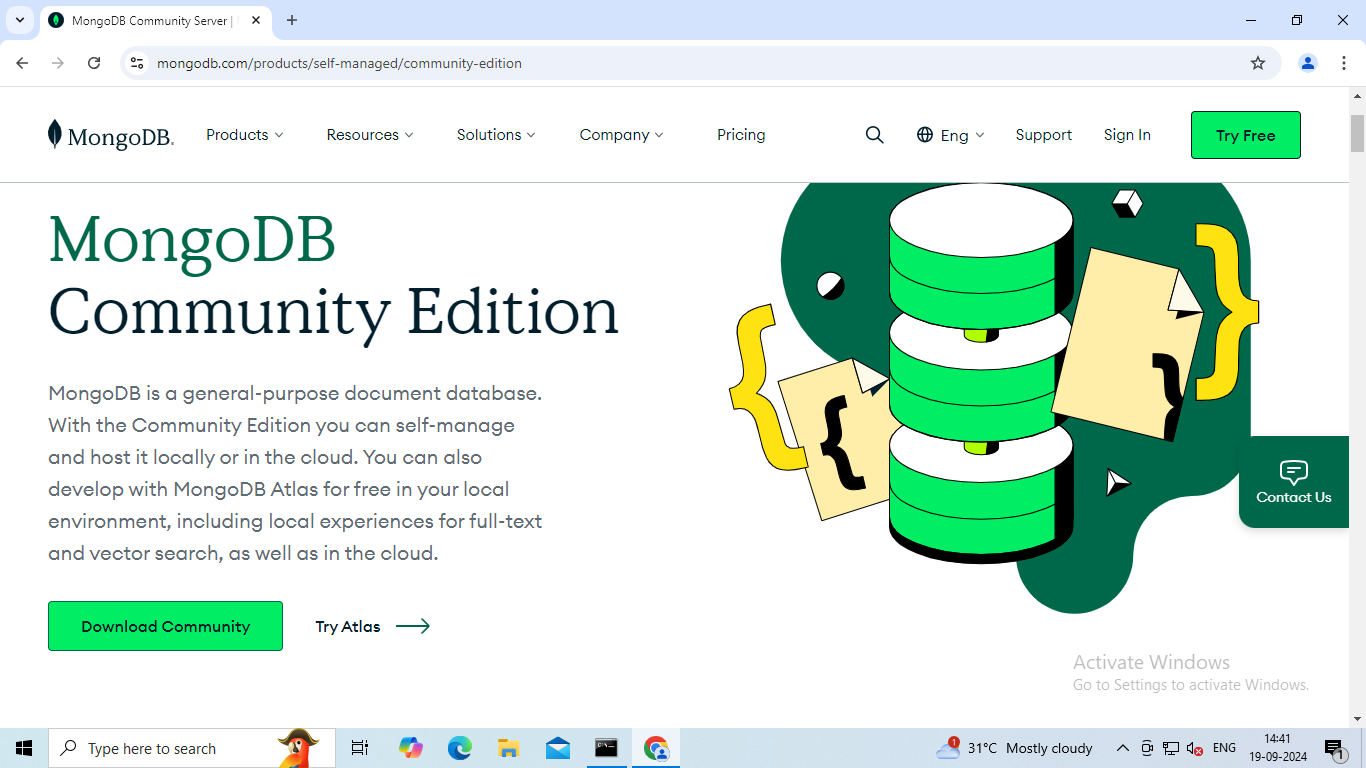
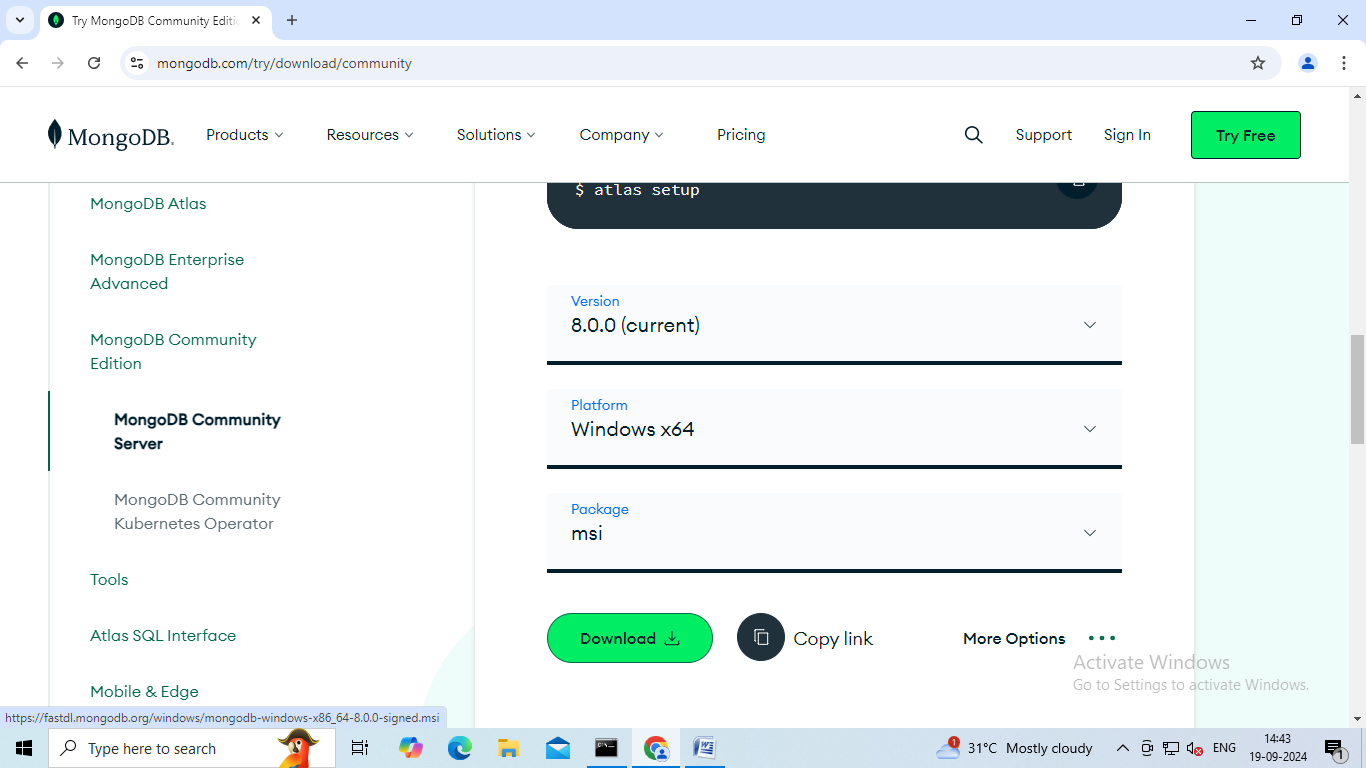
**6. Create REST Controller for CRUD operations using MongoDB and tests and test CRUD services.**

Step1: Download and install the MongoDB Community Edition and MongoDB compass.

Installation: Go to the browser search-MongoDB🡪Click on the products🡪Select Community Edition🡪Click on download.

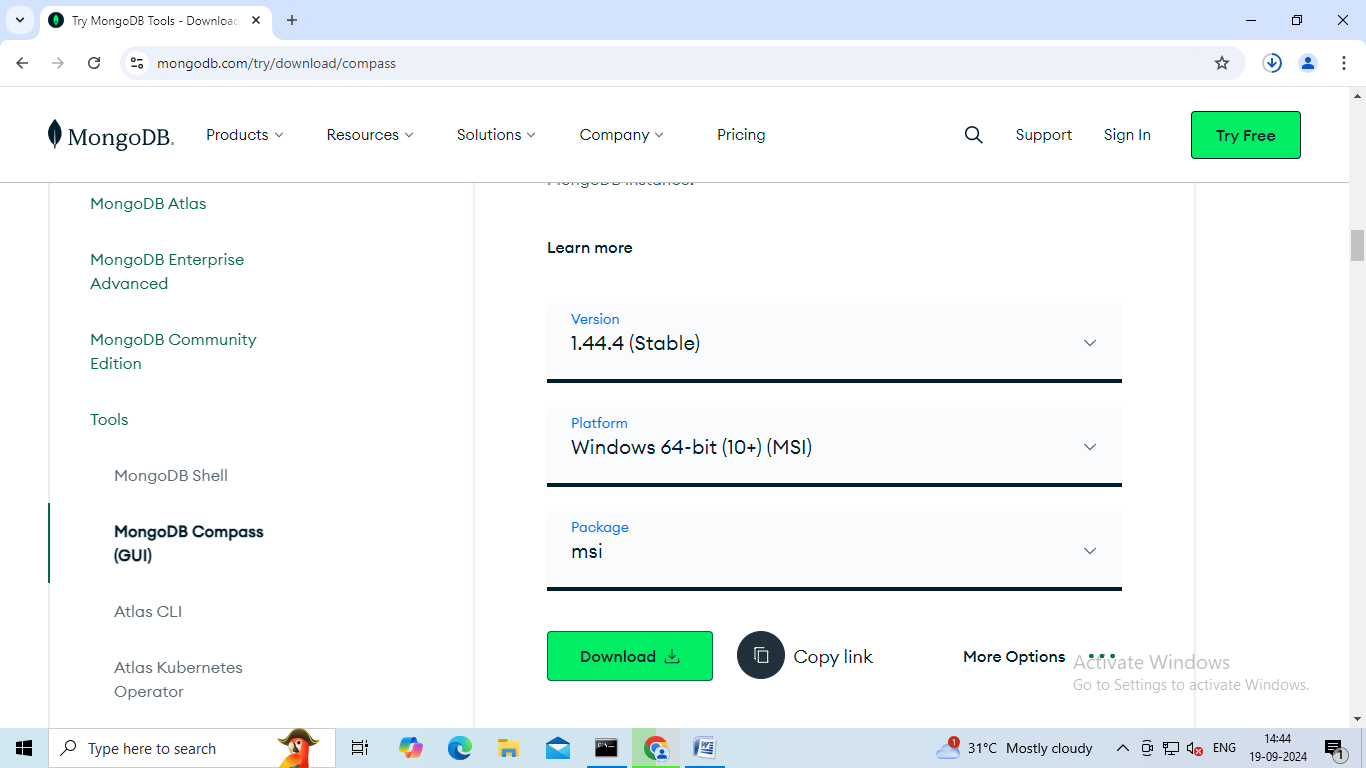


Next set the version, platform and package🡪Click on download.



Next click on the Tools🡪Select MongoDB Compass(GUI).

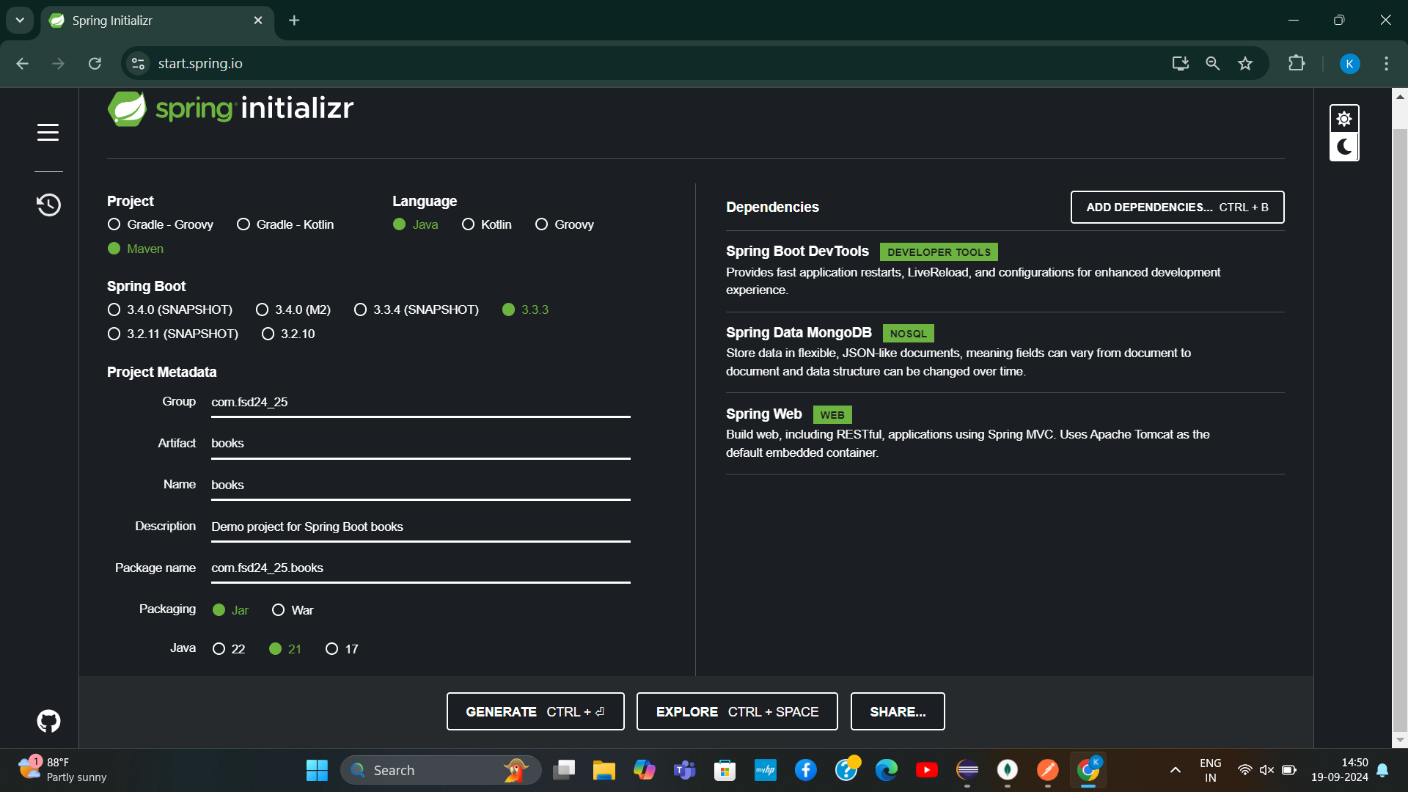
Set the version, platform and package🡪Click on download.



Step2: To set project configuration🡪Go to browser🡪Search **start.spring.io**

Step3: In spring initializr🡪Select Project-Maven🡪Language-Java🡪Spring Boot-3.3.3🡪 Group-com.fsd24\_25🡪Artifact-books🡪Description-Demo project for spring boot CrudMongo🡪Packaging-Jar🡪Java-21.

Dependencies🡪ADD DEPENDENCIES🡪Select-Spring Boot Dev Tools ,Spring Web and Spring Data MongoDB🡪 Click on Generate.

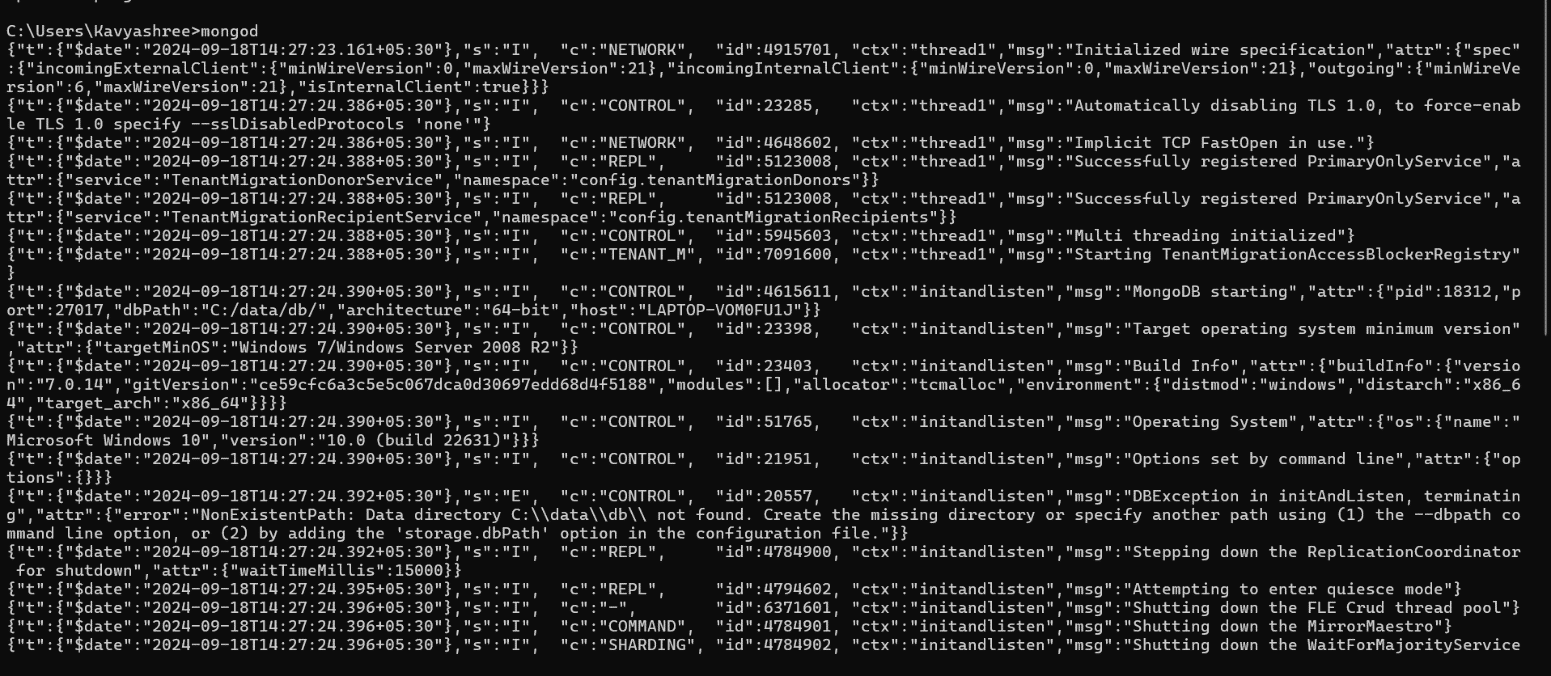


Step4: Next extract your zip file to workspace.

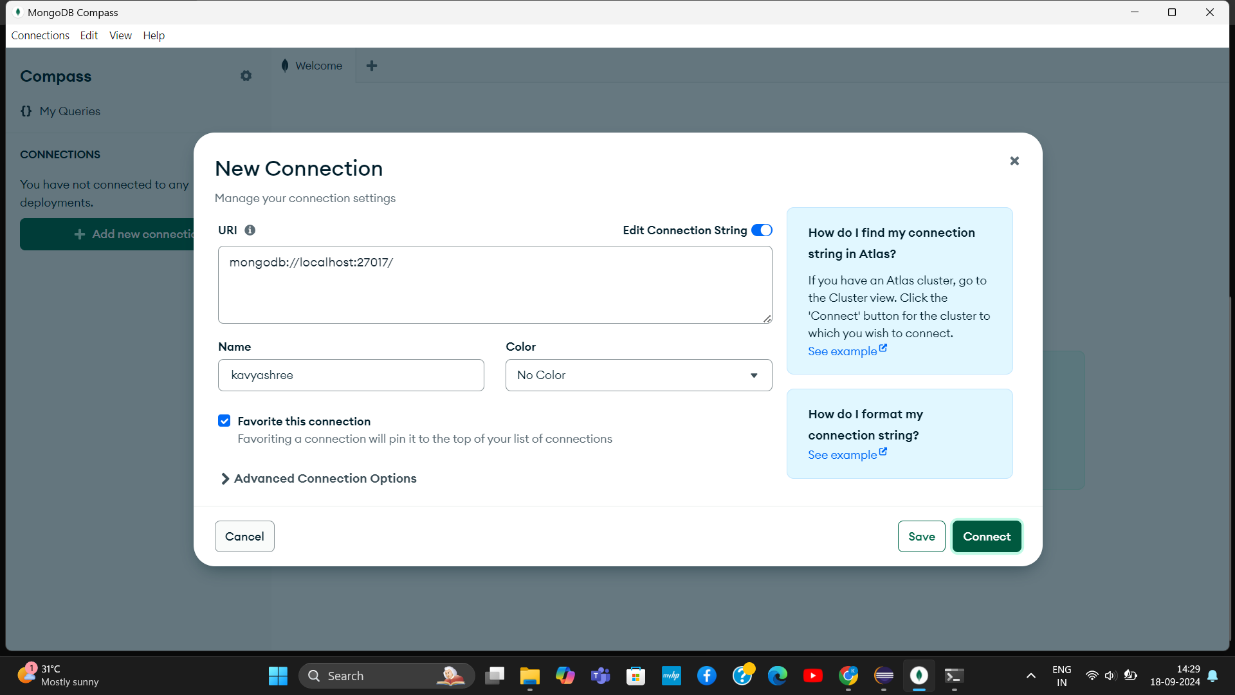
Step5: Select a destination.

Step6: Now set the mongodb sever path and run the server in command prompt.

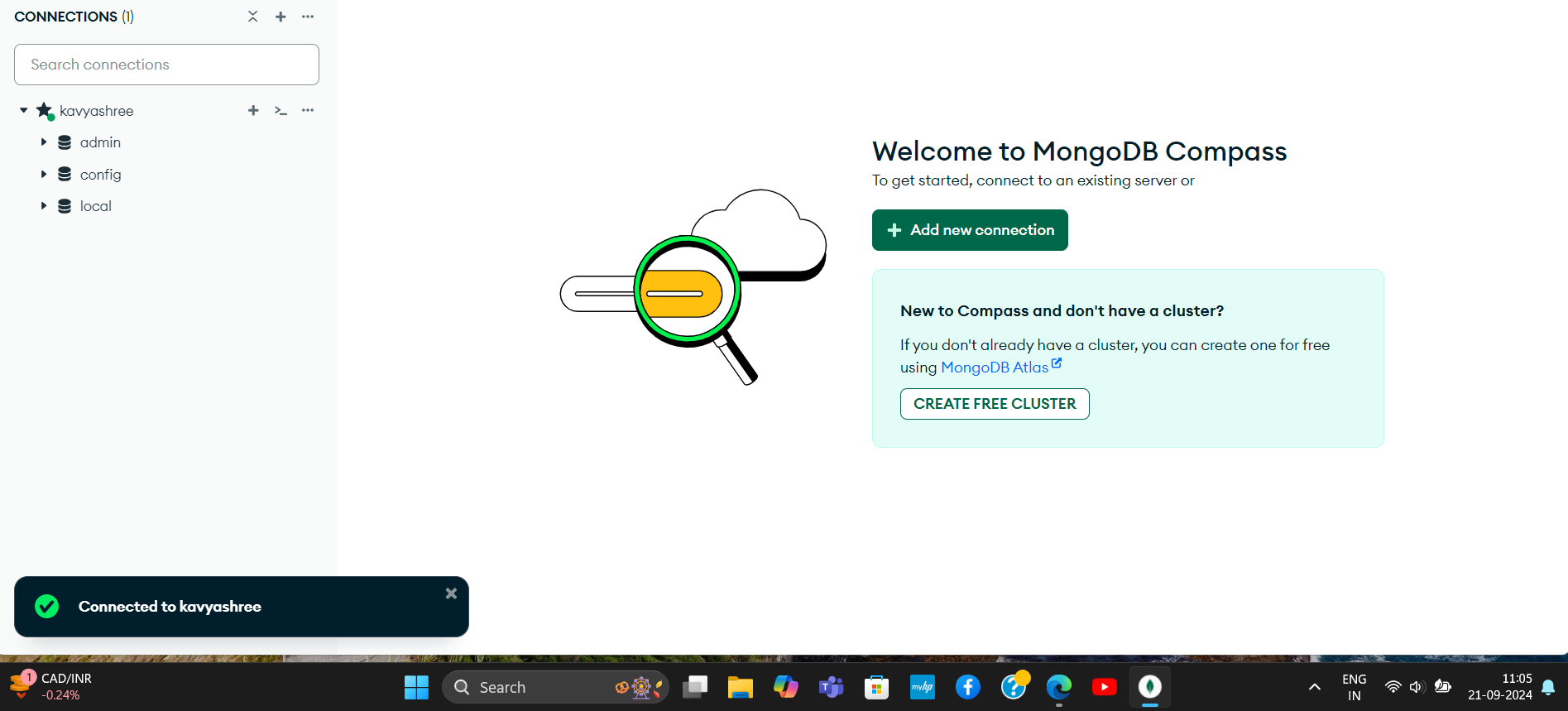
Give the command as🡪**mongod**



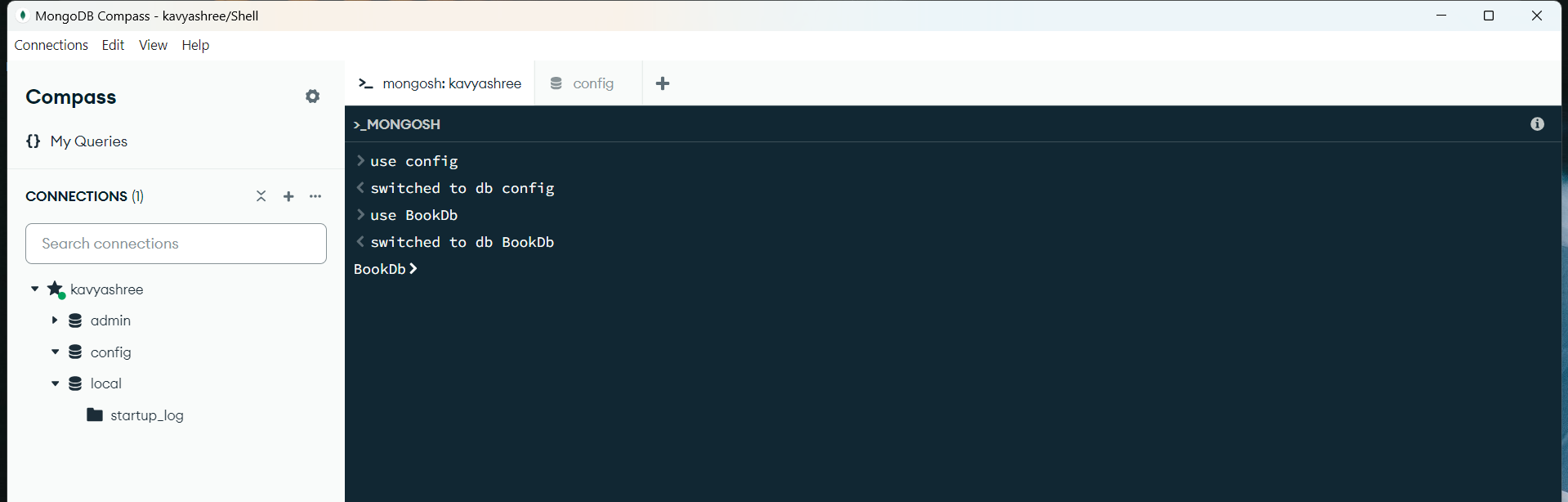
Step7: Open the MongoDB compass🡪click on + symbol to connect🡪Give your name🡪click on connect.



Now we can see our connections.

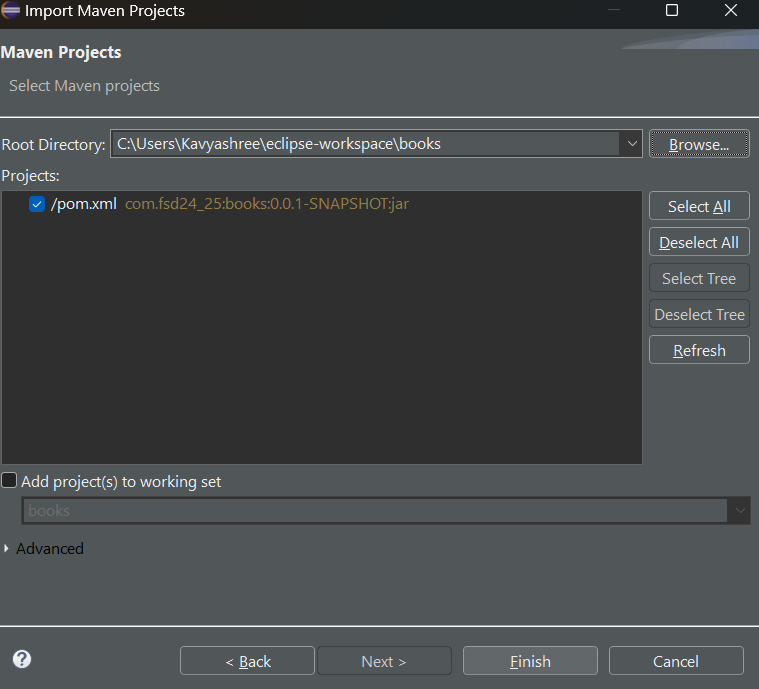


Click on local🡪open mongosh🡪Give the command to create a database-**use BookD**

****

Step8: Go to Eclipse🡪File🡪Import🡪Maven🡪Existing maven🡪Click on Browse and select the folder from where you extracted🡪Tick mark on pom.xml🡪Click on Finish.

Save the pom.xml



Step9: Create four packages:

com.fsd24\_25.books.Entity

com.fsd24\_25.books.Controller

com.fsd24\_25.books.Repository

com.fsd24\_25.books.Service

Step10: Create a books class inside com.fsd24\_25.books.Entity

**books.java:**

package com.fsd24\_25.books.Entity;

import org.springframework.data.annotation.Id;

import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection="books")

public class books {

@Id

private String \_id;

private String isbn;

private String bookName;

private String author;

private String publisher;

private float price;

public books () {}

public books(String isbn, String bookName, String author, String publisher, float price) {

super();

this.isbn = isbn;

this.bookName = bookName;

this.author = author;

this.publisher = publisher;

this.price = price;

}

public String getIsbn() {

return isbn;

}

public void setIsbn(String isbn) {

this.isbn = isbn;

}

public String getBookName() {

return bookName;

}

public void setBookName(String bookName) {

this.bookName = bookName;

}

public String getAuthor() {

return author;

}

public void setAuthor(String author) {

this.author = author;

}

public String getPublisher() {

return publisher;

}

public void setPublisher(String publisher) {

this.publisher = publisher;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

@Override

public String toString() {

return ("{\"isbn\":\"" + this.isbn + "\",\n\"bookName\":\"" + this.bookName + "\",\n\"author\":\"" + this.author + "\",\n\"publisher\":\"" + this.publisher + "\",\n\"price\":\"" + this.price + "\"\n}");

}

}

Step11: Create another class bookController inside com.fsd24\_25.books.Controller

**bookController.java:**

package com.fsd24\_25.books.Controller;

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

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

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

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

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

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

import com.fsd24\_25.books.Entity.books;

import com.fsd24\_25.books.Service.bookService;

@RestController

public class bookController {

@Autowired

private bookService s;

@GetMapping("/")

public ResponseEntity<?> disp() {

return s.load();

}

@GetMapping("/list")

public ResponseEntity<?>getbooks(){

return s.findAllBooks();

}

@PostMapping("/insert")

public ResponseEntity<?> insertBooks(@RequestBody books b ){

return s.insertbooks(b);

}

@PutMapping("/update")

public ResponseEntity<?> updateBooks(@RequestBody books b){

return s.updateBooks(b);

}

@DeleteMapping("del/{isbn}")

public ResponseEntity<?> deleteBooks(@PathVariable String isbn){

return s.deleteBooks(isbn);

}

}

Step12: Create another class bookRepo inside com.fsd24\_25.books.Repository

**bookRepo.java:**

package com.fsd24\_25.books.Repository;

import java.util.List;

import java.util.Optional;

import org.springframework.data.mongodb.repository.MongoRepository;

import org.springframework.stereotype.Repository;

import com.fsd24\_25.books.Entity.books;

*@Repository*

public interface bookRepo extends MongoRepository <com.fsd24\_25.books.Entity.books,String>{

public List<books>deleteByIsbn(String isbn);

public Optional<books> findByIsbn(String isbn);

public boolean existsByIsbn(String isbn);

}

Step13: Create another class bookService inside com.fsd24\_25.books.Service

**bookService.java:**

package com.fsd24\_25.books.Service;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Service;

import com.fsd24\_25.books.Entity.books;

import com.fsd24\_25.books.Repository.bookRepo;

import com.mongodb.MongoException;

*@Service*

public class bookService {

*@Autowired*

private bookRepo repo;

public ResponseEntity<?> load(){

return ResponseEntity.*ok*("Welcome to MongoDb Crud Demo...");

}

public ResponseEntity<?>findAllBooks(){

List<books> allbooks=new ArrayList<>();

allbooks=repo.findAll();

if(allbooks.isEmpty()) {

return ResponseEntity.*status*(*HttpStatus*.***NOT\_FOUND***).body("No books in Database");

}

else {

return ResponseEntity.*ok*(allbooks);

}

}

public ResponseEntity<?> insertbooks(books b) {

List<books> booklist=new ArrayList<books>();

String err\_msg="";

try {

repo.save(b);

}

catch(MongoException e) {

err\_msg=e.getMessage();

return ResponseEntity.*internalServerError*().body(err\_msg);

}

booklist=repo.findAll();

if (booklist.isEmpty()){

return ResponseEntity.*status*(*HttpStatus*.***NOT\_FOUND***).body("No Books Found");

}

else {

return ResponseEntity.*ok*(booklist);

}

}

public ResponseEntity<?> updateBooks(books b){

Optional<books> existingbook=repo.findByIsbn(b.getIsbn());

List<books> updatedbooklist=new ArrayList<>();

if (!existingbook.isPresent()) {

return ResponseEntity.*status*(*HttpStatus*.***NOT\_FOUND***).body("Specified book with ISBN : "+b.getIsbn()+" not found.");

}

else {

try{

repo.deleteByIsbn(b.getIsbn());

repo.save(b);

updatedbooklist=repo.findAll();

return ResponseEntity.*ok*(updatedbooklist);

}

catch (MongoException e) {

return ResponseEntity.*internalServerError*().body(e.getMessage());

}

}

}

public ResponseEntity<?> deleteBooks(String isbn){

List<books> newbooklist=new ArrayList<>();

if (repo.existsByIsbn(isbn)) {

try{

repo.deleteByIsbn(isbn);

newbooklist=repo.findAll();

return ResponseEntity.*ok*(newbooklist);

}

catch (MongoException e) {

return ResponseEntity.*internalServerError*().body(e.getMessage());

}

}

else {

return ResponseEntity.*status*(*HttpStatus*.***NOT\_FOUND***).body("Specified book with ISBN : "+isbn+" not found.");

}

}

}

Step14: Next write the following code in src/main/resources

**Application.properties:**

spring.application.name=books

spring.data.mongodb.host=localhost

spring.data.mongodb.port=27017

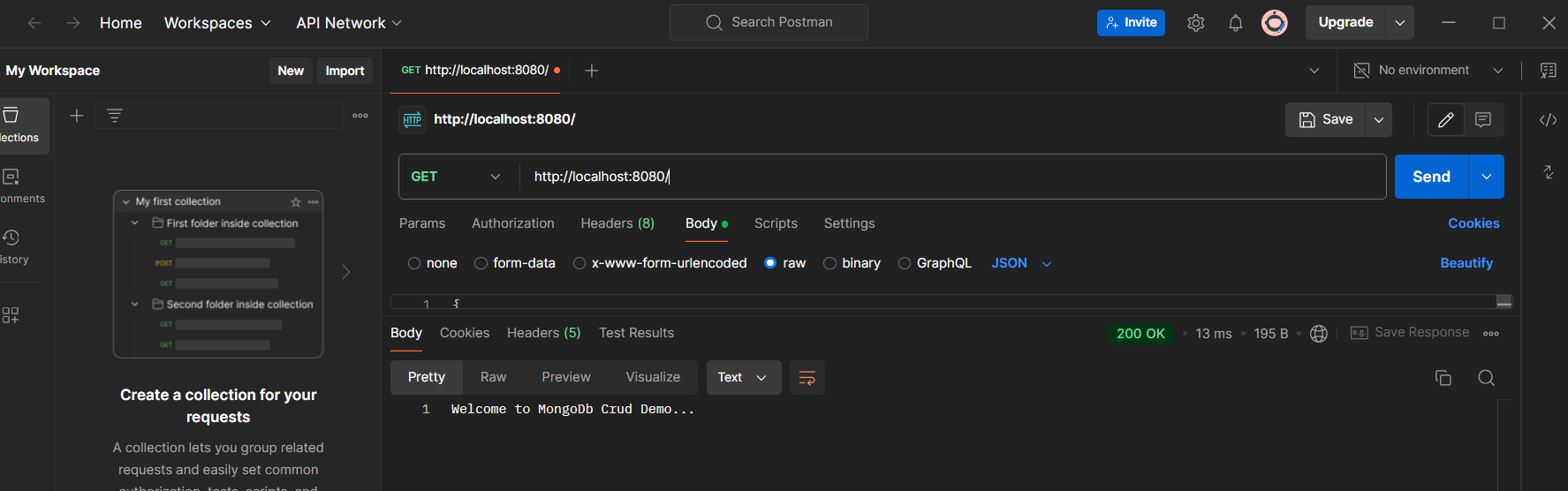
spring.data.mongodb.database=bookdb

**Now run the code:**

Right click to your project🡪Run as🡪Java Application🡪Search your project🡪Click on OK.

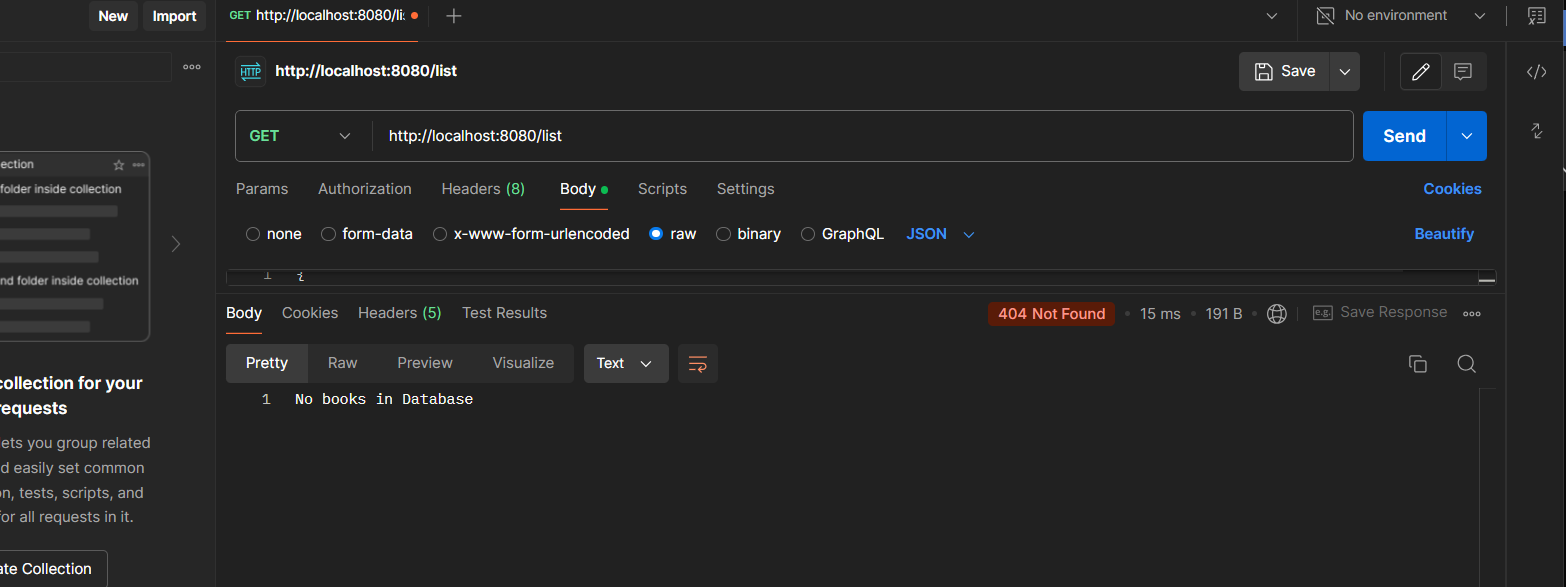
Go to the POSTMAN🡪Set the request GET🡪Enter the URL as

http://localhost:8080/



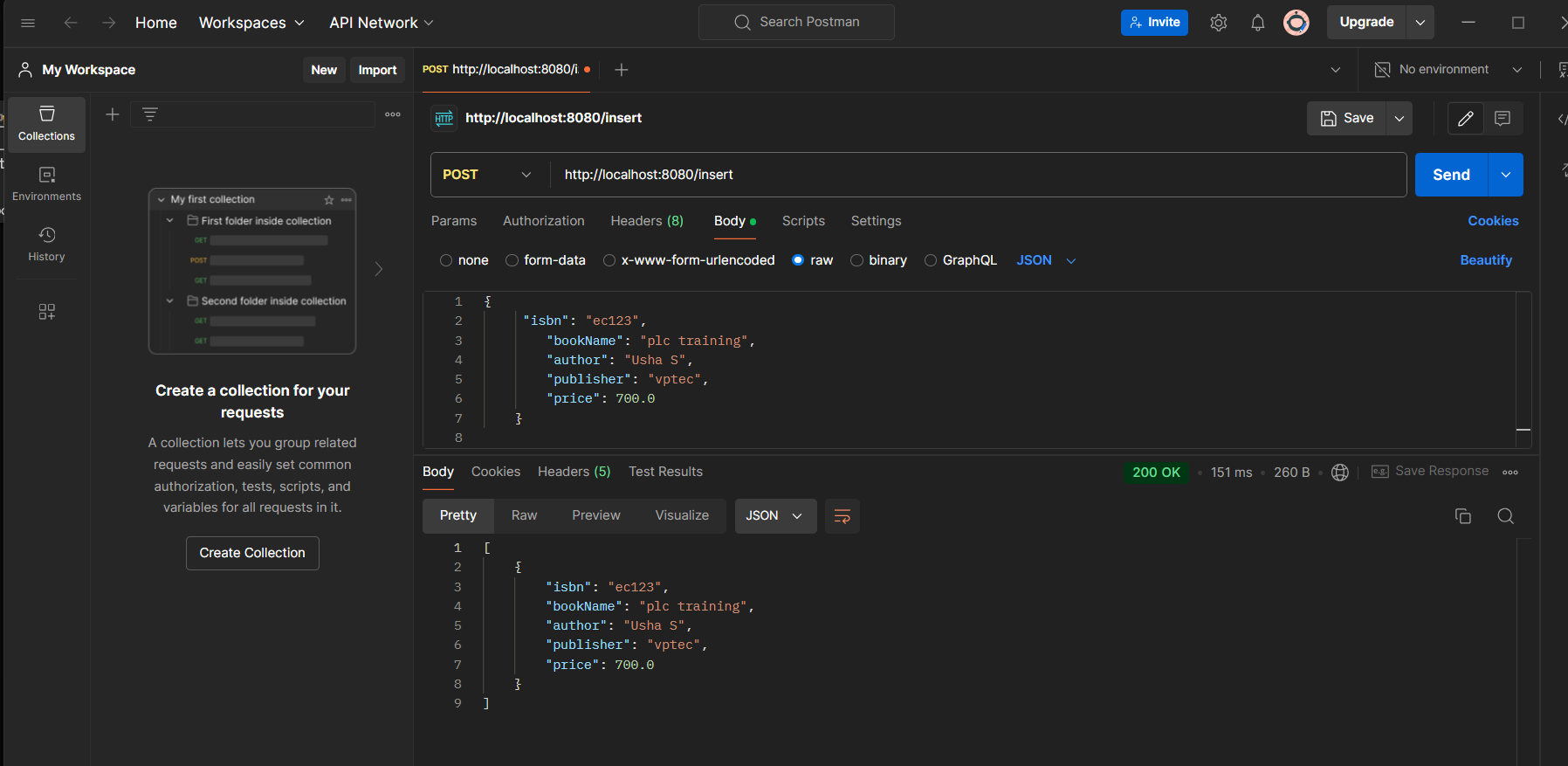
Next to get the book details set the request to GET🡪set the url as

<http://localhost:8080/list>

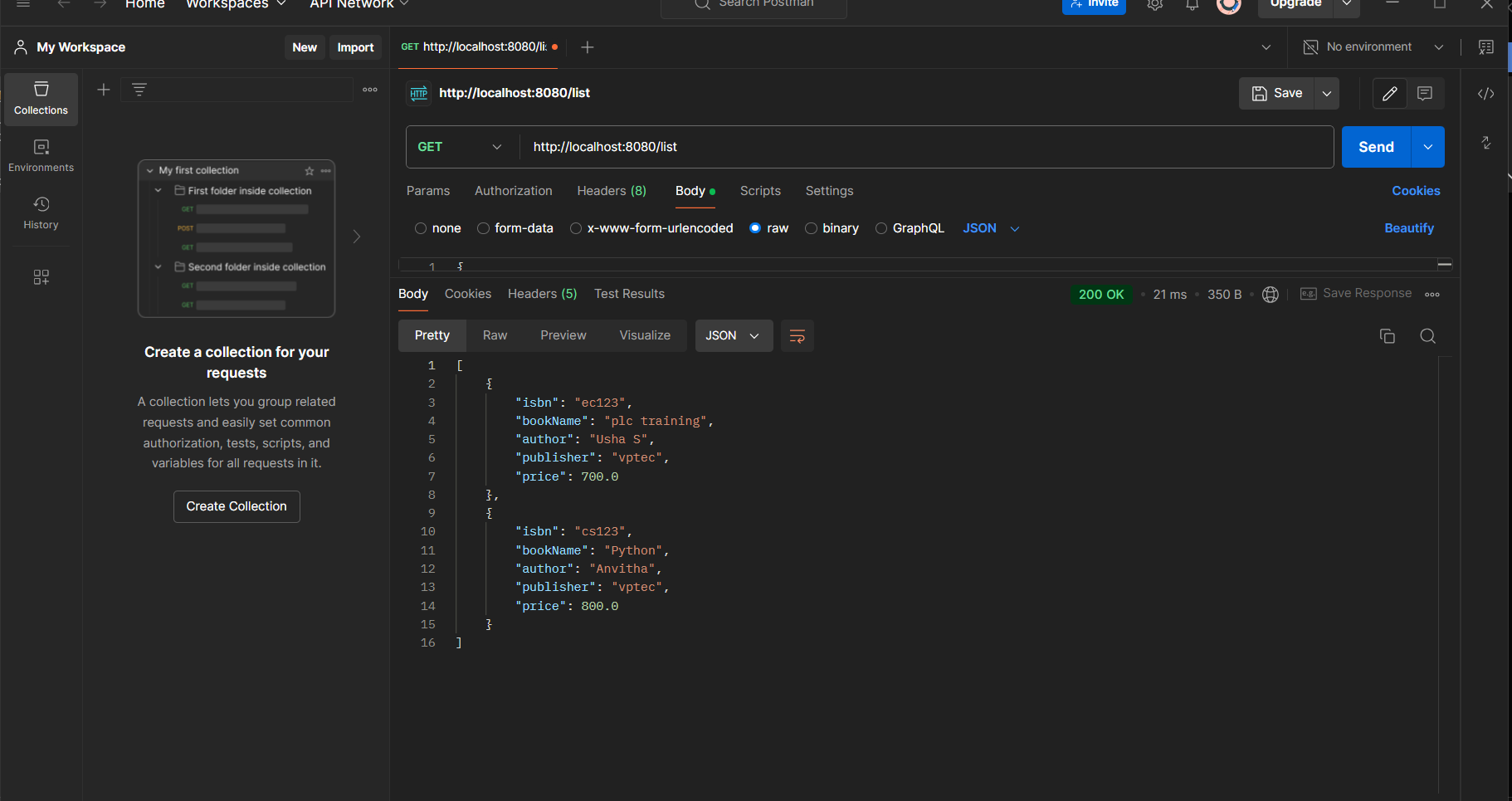


Next to insert the book details set the request to POST🡪set the url as

<http://localhost:8080/insert>

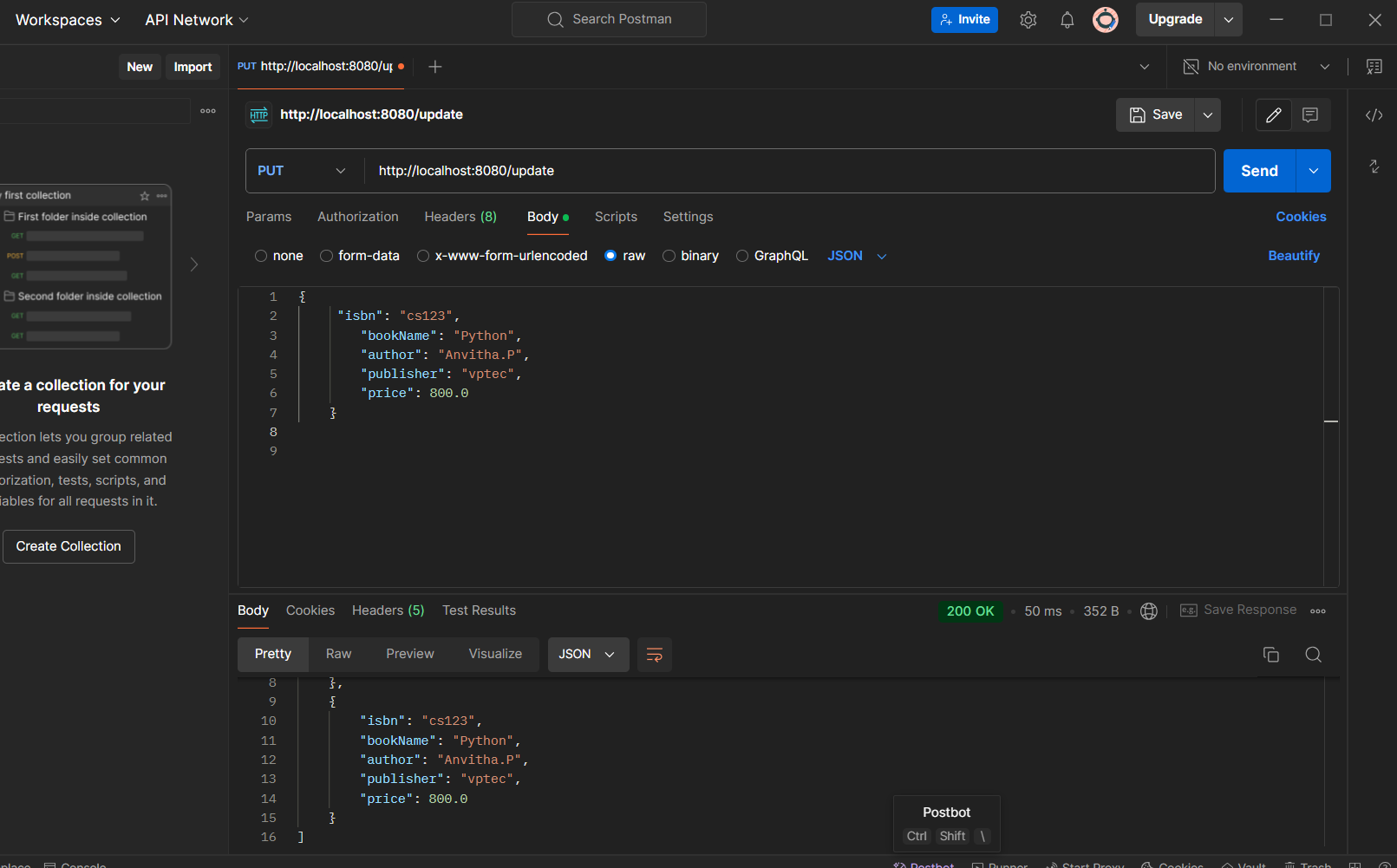


Now we can our inserted details by entering <http://localhost:8080/list>



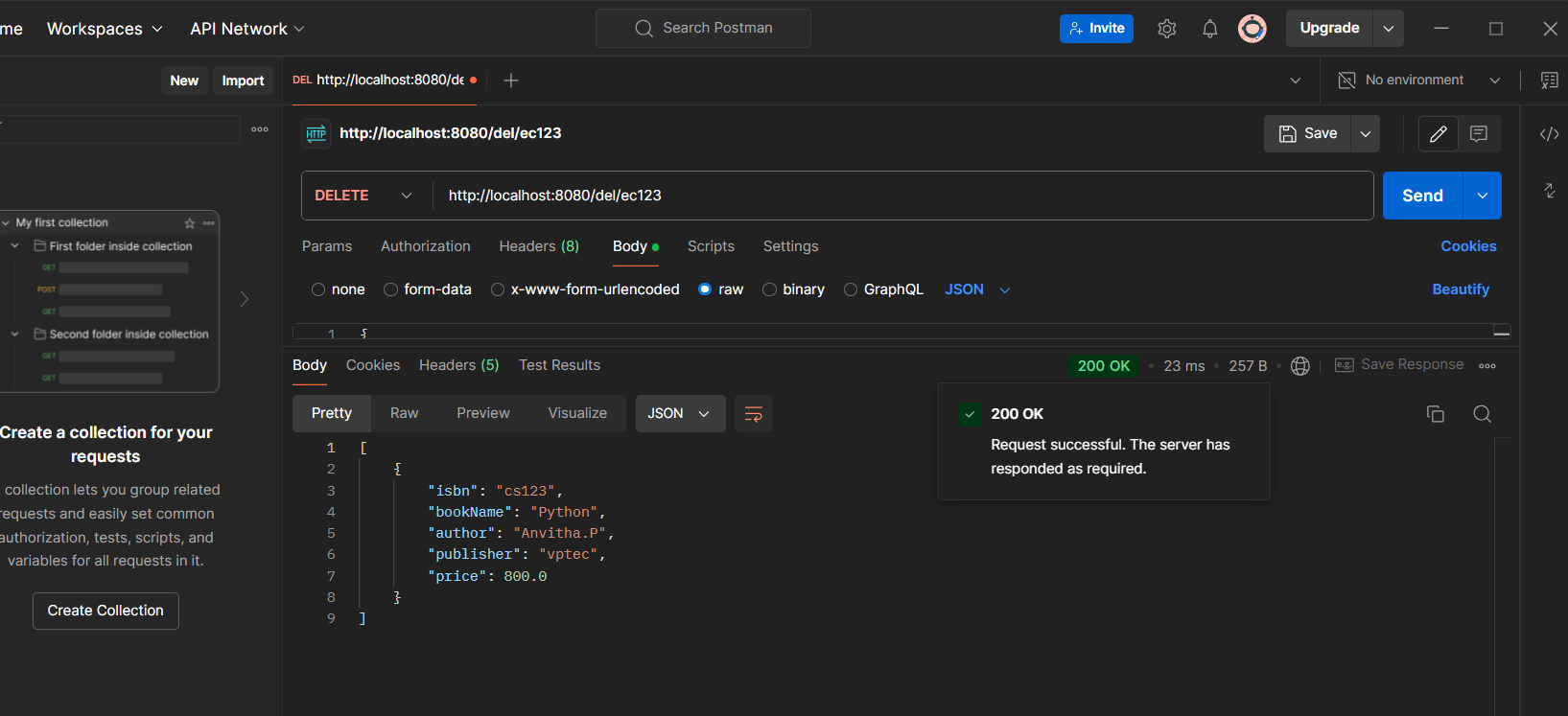
Next to update the book details set the request to PUT🡪set the url as

<http://localhost:8080/update>

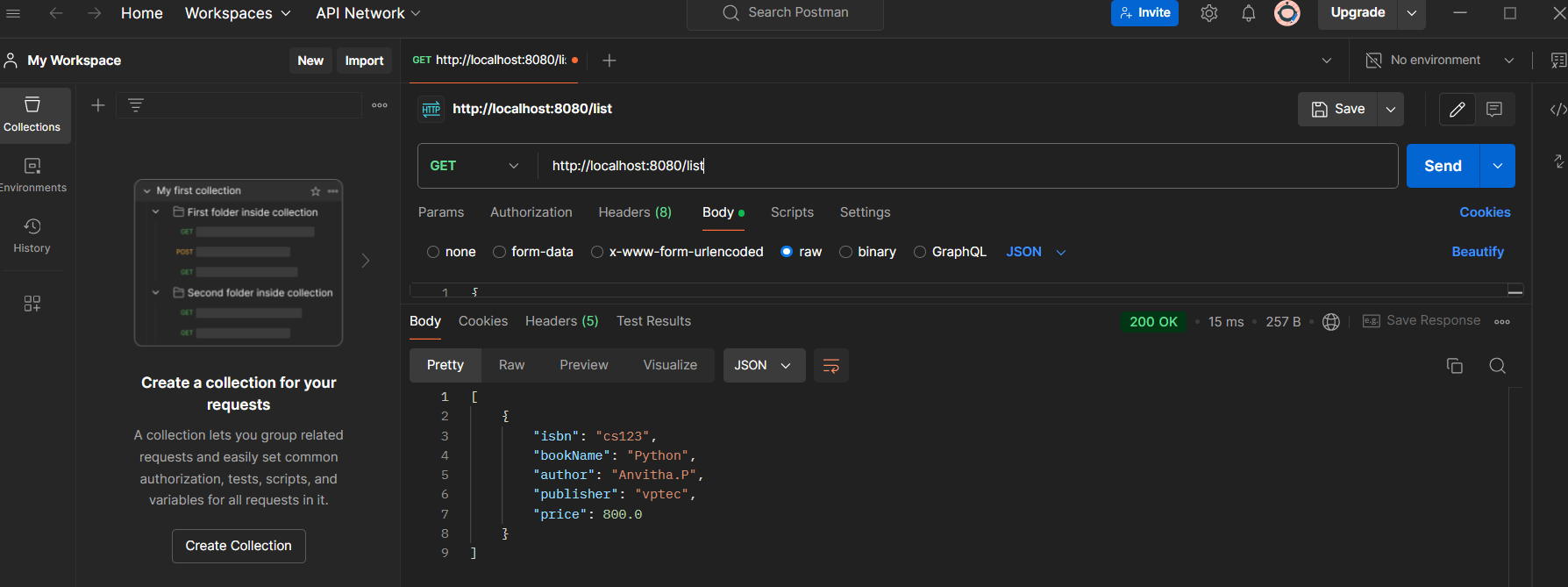


Next to delete the book details set the request to DELETE🡪set the url as

<http://localhost:8080/del/ec123>



Now we can book details by entering <http://localhost:8080/list>



**Now write the tests to test CRUD Services:**

**Src/test/java:**

**BooksApplicatiom.java:**

package com.fsd24\_25.books;

import java.util.List;

import java.util.Optional;

import org.assertj.core.api.SoftAssertions;

import org.junit.jupiter.api.Assertions;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.http.ResponseEntity;

import com.fsd24\_25.books.Entity.books;

import com.fsd24\_25.books.Repository.bookRepo;

import com.fsd24\_25.books.Service.bookService;

@SpringBootTest

class BooksApplicationTests {

@Autowired

private bookRepo r;

@Autowired

private bookService s;

@Test

void contextLoads() {

ResponseEntity<?> basic\_response = s.load();

Assertions.assertEquals(basic\_response.getBody(), "Welcome to MongoDb Crud Demo...");

}

@Test

void testInsert() {

books newBook = new books();

newBook.setIsbn("20EC51I");

newBook.setBookName("PLC Training Kit");

newBook.setAuthor("Usha Kiran");

newBook.setPublisher("VPT Publications");

newBook.setPrice(500);

s.insertbooks(newBook);

books insertedBook = r.findByIsbn("20EC51I").get();

SoftAssertions checks = new SoftAssertions();

checks.assertThat(insertedBook.getIsbn()).isEqualTo("20EC51I").withFailMessage("isbn mismatch");

checks.assertThat(insertedBook.getBookName()).isEqualTo("PLC Training Kit").withFailMessage("bookName mismatch");

checks.assertThat(insertedBook.getAuthor()).isEqualTo("Usha Kiran").withFailMessage("author mismatch");

checks.assertThat(insertedBook.getPublisher()).isEqualTo("VPT Publications").withFailMessage("publisher mismatch");

checks.assertThat(insertedBook.getPrice()).isEqualTo(500).withFailMessage("price mismatch");

checks.assertAll();

r.deleteByIsbn("20EC51I");

}

@Test

void testUpdate() {

books newBook = new books("20EC51I", "PLC Training Kit", "Usha Kiran", "VPT Publications", 500);

r.save(newBook);

books updatedBook = new books("20EC51I", "PLC And PCB", "Usha Kiran B", "VPT EC Publications", 550);

s.updateBooks(updatedBook);

books retrievedBook = r.findByIsbn("20EC51I").get();

SoftAssertions checks = new SoftAssertions();

checks.assertThat(retrievedBook.getIsbn()).isEqualTo("20EC51I").withFailMessage("isbn mismatch");

checks.assertThat(retrievedBook.getBookName()).isEqualTo("PLC And PCB").withFailMessage("bookName not updated");

checks.assertThat(retrievedBook.getAuthor()).isEqualTo("Usha Kiran B").withFailMessage("author not updated");

checks.assertThat(retrievedBook.getPublisher()).isEqualTo("VPT EC Publications").withFailMessage("publisher not updated");

checks.assertThat(retrievedBook.getPrice()).isEqualTo(550).withFailMessage("price not updated");

checks.assertAll();

r.deleteByIsbn("20EC51I");

}

@Test

void testDelete() {

books newBook = new books("20EC51I", "PLC Training Kit", "Usha Kiran", "VPT Publications", 500);

r.save(newBook);

s.deleteBooks("20EC51I");

Assertions.assertEquals(Optional.empty(), r.findByIsbn("20EC51I"));

}

@Test

void testRead() {

int countBefore = r.findAll().size();

books newBook = new books("20EC51I", "PLC Training Kit", "Usha Kiran", "VPT Publications", 500);

r.save(newBook);

List<books> bookList = r.findAll();

int countAfter = bookList.size();

Assertions.assertEquals(countBefore + 1, countAfter);

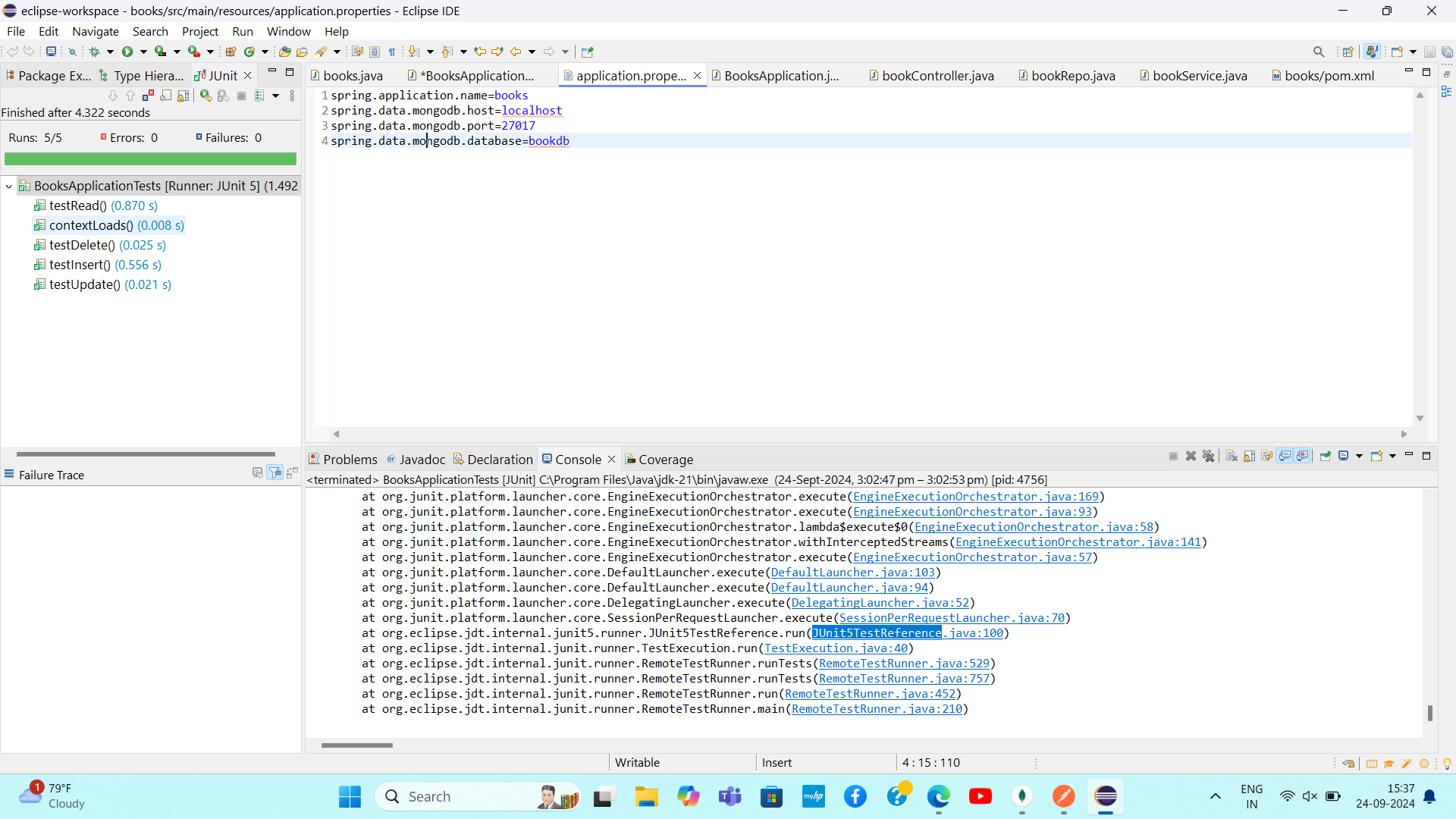
r.deleteByIsbn("20EC51I");

}

}

**Now run it:**

Right Click to your BooksApplicatiom.java 🡪Run as🡪Junit.



Check it in your mongodb compass:

