestMapping(method = RequestMethod.PUT)

7. @DeleteMapping:
   It maps the HTTP DELETE requests on the specific handler method. It is used to create a web service endpoint that deletes a resource. It is used instead of using:@RequestMapping(method =RequestMethod.DELETE)

8. @PatchMapping:
   It maps the HTTP PATCH requests on the specific handler method. It is used instead of using: @RequestMapping(method = RequestMethod.PATCH)

9. @NotNull:
   It is actually, an explicit contract declaring that: A method should not return null. Variables (fields, local variables, and parameters) cannot hold a null value.

10. @SpringBootApplication:
This annotation is used to mark a configuration class that declares one or more @Bean methods and also triggers auto-configuration and component scanning.

11.@Id:
This annotation is inherited from javax.persistence.Id, indicating the member field below is the primary key of the current entity.

12.@Generated:
The Generated annotation is used to mark source code that has been generated.

14. @ManyToOne:
This mapping means that many instances of this entity are mapped to one instance of another entity – many items in one cart.

15.@ GeneratedValue:
This annotation specifies the generation strategies for the values of primary keys

16.@ Column:
This annotation is used for Adding the column the name in the table of a particular MySQL database.

17. @JoinTable:
This specifies the mapping of associations. It is applied to the owning side of an association.

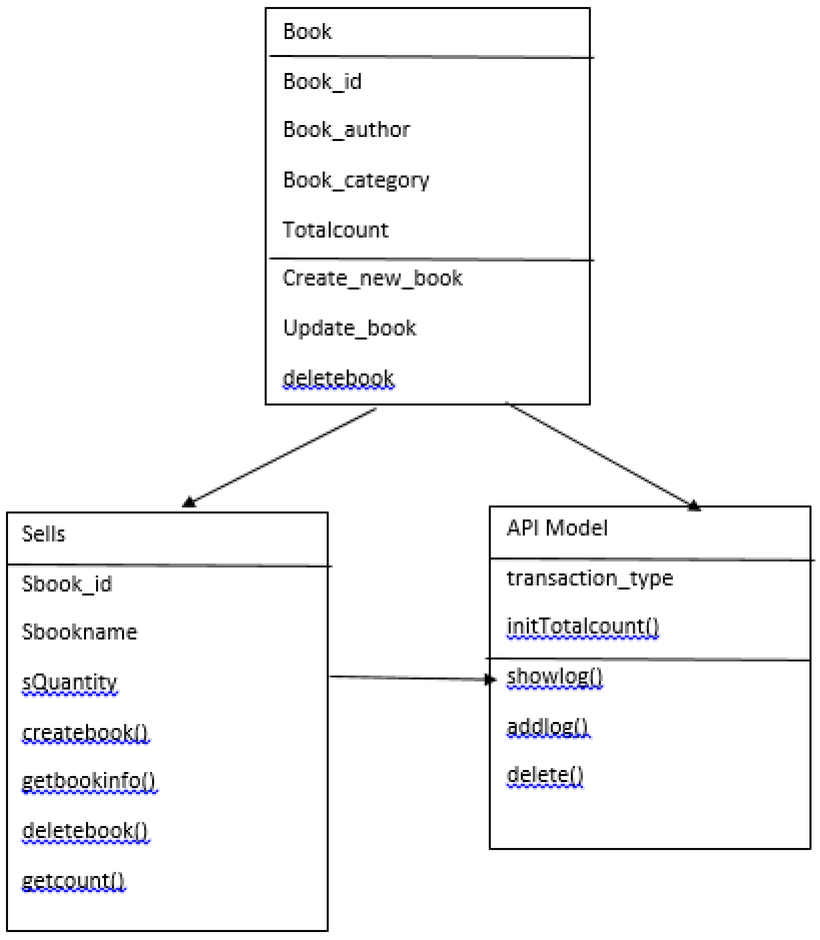
18.@Service:
It is also used at class level. It tells the spring that class contains the business logic.

19. @Entity:
This annotation defines that a class can be mapped to a table. And that is it, it is just a marker, like for example Serializable interface.

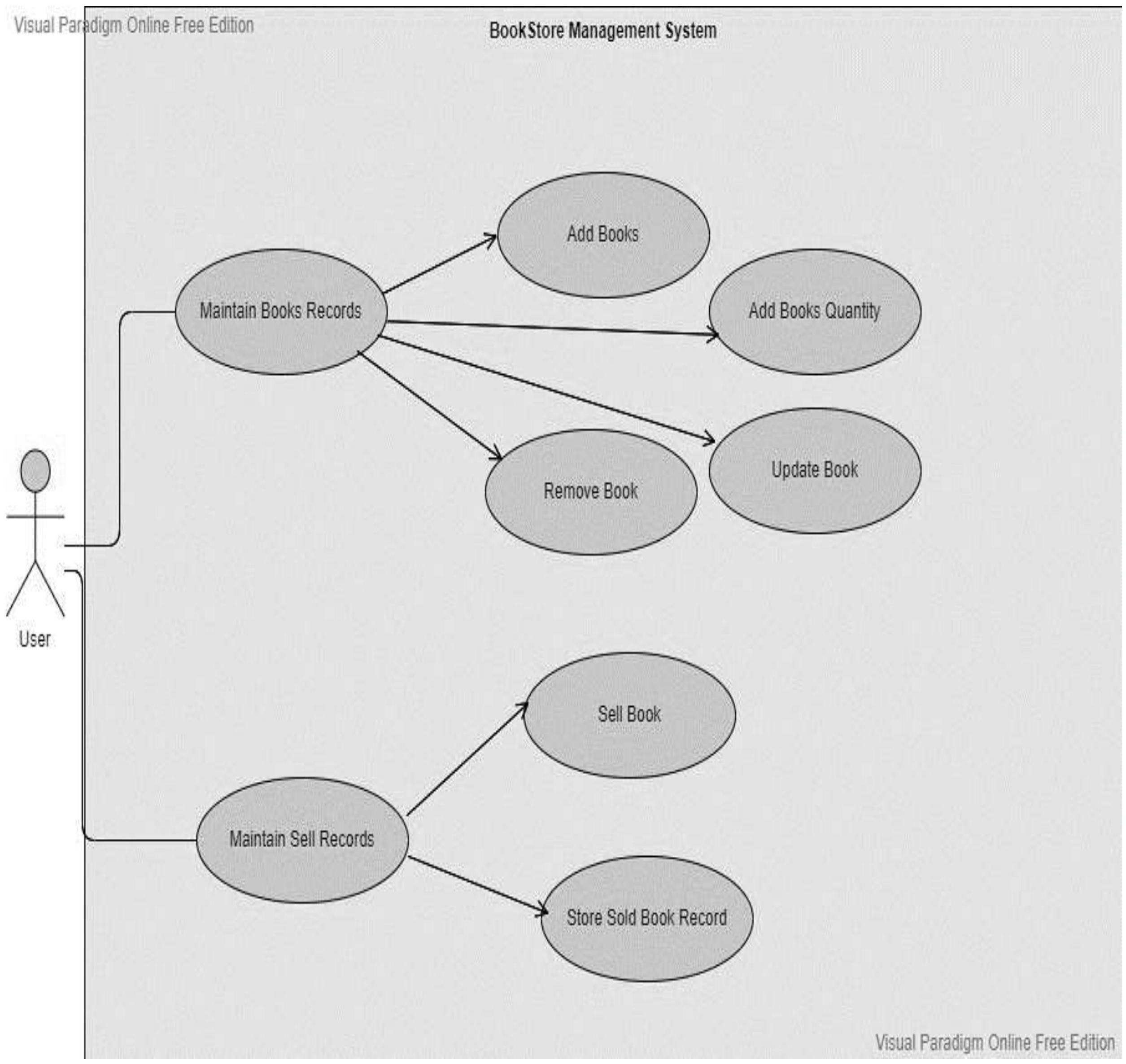
20.@JoinColumn:
Specifies a column for joining an entity association or element collection.

6. Diagram

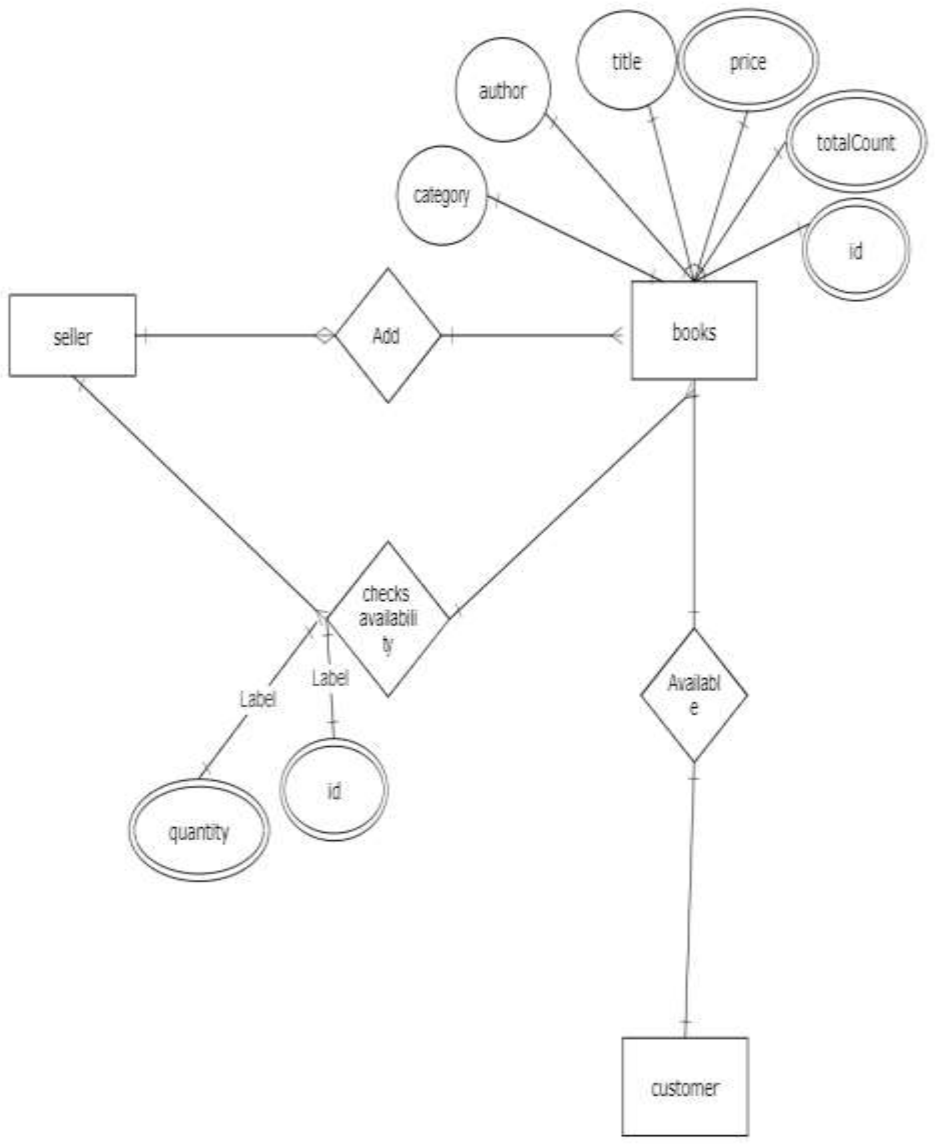
1. Class diagram



2. Use case diagram



3. ER diagram

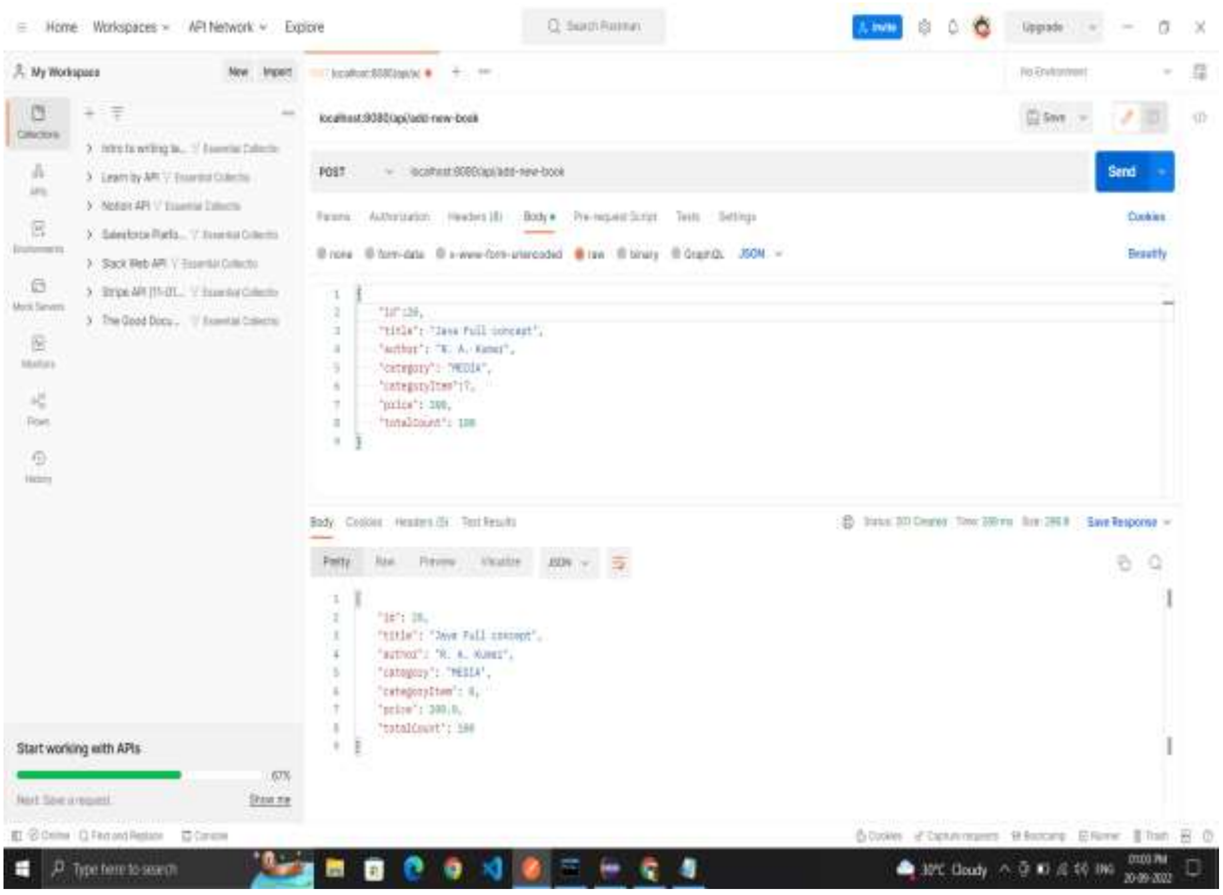


author
title
price
totalCount
category
id
seller
Add
books
checks availability
Available
Label
Label
quantity
id
customer

7. Screenshots

1. Post:

Adding new Books:



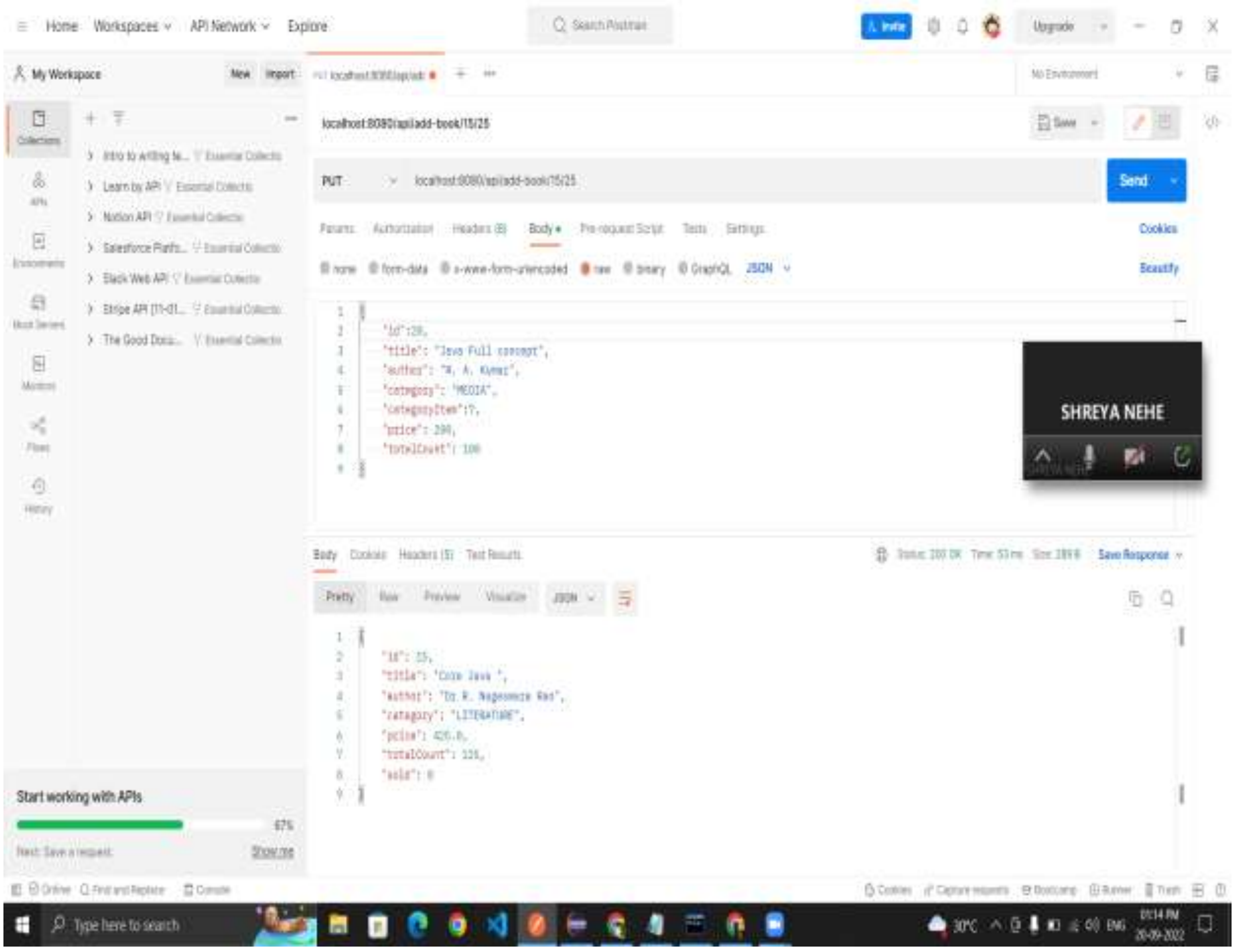
POST localhost:8080/api/add-new-book

```
{
    "id":28,
    "title": "Java Full concept",
    "author": "R. A. Kumar",
    "category": "MEDIA",
    "categoryItem":7,
    "price": 300,
    "totalCount": 100
}
```

```
{
    "id": 28,
    "title": "Java Full concept",
    "author": "R. A. Kumar",
    "category": "MEDIA",
    "categoryItem": 0,
    "price": 300.0,
    "totalCount": 100
}
```

2. PUT:

1. This method add quantity of book to the books which are already registered.



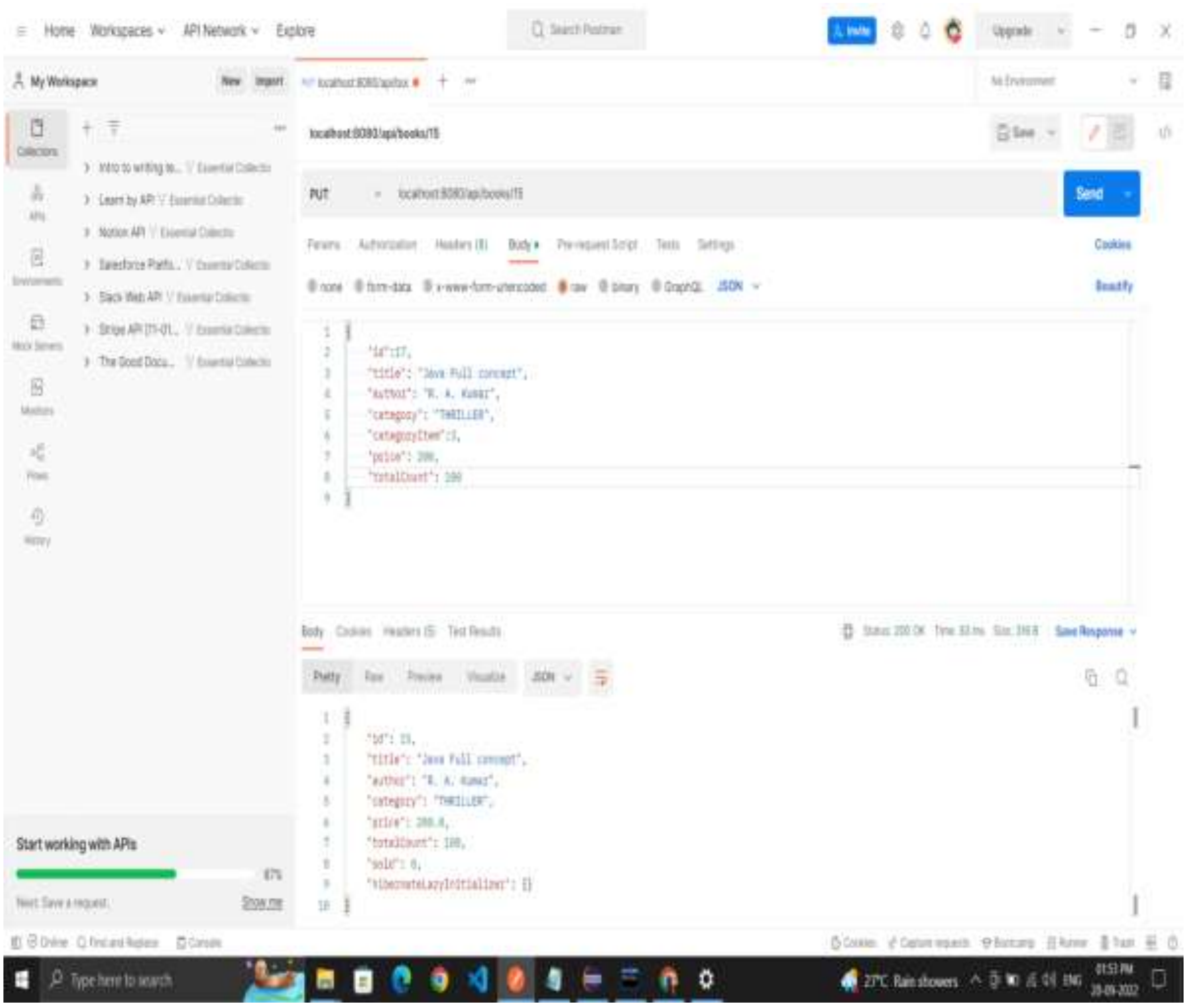
PUT localhost:8080/api/add-book/15/25

```
{
    "id":20,
    "title": "Java Full concept",
    "author": "R. A. Kumar",
    "category": "MEDIA",
    "categoryItem":7,
    "price": 200,
    "totalCount": 100
}
```

Status: 200 OK Time: 53 ms Size: 289 B

```
{
    "id": 15,
    "title": "Core Java ",
    "author": "Dr. R. Nageswara Rao",
    "category": "LITERATURE",
    "price": 425.0,
    "totalCount": 125,
    "sold": 0
}
```

2. Update a book.



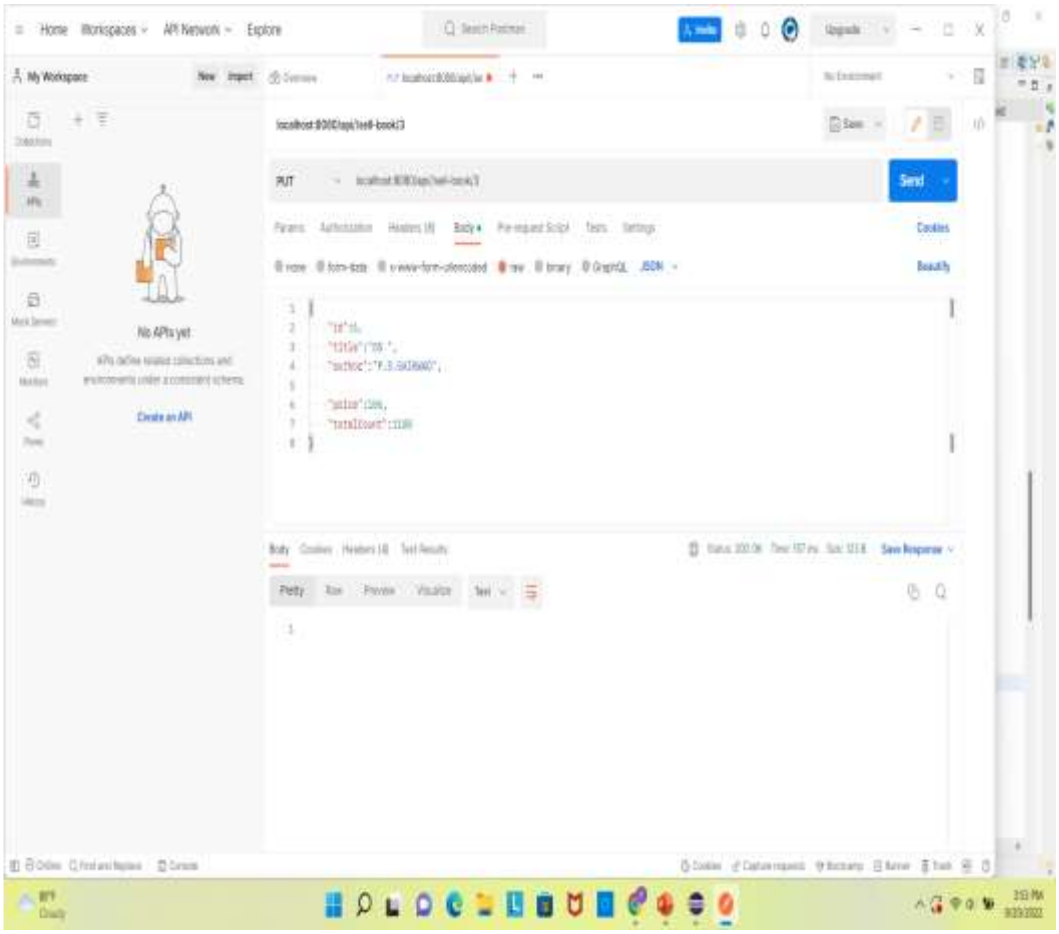
PUT localhost:8080/api/books/15

```
{
    "id":17,
    "title": "Java Full concept",
    "author": "R. A. Kumar",
    "category": "THRILLER",
    "categoryItem":3,
    "price": 200,
    "totalCount": 100
}
```

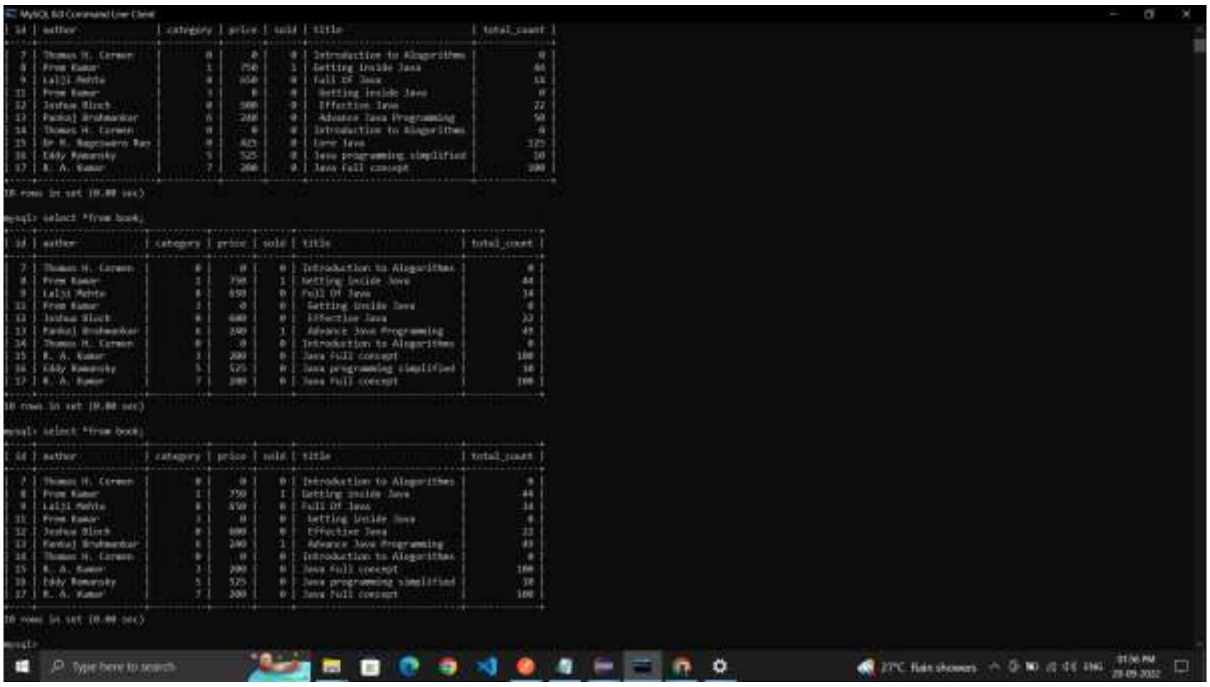
Status: 200 OK

```
{
    "id": 15,
    "title": "Java Full concept",
    "author": "R. A. Kumar",
    "category": "THRILLER",
    "price": 200.0,
    "totalCount": 100,
    "sold": 0,
    "hibernateLazyInitializer": {}
}
```

3. Sell Book



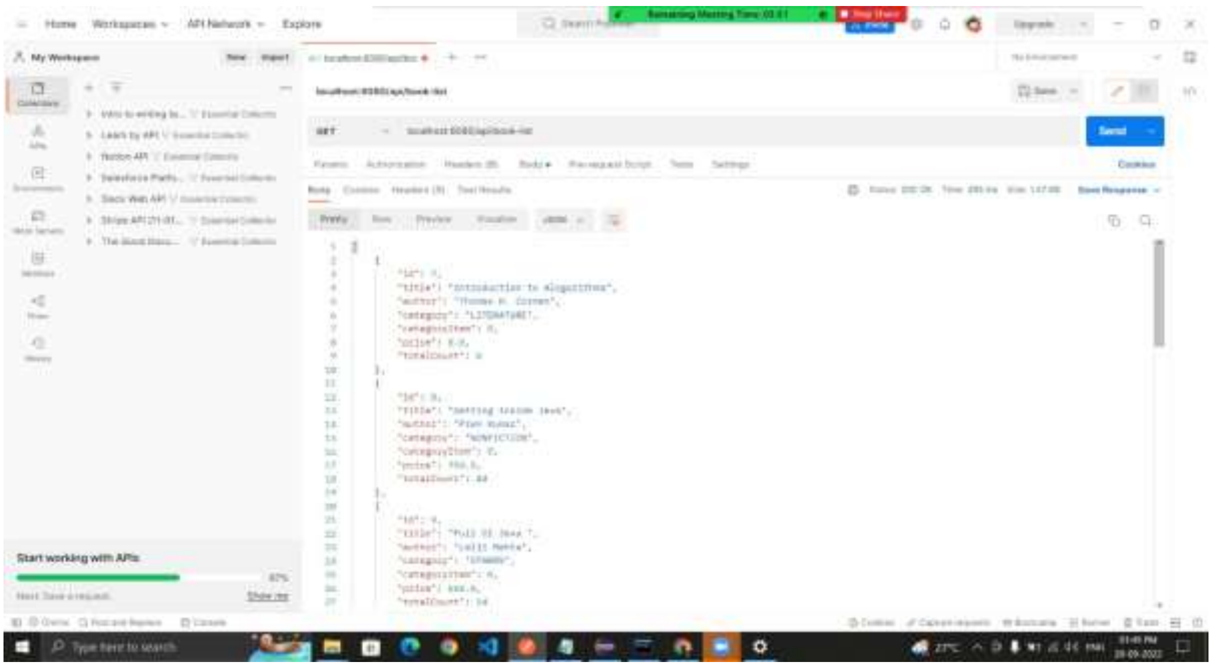
```
PUT   localhost:8080/api/sell-book/3

{
    "id":15,
    "title":"[illegible]",
    "author":"P.S.[illegible]",

    "price":[illegible],
    "totalCount":[illegible]
}
```



```
| id | author              | category | price | sold | title                          | total_count |
+----+---------------------+----------+-------+------+--------------------------------+-------------+
|  7 | Thomas H. Cormen    |        0 |     0 |    0 | Introduction to Alogorithms    |           0 |
|  8 | Prem Kumar          |        1 |   750 |    1 | Getting inside Java            |          44 |
|  9 | Lalji Mehta         |        8 |   650 |    0 | Full Of Java                   |          14 |
| 11 | Prem Kumar          |        1 |     0 |    0 |  Getting inside Java           |           0 |
| 12 | Joshua Bloch        |        0 |   500 |    0 |  Effective Java                |          22 |
| 13 | Pankaj Brahmankar   |        6 |   240 |    0 |  Advance Java Programming      |          50 |
| 14 | Thomas H. Cormen    |        0 |     0 |    0 | Introduction to Alogorithms    |           0 |
| 15 | Dr R. Nageswara Rao |        0 |   425 |    0 | Core Java                      |         125 |
| 16 | Eddy Romansky       |        5 |   525 |    0 | Java programming simplified    |          10 |
| 17 | R. A. Kumar         |        7 |   200 |    0 | Java Full concept              |         100 |
+----+---------------------+----------+-------+------+--------------------------------+-------------+
10 rows in set (0.00 sec)

mysql> select *from book;
+----+-------------------+----------+-------+------+-------------------------------+-------------+
| id | author            | category | price | sold | title                         | total_count |
+----+-------------------+----------+-------+------+-------------------------------+-------------+
|  7 | Thomas H. Cormen  |        0 |     0 |    0 | Introduction to Alogorithms   |           0 |
|  8 | Prem Kumar        |        1 |   750 |    1 | Getting inside Java           |          44 |
|  9 | Lalji Mehta       |        8 |   650 |    0 | Full Of Java                  |          14 |
| 11 | Prem Kumar        |        1 |     0 |    0 |  Getting inside Java          |           0 |
| 12 | Joshua Bloch      |        0 |   600 |    0 |  Effective Java               |          22 |
| 13 | Pankaj Brahmankar |        6 |   240 |    1 |  Advance Java Programming     |          49 |
| 14 | Thomas H. Cormen  |        0 |     0 |    0 | Introduction to Alogorithms   |           0 |
| 15 | R. A. Kumar       |        7 |   200 |    0 | Java Full concept             |         100 |
| 16 | Eddy Romansky     |        5 |   525 |    0 | Java programming simplified   |          10 |
| 17 | R. A. Kumar       |        7 |   200 |    0 | Java Full concept             |         100 |
+----+-------------------+----------+-------+------+-------------------------------+-------------+
10 rows in set (0.00 sec)

mysql> select *from book;
+----+-------------------+----------+-------+------+-------------------------------+-------------+
| id | author            | category | price | sold | title                         | total_count |
+----+-------------------+----------+-------+------+-------------------------------+-------------+
|  7 | Thomas H. Cormen  |        0 |     0 |    0 | Introduction to Alogorithms   |           0 |
|  8 | Prem Kumar        |        1 |   750 |    1 | Getting inside Java           |          44 |
|  9 | Lalji Mehta       |        8 |   650 |    0 | Full Of Java                  |          14 |
| 11 | Prem Kumar        |        1 |     0 |    0 |  Getting inside Java          |           0 |
| 12 | Joshua Bloch      |        0 |   600 |    0 |  Effective Java               |          22 |
| 13 | Pankaj Brahmankar |        6 |   240 |    1 |  Advance Java Programming     |          49 |
| 14 | Thomas H. Cormen  |        0 |     0 |    0 | Introduction to Alogorithms   |           0 |
| 15 | R. A. Kumar       |        7 |   200 |    0 | Java Full concept             |         100 |
| 16 | Eddy Romansky     |        5 |   525 |    0 | Java programming simplified   |          10 |
| 17 | R. A. Kumar       |        7 |   200 |    0 | Java Full concept             |         100 |
+----+-------------------+----------+-------+------+-------------------------------+-------------+
10 rows in set (0.00 sec)

mysql>
```

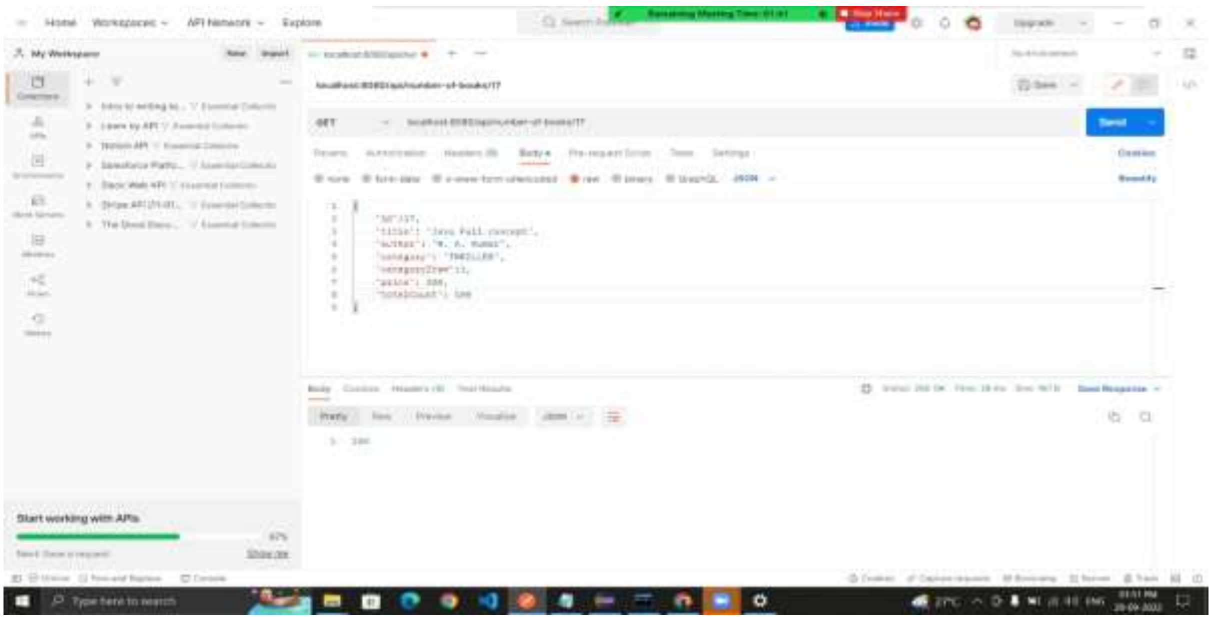
3. GET:

1. Get All Books



```
GET    localhost:8080/api/book-list

[
    {
        "id": [illegible],
        "title": "Introduction to Algorithms",
        "author": "Thomas H. Cormen",
        "category": "LITERATURE",
        "categoryItem": [illegible],
        "price": [illegible],
        "totalCount": [illegible]
    },
    {
        "id": [illegible],
        "title": "Getting Started Java",
        "author": "[illegible] Kumar",
        "category": "NONFICTION",
        "categoryItem": [illegible],
        "price": [illegible],
        "totalCount": [illegible]
    },
    {
        "id": [illegible],
        "title": "Full [illegible] Java",
        "author": "Lalit Mehta",
        "category": "[illegible]",
        "categoryItem": [illegible],
        "price": [illegible],
        "totalCount": [illegible]
```

2. Get number of books available by id

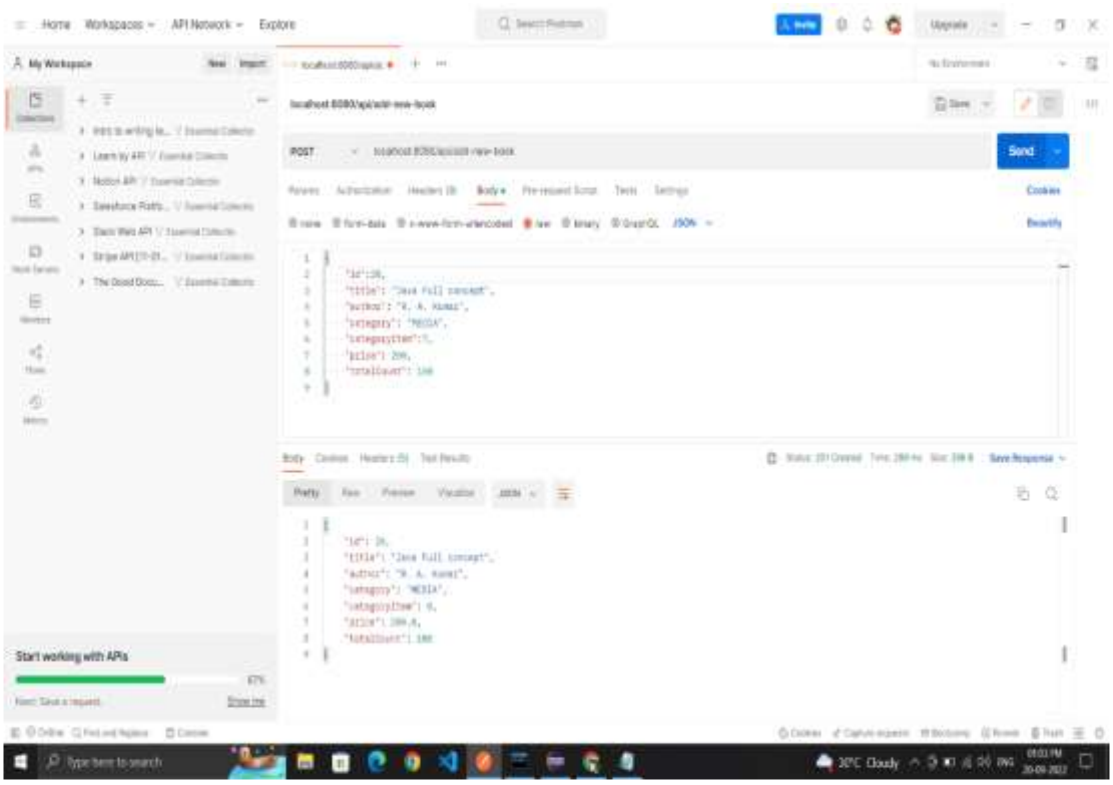


```
GET    localhost:8080/api/number-of-books/17

{
    "id":17,
    "title": "Java Full concept",
    "author": "[illegible]",
    "category": "THRILLER",
    "categoryItem":11,
    "price": [illegible],
    "totalCount": [illegible]
}
```

```
[illegible]
```

4. DELETE:

Delete the book by using id.



localhost:8080/api/add-new-book

POST localhost:8080/api/add-new-book

```
{
    "id":28,
    "title": "Java Full concept",
    "author": "R. A. Kumar",
    "category": "MEDIA",
    "categoryItem":1,
    "price": 200,
    "totalCount": 100
}
```

Status: 201 Created

```
{
    "id": 28,
    "title": "Java Full concept",
    "author": "R. A. Kumar",
    "category": "MEDIA",
    "categoryItem": 0,
    "price": 200.0,
    "totalCount": 100
}
```

8. Conclusion

The switch from written books being from bookstores to being ordered online or even just digital copies has had profound effects on the industry including bookstores and libraries and the general people of the world.

The positives include easy access for everyone and cheaper books along with saving natural resources. The negatives however are much greater and cannot be ignored.

9. References

J. Howell. Number of connecte bookstore devices will surge to 125 billion by 2030, ihs markit says - ihs technology. [Online]. Available: https://technology.ihs.com/596542/, last accessed: 11/07/2018.
E. Borgia, ”The internet of things vision: Key features, applications and open issues,” Computer Communications, vol. 54, pp. 1-31, 2014.
F. Restuccia, S. D’Oro, and T. Melodia, ”Securing the internet of things: New perspectives and research challenges,” IEEE Internet of Things Journal, vol. 1, no. 1, pp. 1-14, 2018.
J. A. Stankovic, ”Research directions for the internet of things,” IEEE Internet of Things Journal, vol. 1, no. 1, pp. 3-9, 2014.
M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis et al., ”Understanding the mirai botnet;’ in USENIX Security Symposium, 2017, pp. 1092-1110
B. B. Zarpelao, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, ”A survey of intrusion detection in internet of things;’ Journal of Network and Computer Applications, vol. 84, pp. 25-37, 2017.
B. B. Zarpelao, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, ”A survey of intrusion detection in internet of things;’ Journal of Network and Computer Applications, vol. 84, pp. 25-37, 2017.
J. Santos, P. Leroux, T. Wauters, B. Volckaert, and F. D. Turck, ”Anomaly detection for smart city applications over 5g low power wide area networks,” in NOMS 2018 - 2018

IEEE/IFIP Network Operations and Management Symposium, 2018, pp. 1-9.
A. Yousefpour, G. Ishigaki, and J. P. Jue, ”Fog computing: Towards minimizing delay in the internet of things,” in Edge Computing (EDGE), 2017
A. Abeshu and N. Chilarnkurti, ”Deep learning: the frontier for distributed attack detection in fog-to-things computing,” IEEE Communications Magazine, vol. 56, no. 2, pp. 169-175, 2018.
R. Roman, J. Zhou, and J. Lopez, ”On the features and challenges of security and privacy in distributed internet of things,” Computer Networks, vol. 57, no. 10, pp. 2266-2279, 2013.