**Cognizant\_Digital Nurture 4.0\_Deep Skilling**

**Products and Frameworks**

**Module 5 - Spring Core and Maven**

**Exercise 1: Configuring a Basic Spring Application**

**LibraryManagement/pom.xml**

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</project>

**applicationContext.xml**

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

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*https://www.springframework.org/schema/beans/spring-beans.xsd"*>

<!-- Define BookRepository bean -->

<**bean** id=*"bookRepository"* class=*"com.library.repository.BookRepository"* />

<!-- Define BookService bean and inject BookRepository -->

<**bean** id=*"bookService"* class=*"com.library.service.BookService"*>

<**property** name=*"bookRepository"* ref=*"bookRepository"* />

</**bean**>

</**beans**>

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.***out***.println("Saving book: " + bookName + " to the database.");

}

public void deleteBook(String bookName) {

System.***out***.println("Deleting book: " + bookName + " from the database.");

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.***out***.println("BookService: Adding book - " + bookName);

bookRepository.saveBook(bookName);

}

public void removeBook(String bookName) {

System.***out***.println("BookService: Removing book - " + bookName);

bookRepository.deleteBook(bookName);

}

}

**App.java**

package com.library.LibraryManagement;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

// Load the Spring application context from the XML file

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

// Get the BookService bean from the context

BookService bookService = (BookService) context.getBean("bookService");

// Use the BookService to perform operations

bookService.addBook("The Great Gatsby");

bookService.removeBook("1984");

// Close the context (important for proper resource release in production apps)

((ClassPathXmlApplicationContext) context).close();

}

}

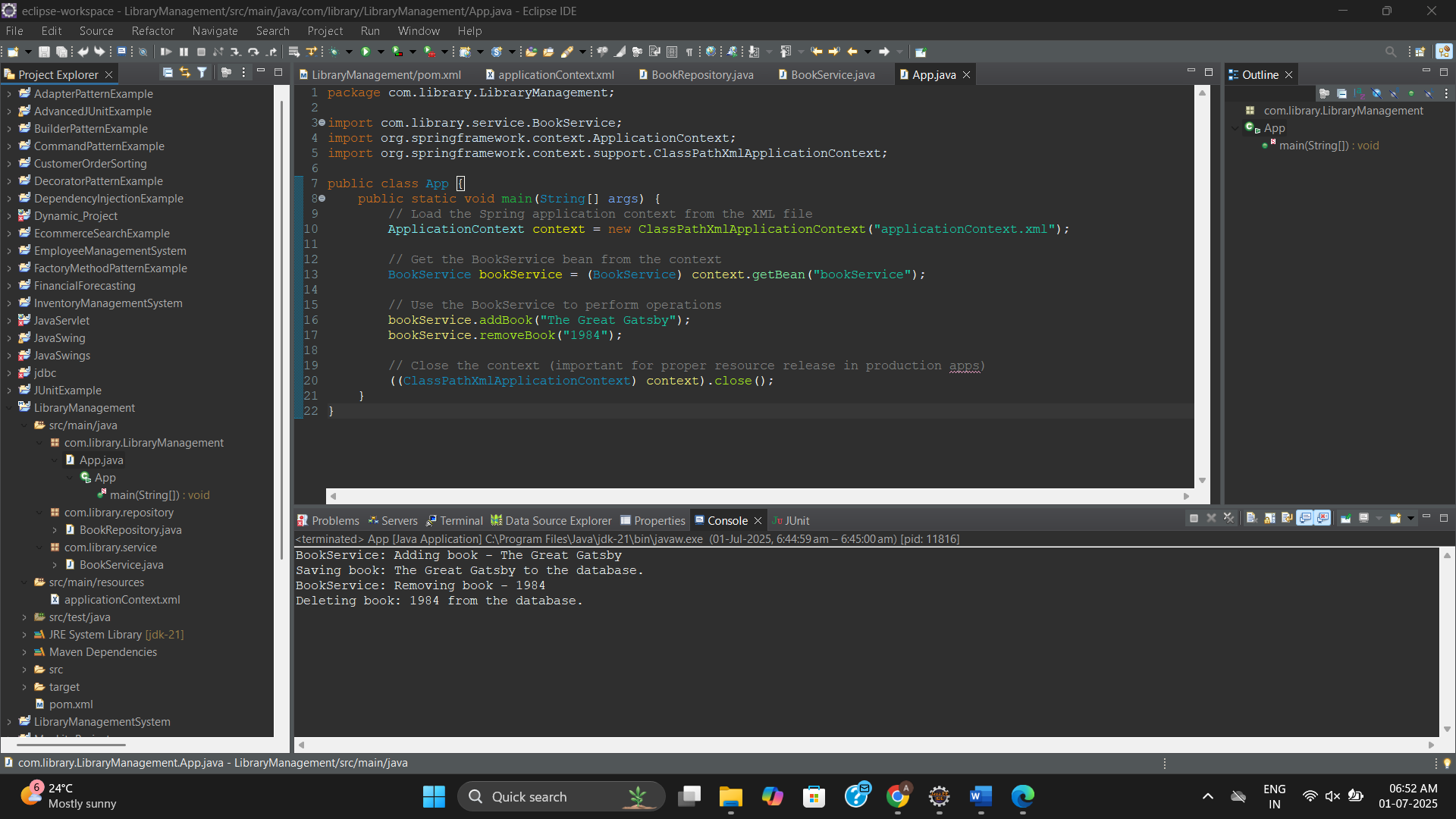
**Output:**

BookService: Adding book - The Great Gatsby

Saving book: The Great Gatsby to the database.

BookService: Removing book - 1984

Deleting book: 1984 from the database.

****

**Exercise 2: Implementing Dependency Injection**

**LibraryManagement/pom.xml**

<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</project>

**applicationContext.xml**

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

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*https://www.springframework.org/schema/beans/spring-beans.xsd"*>

<!-- Define BookRepository bean -->

<**bean** id=*"bookRepository"* class=*"com.library.repository.BookRepository"* />

<!-- Define BookService bean and inject BookRepository -->

<**bean** id=*"bookService"* class=*"com.library.service.BookService"*>

<**property** name=*"bookRepository"* ref=*"bookRepository"* />

</**bean**>

</**beans**>

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.***out***.println("Saving book: " + bookName + " to the database.");

}

public void deleteBook(String bookName) {

System.***out***.println("Deleting book: " + bookName + " from the database.");

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.***out***.println("BookService: Adding book - " + bookName);

bookRepository.saveBook(bookName);

}

public void removeBook(String bookName) {

System.***out***.println("BookService: Removing book - " + bookName);

bookRepository.deleteBook(bookName);

}

}

**App.java**

package com.library.LibraryManagement;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

// Load the Spring application context from the XML file

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

// Get the BookService bean from the context

BookService bookService = (BookService) context.getBean("bookService");

// Use the BookService to perform operations

bookService.addBook("The Great Gatsby");

bookService.removeBook("1984");

// Close the context (important for proper resource release in production apps)

((ClassPathXmlApplicationContext) context).close();

}

}

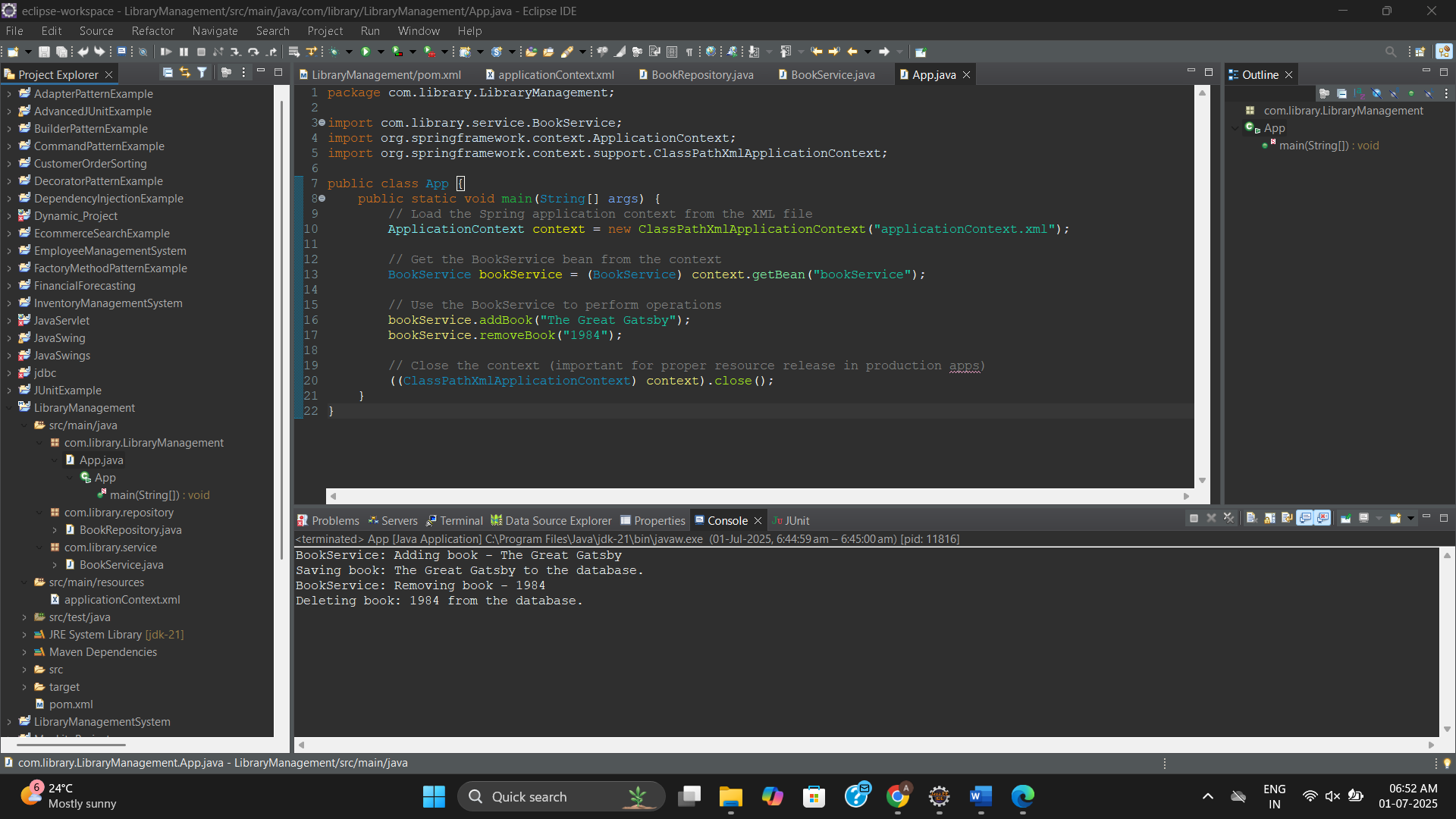
**Output:**

BookService: Adding book - The Great Gatsby

Saving book: The Great Gatsby to the database.

BookService: Removing book - 1984

Deleting book: 1984 from the database.

****

**Exercise 4: Creating and Configuring a Maven Project**

**LibraryManagement/pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring WebMVC (optional but good to include for web support) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler plugin for Java 21 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>21</source>

<target>21</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.***out***.println("Book saved: " + bookName);

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookName) {

System.***out***.println("Adding book...");

bookRepository.saveBook(bookName);

}

}

**applicationContext.xml**

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

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*https://www.springframework.org/schema/beans/spring-beans.xsd"*>

<**bean** id=*"bookRepository"* class=*"com.library.repository.BookRepository"* />

<**bean** id=*"bookService"* class=*"com.library.service.BookService"*>

<**property** name=*"bookRepository"* ref=*"bookRepository"*/>

</**bean**>

</**beans**>

**App.java**

package com.library.LibraryManagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class App {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.addBook("The Great Gatsby");

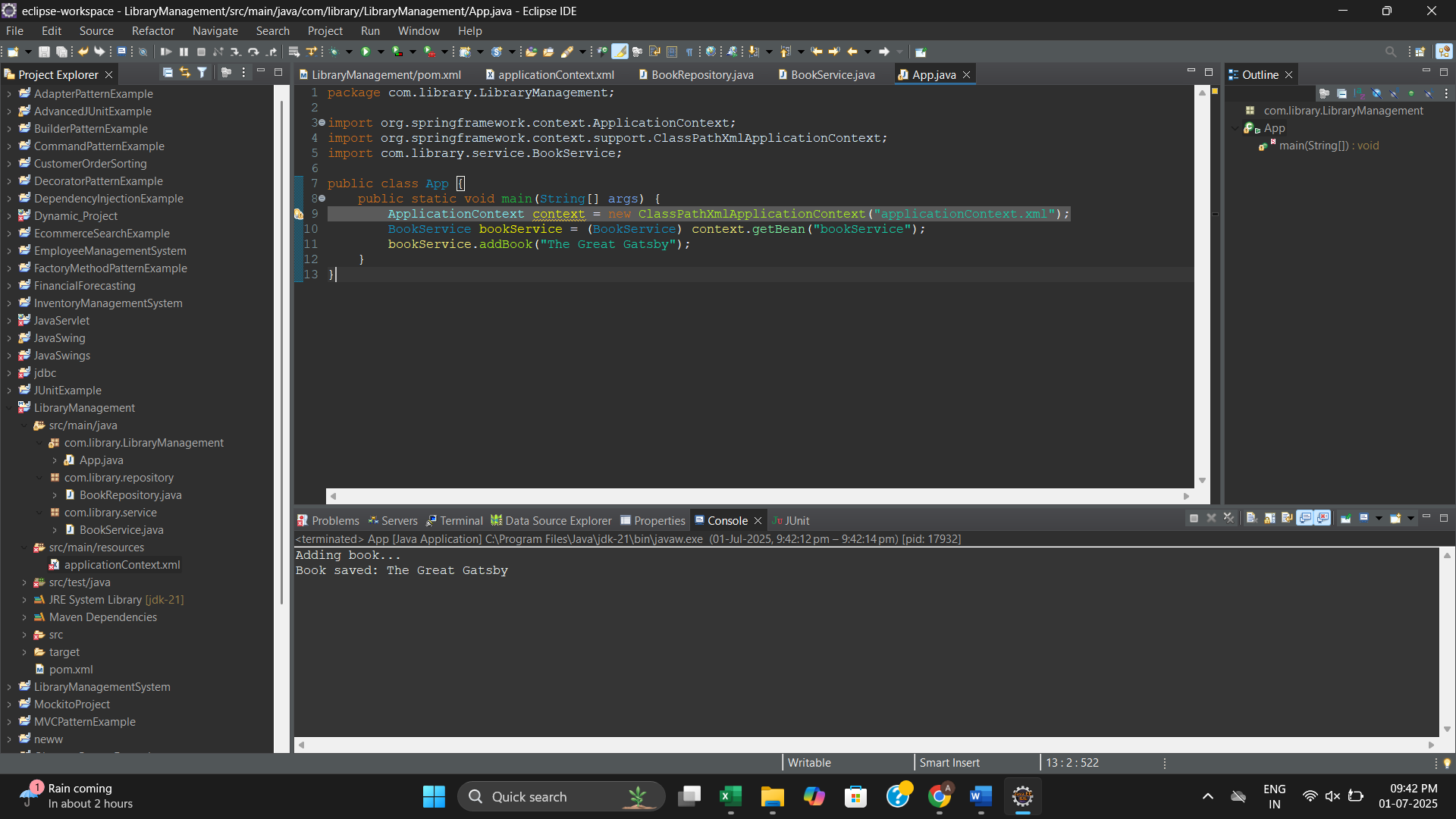
}

}

**Output:**

Adding book...

Book saved: The Great Gatsby

****

**Exercise 5: Configuring the Spring IoC Container**

**LibraryManagement/pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring WebMVC (optional but good to include for web support) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler plugin for Java 21 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>21</source>

<target>21</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**applicationContext.xml**

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

<**beans** xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<!-- Bean for BookRepository -->

<**bean** id=*"bookRepository"* class=*"com.library.repository.BookRepository"* />

<!-- Bean for BookService with dependency injection -->

<**bean** id=*"bookService"* class=*"com.library.service.BookService"*>

<**property** name=*"bookRepository"* ref=*"bookRepository"* />

</**bean**>

</**beans**>

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void displayBookList() {

System.***out***.println("Displaying list of books from BookRepository.");

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void showBooks() {

System.***out***.println("Inside BookService:");

bookRepository.displayBookList();

}

}

**App.java**

package com.library.LibraryManagement;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.showBooks();

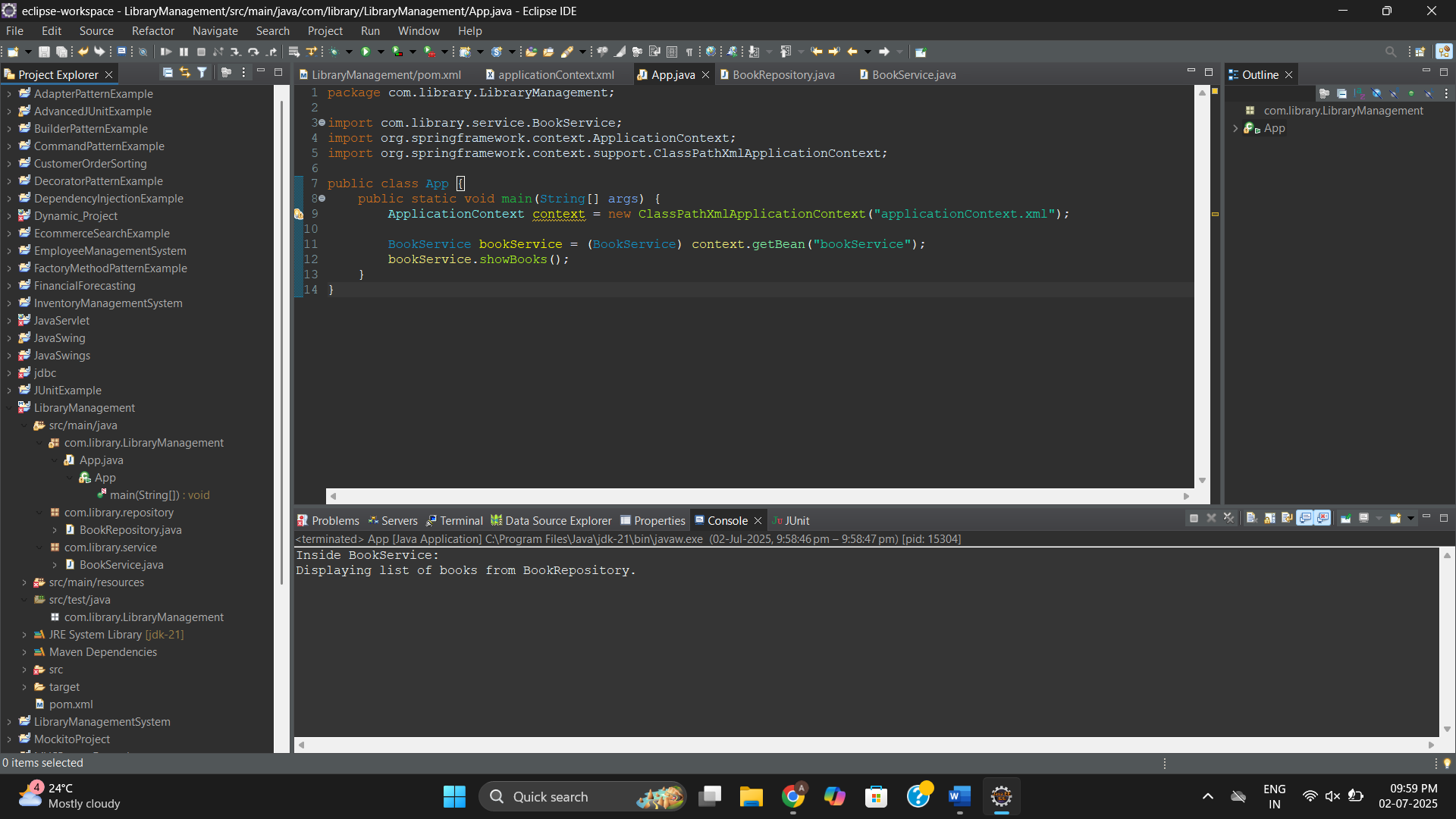
}

}

**Output:**

Inside BookService:

Displaying list of books from BookRepository.

****

**Exercise 7: Implementing Constructor and Setter Injection**

**LibraryManagement/pom.xml**

<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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring WebMVC (optional but good to include for web support) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Compiler plugin for Java 21 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>21</source>

<target>21</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

**applicationContext.xml**

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- Bean for BookRepository -->

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<!-- Bean for BookService with constructor and setter injection -->

<bean id="bookService" class="com.library.service.BookService">

<!-- Constructor injection -->

<constructor-arg ref="bookRepository" />

<!-- Setter injection (overrides constructor value) -->

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void displayBookList() {

System.out.println("Displaying list of books from BookRepository.");

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Constructor for constructor injection

public BookService(BookRepository bookRepository) {

System.out.println("Constructor injection called.");

this.bookRepository = bookRepository;

}

// Setter for setter injection

public void setBookRepository(BookRepository bookRepository) {

System.out.println("Setter injection called.");

this.bookRepository = bookRepository;

}

public void showBooks() {

System.out.println("Inside BookService:");

bookRepository.displayBookList();

}

}

**App.java**

package com.library.LibraryManagement;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

BookService bookService = (BookService) context.getBean("bookService");

bookService.showBooks();

}

}

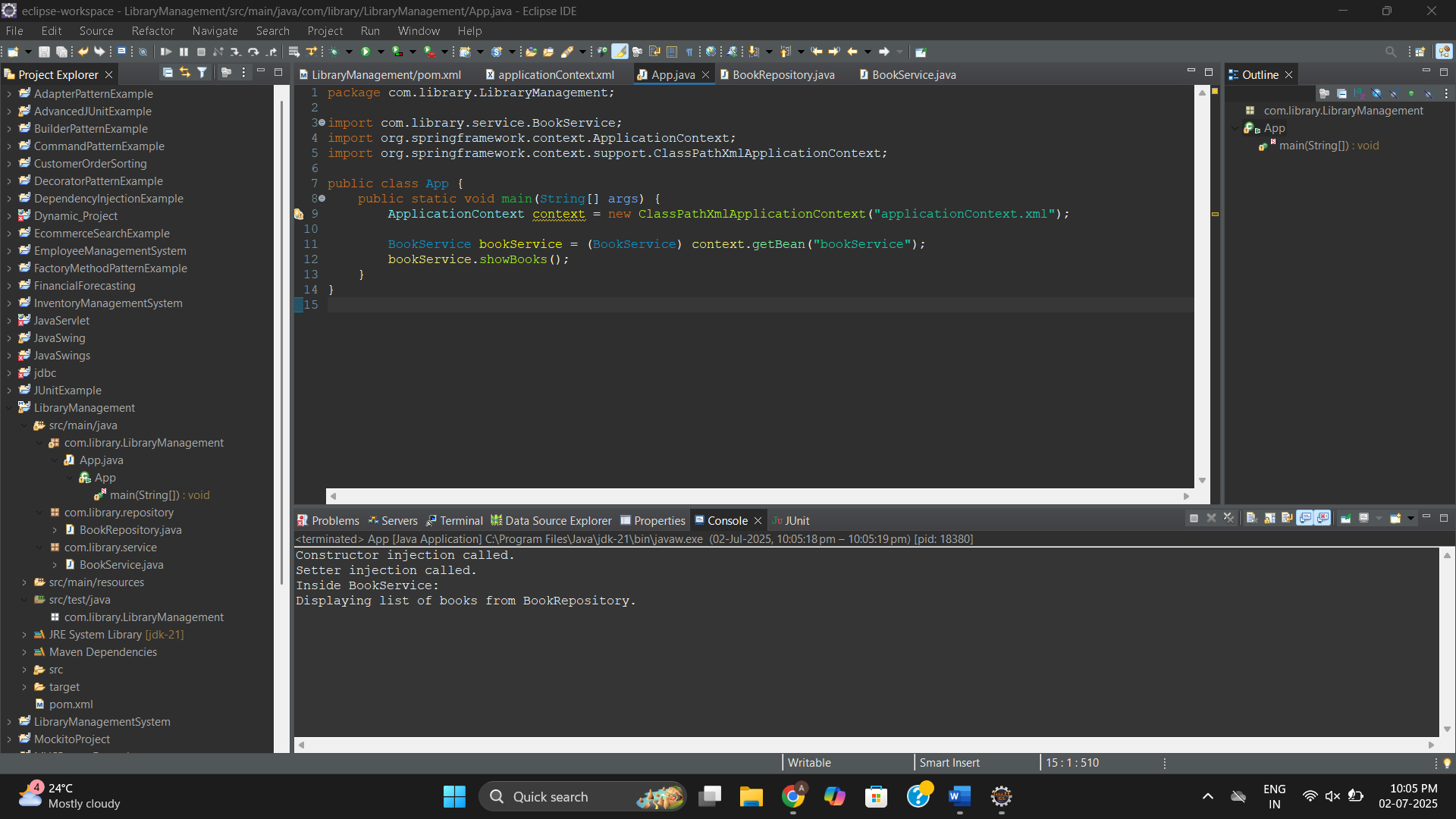
**Output:**

Constructor injection called.

Setter injection called.

Inside BookService:

Displaying list of books from BookRepository.

****

**Exercise 9: Creating a Spring Boot Application**

**LibraryManagementBootApplication.java**

package com.example.LibraryManagementBoot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.CommandLineRunner;

import org.springframework.context.annotation.Bean;

import com.example.LibraryManagementBoot.entity.Book;

import com.example.LibraryManagementBoot.repository.BookRepository;

*@SpringBootApplication*

public class LibraryManagementBootApplication {

public static void main(String[] args) {

SpringApplication.*run*(LibraryManagementBootApplication.class, args);

}

*@Bean*

public CommandLineRunner addSampleBooks(BookRepository bookRepository) {

return args -> {

bookRepository.save(new Book("Wings of Fire", "A. P. J. Abdul Kalam"));

bookRepository.save(new Book("The Alchemist", "Paulo Coelho"));

bookRepository.save(new Book("Ignited Minds", "A. P. J. Abdul Kalam"));

};

}

}

**BookController.java**

package com.example.LibraryManagementBoot.controller;

import com.example.LibraryManagementBoot.entity.Book;

import com.example.LibraryManagementBoot.repository.BookRepository;

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

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

import java.util.List;

*@RestController*

*@RequestMapping*("/books")

public class BookController {

*@Autowired*

private BookRepository bookRepository;

*@GetMapping*

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

*@PostMapping*

public Book addBook(*@RequestBody* Book book) {

return bookRepository.save(book);

}

}

**Book.java**

package com.example.LibraryManagementBoot.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

*@Entity*

public class Book {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private Long id;

private String title;

private String author;

public Book() {}

public Book(String title, String author) {

this.title = title;

this.author = author;

}

// Getters and Setters

public Long getId() { return id; }

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

public String getTitle() { return title; }

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

public String getAuthor() { return author; }

public void setAuthor(String author) { this.author = author; }

}

**BookRepository.java**

package com.example.LibraryManagementBoot.repository;

import com.example.LibraryManagementBoot.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

}

**application.properties**

spring.application.name=LibraryManagementBoot

# === Database Configuration ===

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

# === Hibernate / JPA Settings ===

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

# === H2 Console ===

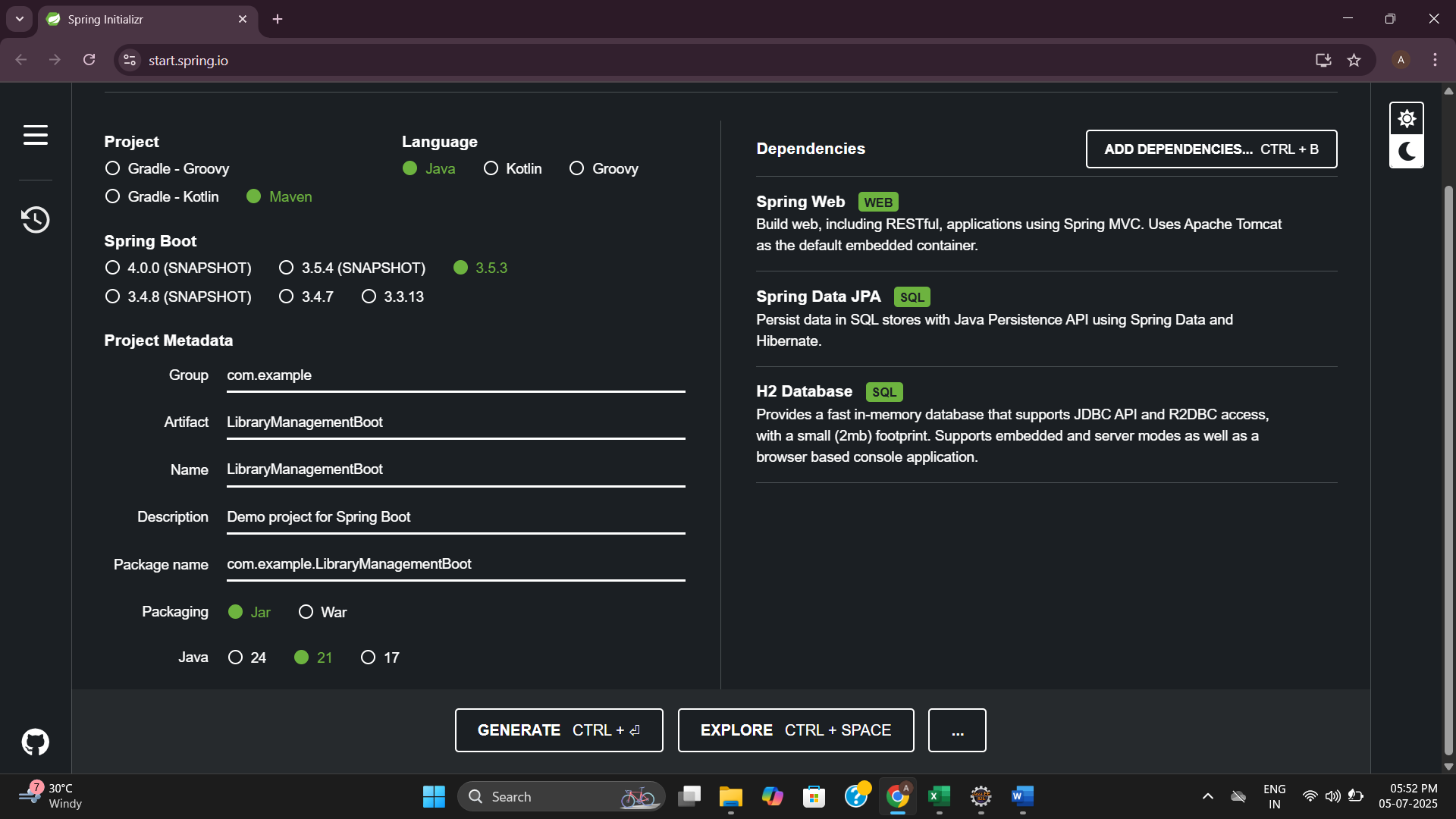
spring.h2.console.enabled=true

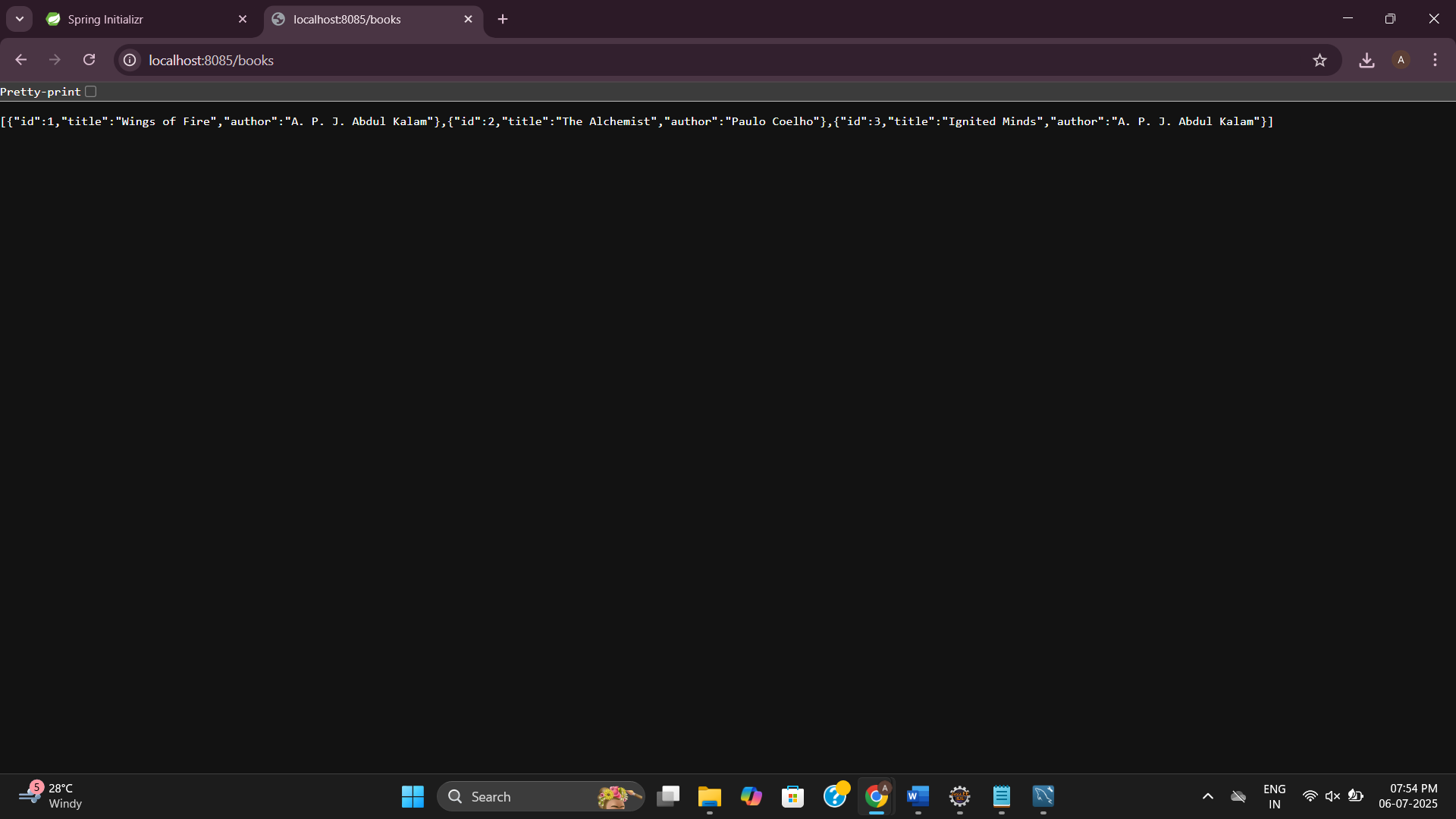
spring.h2.console.path=/h2-console

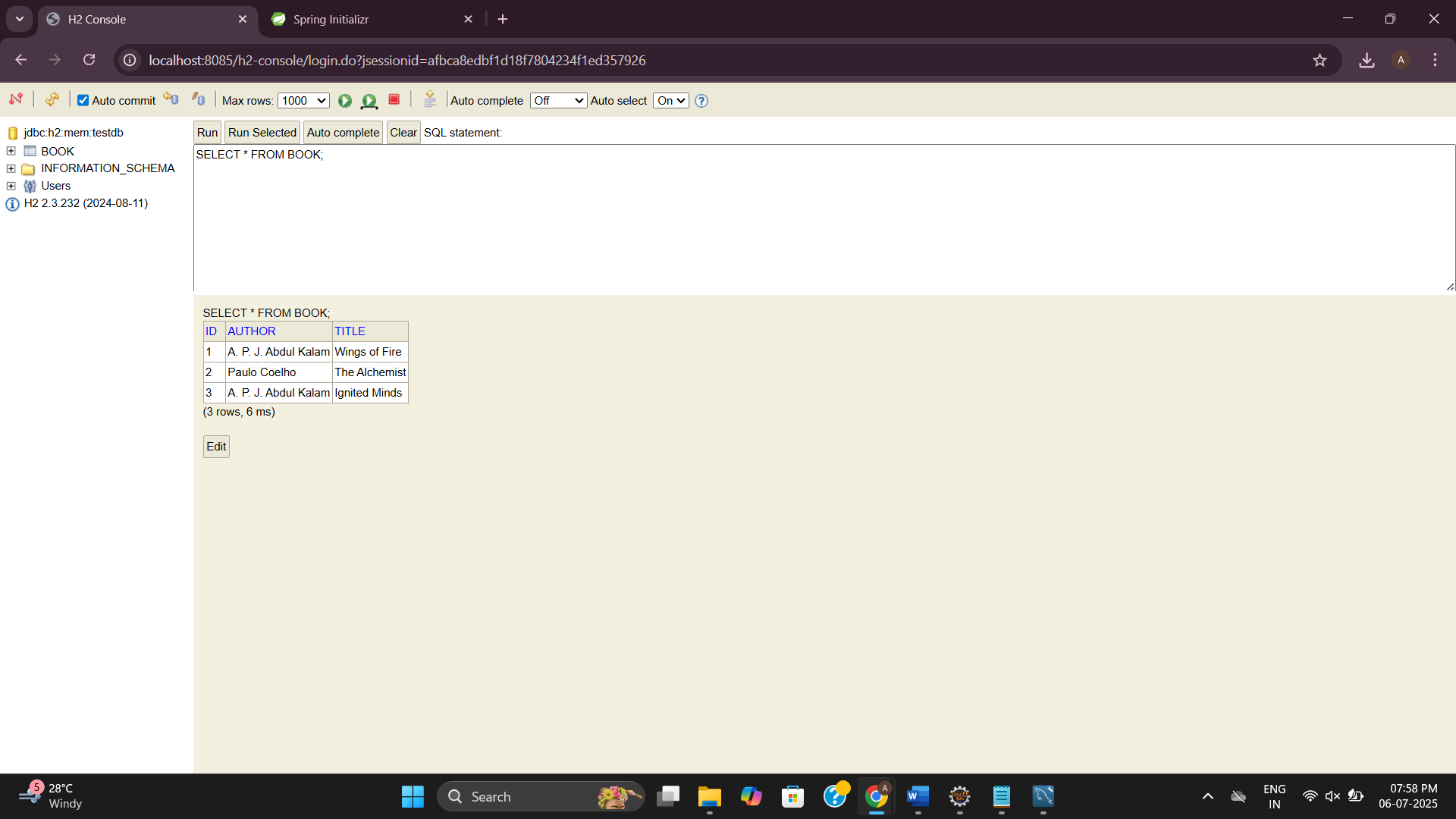
# === Custom Port (optional) ===

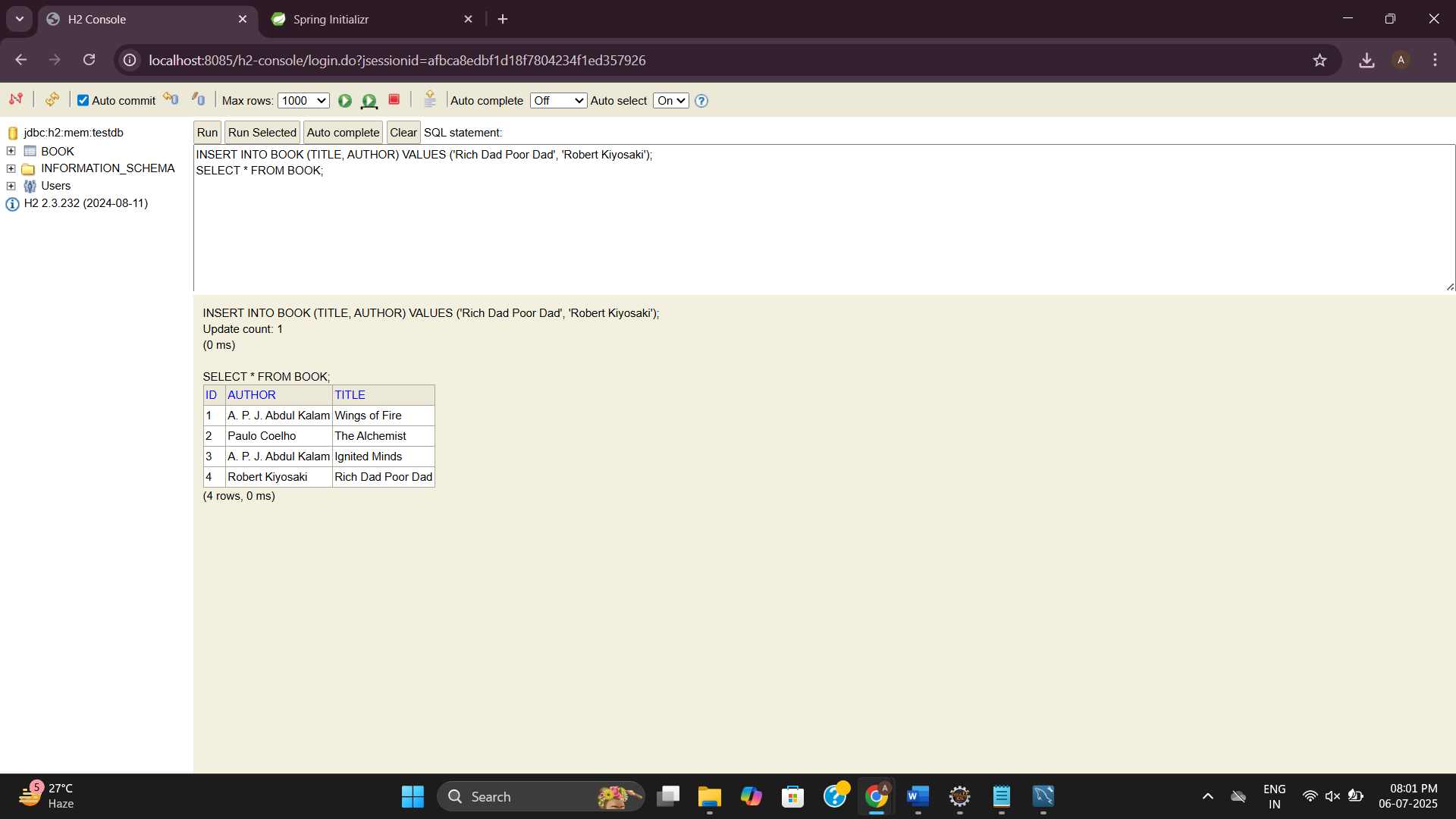
server.port=8085

**Output:**

****

****

****

****