Assignment No. 1

Name - Aditya Pawar

USN - 72233061J

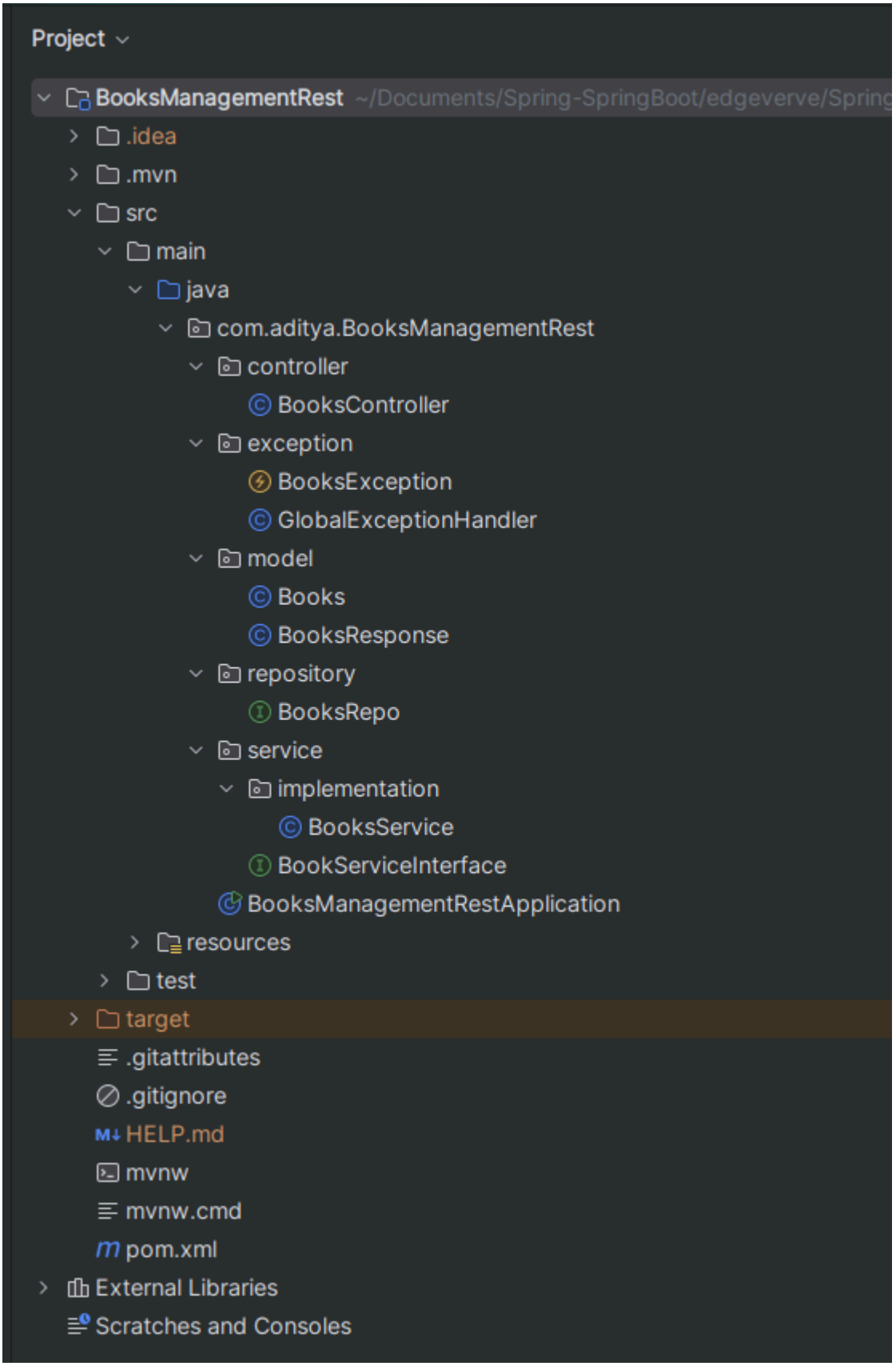
Assignment on Spring REST

Question 1. Create a RESTful web service using Spring Boot that manages a collection of "Books" with basic CRUD (Create, Read, Update, Delete) operations.

Implement a REST API with the following endpoints:

- **GET /books → Retrieve all books**
- **GET /books/{id} → Retrieve a specific book by ID**
- **POST /books → Add a new book**
- **PUT /books/{id} → Update an existing book**
- **DELETE /books/{id} → Delete a book**

Folder Structure:


Project
BooksManagementRest ~/Documents/Spring-SpringBoot/edgeverve/Spring
.idea
.mvn
src
main
java
com.aditya.BooksManagementRest
controller
BooksController
exception
BooksException
GlobalExceptionHandler
model
Books
BooksResponse
repository
BooksRepo
service
implementation
BooksService
BookServiceInterface
BooksManagementRestApplication
resources
test
target
.gitattributes
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="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>3.5.3</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.aditya</groupId>
    <artifactId>BooksManagementRest</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>BooksManagementRest</name>
    <description>Demo project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>21</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-validation</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>org.postgresql</groupId>
      <artifactId>postgresql</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.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
```

```
            <annotationProcessorPaths>
              <path>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
              </path>
            </annotationProcessorPaths>
          </configuration>
        </plugin>
        <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>
```

Application.properties:

```
spring.application.name=BooksManagementRest

spring.datasource.username=postgres
spring.datasource.password=root
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/assignmentsRest
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDiale
spring.jpa.hibernate.ddl-auto=update
spring.data.jpa.repositories.bootstrap-mode=default
spring.jpa.defer-datasource-initialization=true
```

```
server.port=8080
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format-sql=true
```

Controller Package

BooksController.java

```
package com.aditya.BooksManagementRest.controller;

import com.aditya.BooksManagementRest.model.Books;
import com.aditya.BooksManagementRest.service.implementation.BooksServ
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api-books")
public class BooksController {

    BooksService booksService;

    public BooksController(BooksService booksService){
        this.booksService = booksService;
    }

    @GetMapping("/books")
    public List<Books> getBooks(){
        return booksService.getAllBooks();
    }

    @GetMapping("/books/{id}")
    public Books getBookById(@PathVariable("id") int bookId) {
        return booksService.getBookById(bookId);
    }
```

```
    @PostMapping("/books")
    public String addBook(@RequestBody Books book) {
        booksService.addBook(book);
        return "Book Added Successfully!!!";
    }

    @PutMapping("/books/{id}")
    public String updateBook(@RequestBody Books book, @PathVariable("id")
    int bookId) {
        book.setId(bookId);
        booksService.updateBook(book);
        return "Book Updated Successfully!!!";
    }

    @DeleteMapping("/books/{id}")
    public String deleteBook(@PathVariable ("id") int bookId) {
        booksService.deleteBook(bookId);
        return "Book Deleted Successfully!!!";
    }
}
```

Exception Package

BooksException.java

```
package com.aditya.BooksManagementRest.exception;

public class BooksException extends RuntimeException {

    public BooksException() {}

    public BooksException(int id) {
        super("Cannot find book with this Id: " + id);
    }
}
```

GlobalExceptionHandler.java

```
package com.aditya.BooksManagementRest.exception;

import com.aditya.BooksManagementRest.model.BooksResponse;
import org.springframework.core.annotation.Order;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.HttpStatusCode;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation
.ResponseEntityExceptionHandler;

@RestControllerAdvice(annotations = RestController.class)
@Order(1)
public class GlobalExceptionHandler extends ResponseEntityExceptionHandle

    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid(
            MethodArgumentNotValidException ex,
            HttpHeaders headers,
            HttpStatusCode status,
            WebRequest request
        ) {
        BooksResponse response = new BooksResponse();
//        response.setStatusCode(400);
//        response.setStatusMsg("Invalid Input: " + ex.getBindingResult()
//        .getFieldError().getDefaultMessage());
        response.setStatusMsg(ex.getBindingResult().toString());
        return new ResponseEntity<>(response, HttpStatus.BAD_REQUEST);
    }

    @ExceptionHandler
```

```
    public ResponseEntity<BooksResponse> exceptionHandler(BooksExceptior
    booksException) {
        BooksResponse response = new BooksResponse();
        response.setStatusCode(500);
        response.setStatusMsg("Book Not Found");
        return new ResponseEntity<BooksResponse>(response,
        HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

Model Package —>

Books.java

```
package com.aditya.BooksManagementRest.model;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
@Entity(name = "books")
public class Books {
    @Id
    private int id;
    private String name;
    private String author;
    private String publisher;
    private double price;
```

```
}
```

BooksResponse.java

```
package com.aditya.BooksManagementRest.model;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@AllArgsConstructor
@NoArgsConstructor
@Getter
@Setter
public class BooksResponse {
    private int statusCode;
    private String statusMsg;
    private Books book;
}
```

Repository Package —>

BooksRepo.java

```
package com.aditya.BooksManagementRest.repository;

import com.aditya.BooksManagementRest.model.Books;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface BooksRepo extends JpaRepository<Books, Integer> {
}
```

Service Package —>

BooksServiceInterface.java

```
package com.aditya.BooksManagementRest.service;

import com.aditya.BooksManagementRest.model.Books;

import java.util.List;

public interface BookServiceInterface {
    public String addBook(Books book);
    public String updateBook(Books book);
    public String deleteBook(int bookId);
    public Books getBookById(int bookId);
    public List<Books> getAllBooks();
}
```

BooksService.java

```
package com.aditya.BooksManagementRest.service.implementation;

import com.aditya.BooksManagementRest.model.Books;
import com.aditya.BooksManagementRest.repository.BooksRepo;
import com.aditya.BooksManagementRest.service.BookServiceInterface;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class BooksService implements BookServiceInterface {

    BooksRepo booksRepo;

    public BooksService(BooksRepo booksRepo) {
        this.booksRepo = booksRepo;
    }
```

```
    @Override
    public String addBook(Books book) {
        booksRepo.save(book);
        return "Book Added Successfully!!!";
    }

    @Override
    public String updateBook(Books book) {
        if (booksRepo.existsById(book.getId())) {
            booksRepo.save(book);
        }
        return "Book Updated Successfully!!!";
    }

    @Override
    public String deleteBook(int bookId) {
        booksRepo.deleteById(bookId);
        return "book deleted successfully!!!";
    }

    @Override
    public Books getBookById(int bookId) {
        return booksRepo.findById(bookId).orElse(null);
    }

    @Override
    public List<Books> getAllBooks() {
        return booksRepo.findAll();
    }
}
```

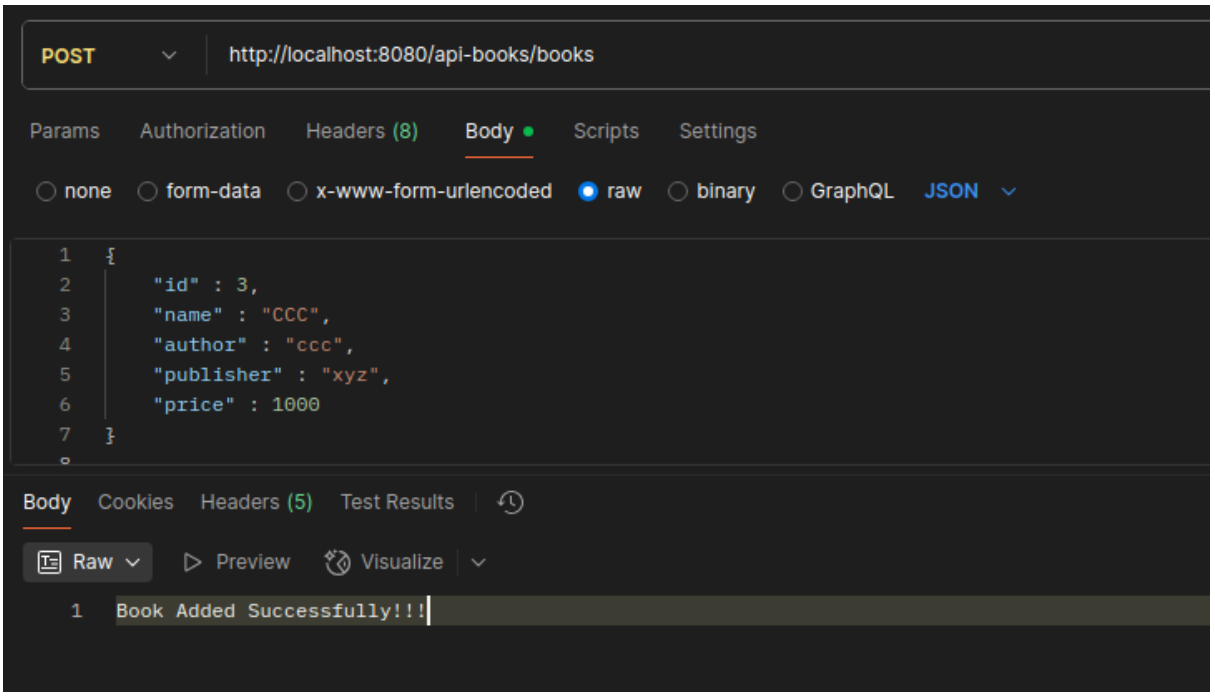
BooksManagementRestApplication.java

```
package com.aditya.BooksManagementRest;

import org.springframework.boot.SpringApplication;
```

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

@SpringBootApplication
public class BooksManagementRestApplication {

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

}
```

Output:

1. POST method:

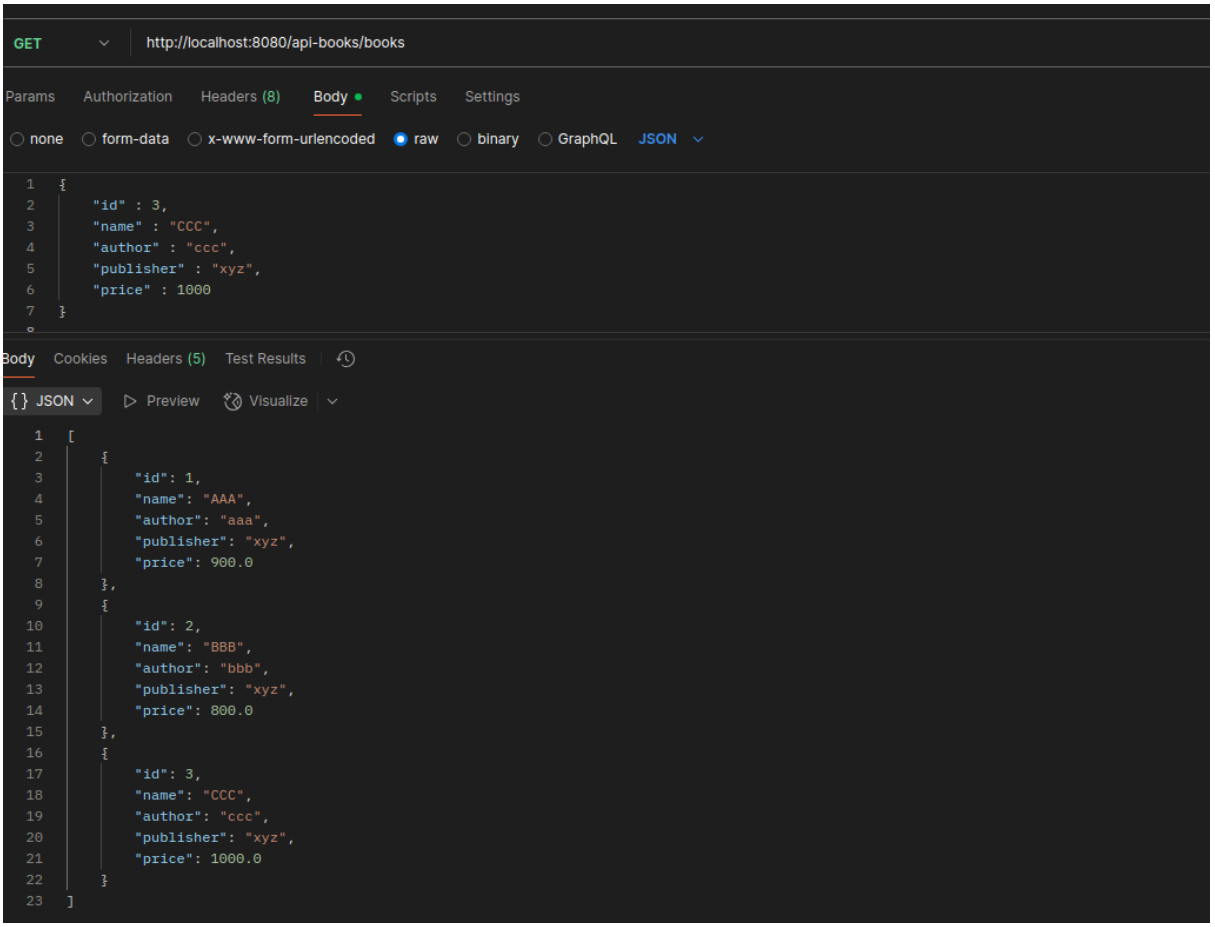


```
select * from books;
```

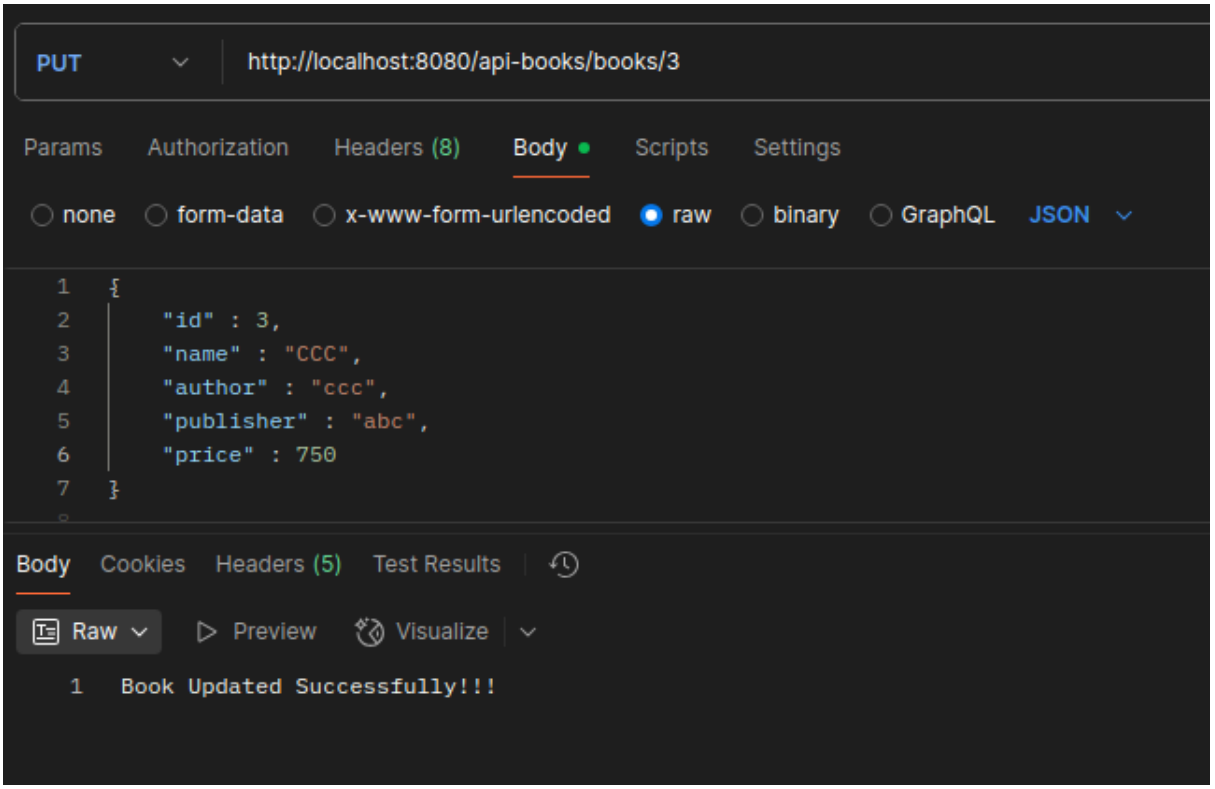
Data Output | Messages | Notifications

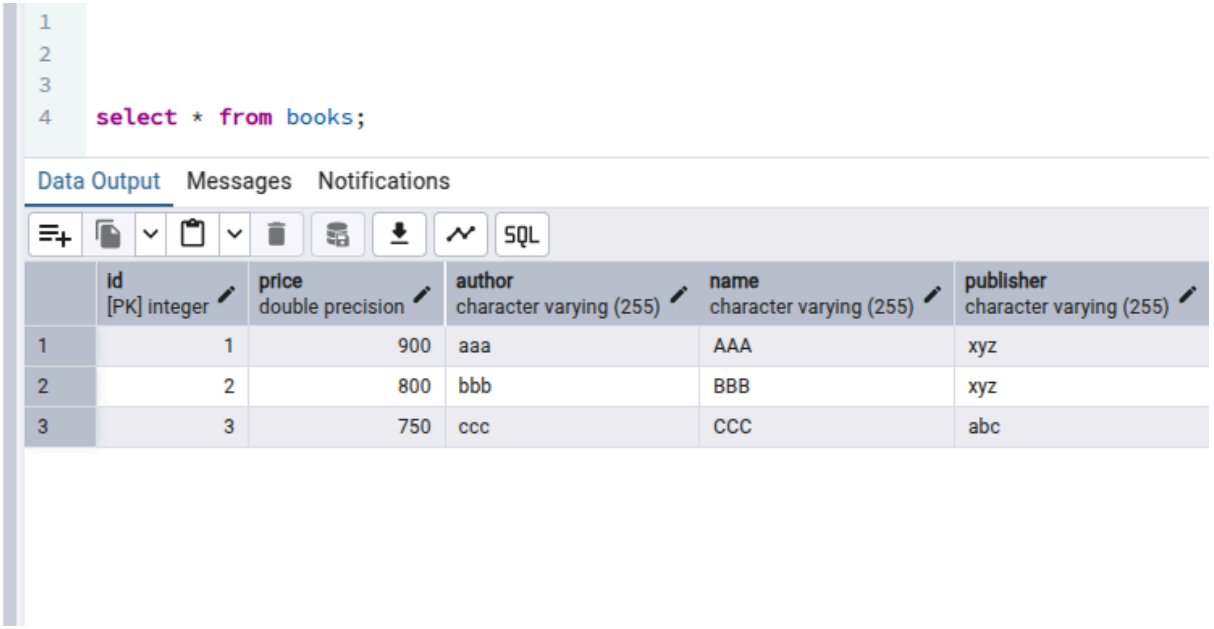
	id [PK] integer	price double precision	author character varying (255)	name character varying (255)	publisher character varying (255)
1	1	900	aaa	AAA	xyz
2	2	800	bbb	BBB	xyz
3	3	1000	ccc	CCC	xyz

2. GET Method:



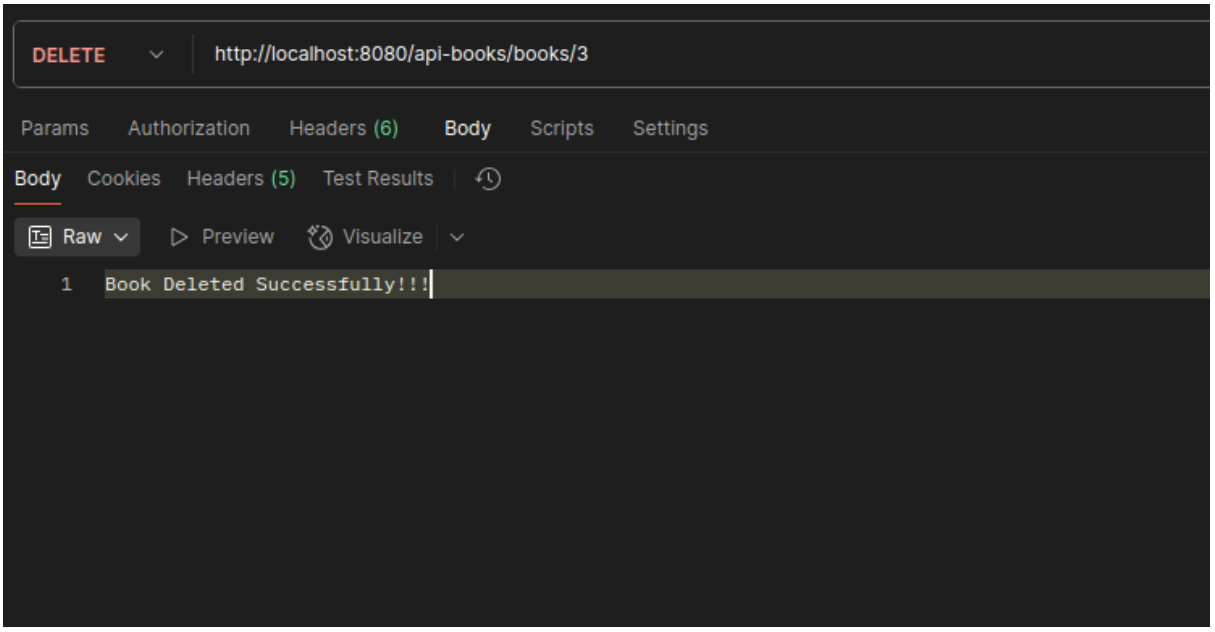
3. PUT Method:

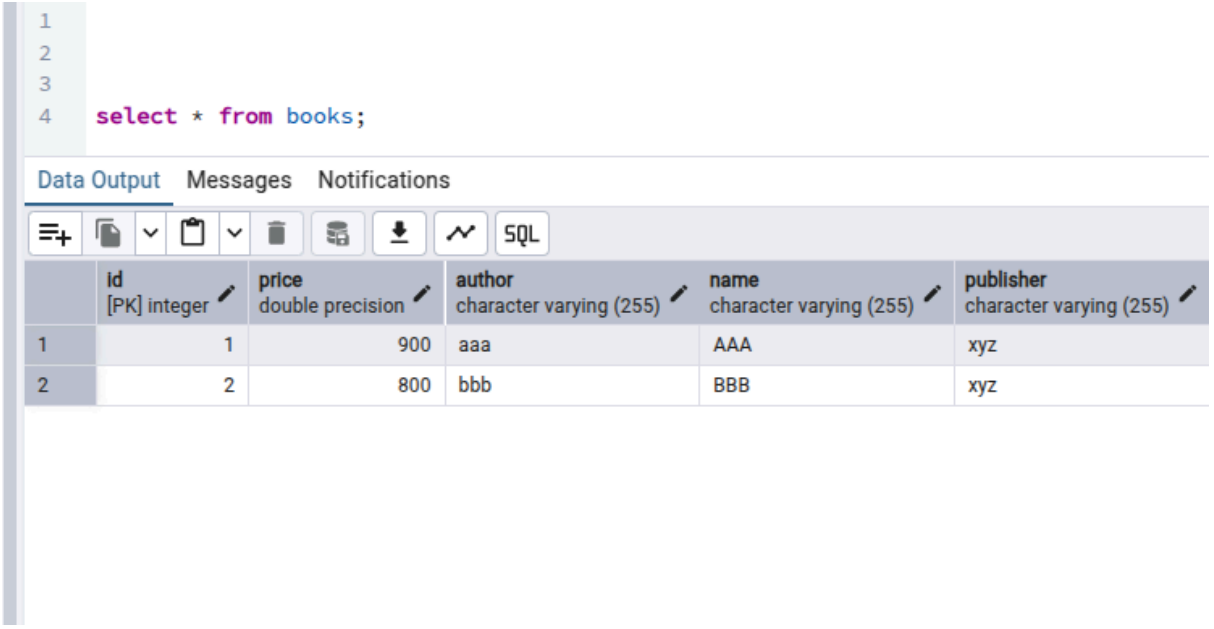




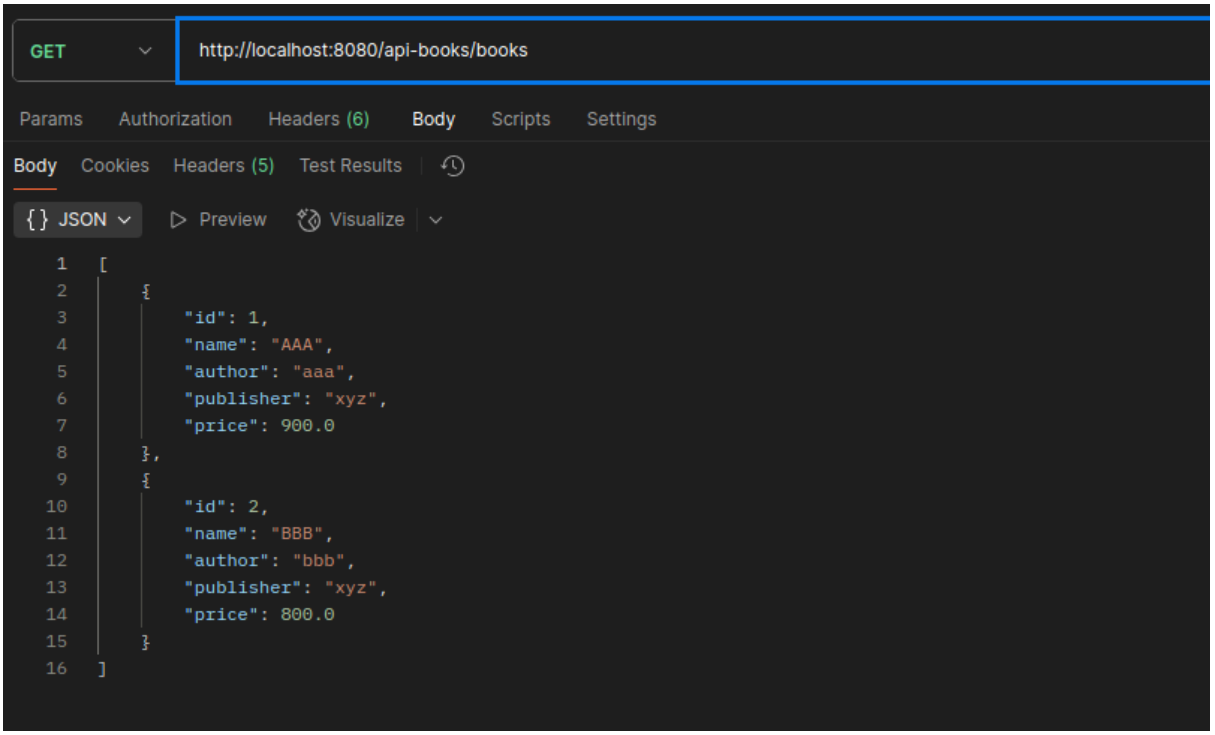
	id [PK] integer	price double precision	author character varying (255)	name character varying (255)	publisher character varying (255)
1	1	900	aaa	AAA	xyz
2	2	800	bbb	BBB	xyz
3	3	750	ccc	CCC	abc

4. DELETE Method:


DELETE
http://localhost:8080/api-books/books/3
Params
Authorization
Headers (6)
Body
Scripts
Settings
Body
Cookies
Headers (5)
Test Results
Raw
Preview
Visualize
1
Book Deleted Successfully!!!


1
2
3
4
select * from books;
Data Output
Messages
Notifications
SQL
id
[PK] integer
price
double precision
author
character varying (255)
name
character varying (255)
publisher
character varying (255)
1
1
900
aaa
AAA
xyz
2
2
800
bbb
BBB
xyz

Final Check:


GET
http://localhost:8080/api-books/books
Params
Authorization
Headers (6)
Body
Scripts
Settings
Body
Cookies
Headers (5)
Test Results
JSON
Preview
Visualize
[
{
"id": 1,
"name": "AAA",
"author": "aaa",
"publisher": "xyz",
"price": 900.0
},
{
"id": 2,
"name": "BBB",
"author": "bbb",
"publisher": "xyz",
"price": 800.0
}
]