**Exercise 1: Configuring a Basic Spring Application for Library Management**

**1. Introduction**

This document provides a step-by-step guide to create a basic Spring Framework-based application to manage library operations. The application is built using Java, Maven, and Spring Core, and is developed in Visual Studio Code (VS Code) on a Windows system. The application demonstrates how to configure a Spring project using XML-based configuration, define service and repository beans, and run the application using Spring’s ApplicationContext.

**2. Project Objective**

To create a simple backend application for a library management system using the Spring Framework. The application will have the following components:

* BookRepository: Handles data-related operations
* BookService: Contains business logic and uses BookRepository

**3. Tools and Technologies Used**

* Java (JDK 8 or above)
* Apache Maven
* Spring Framework (spring-context dependency)
* Visual Studio Code (VS Code)
* Windows Command Prompt

**4. Project Setup**

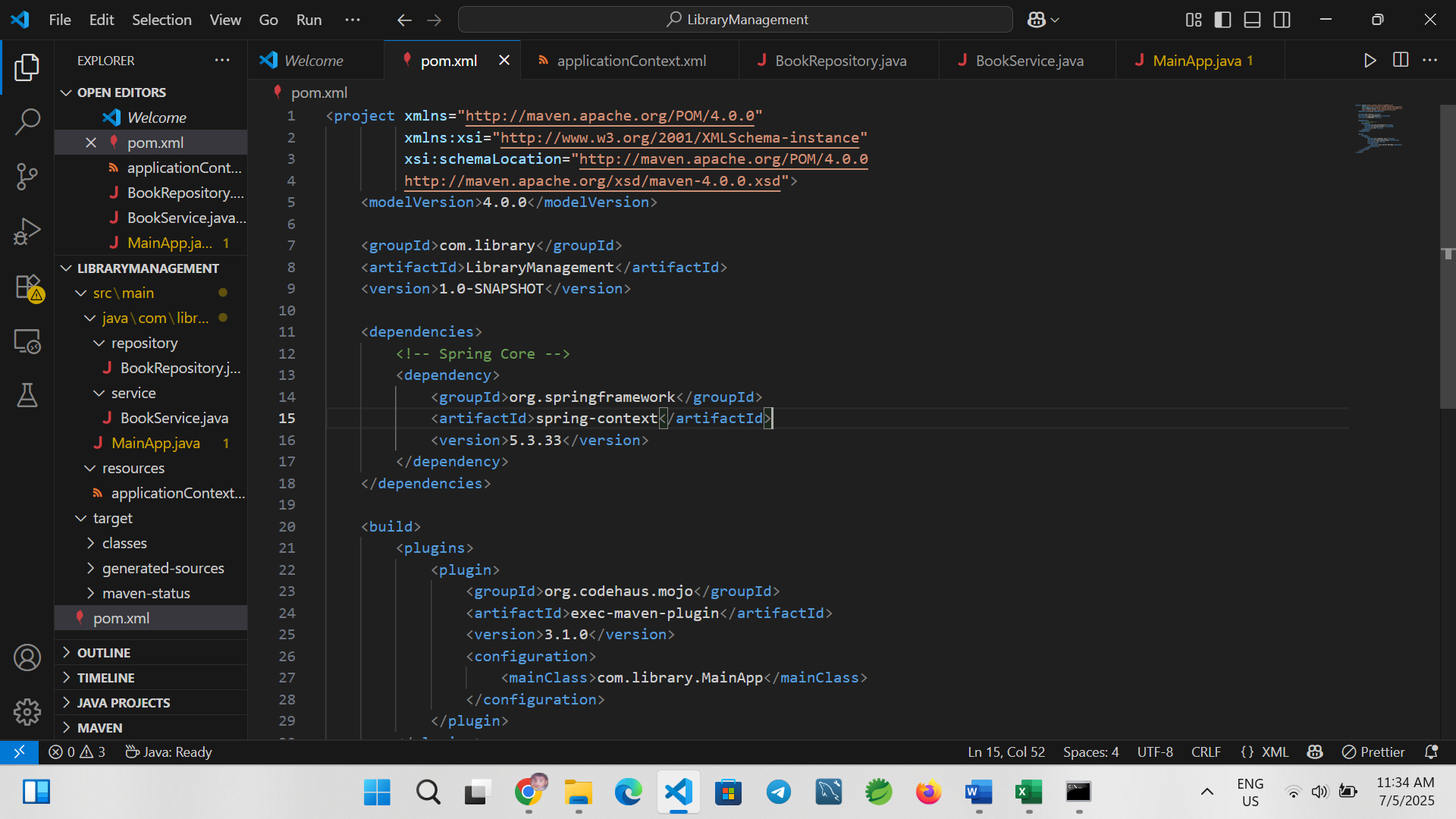
**Step 1: Create Folder Structure** Use Command Prompt to create the required directories:

Mkdir LibraryManagement

Cd LibraryManagement

mkdir src\main\java\com\library\service  
mkdir src\main\java\com\library\repository  
mkdir src\main\resources

**Step 2: Create pom.xml** Create a pom.xml in the root (LibraryManagement) folder with the following content:

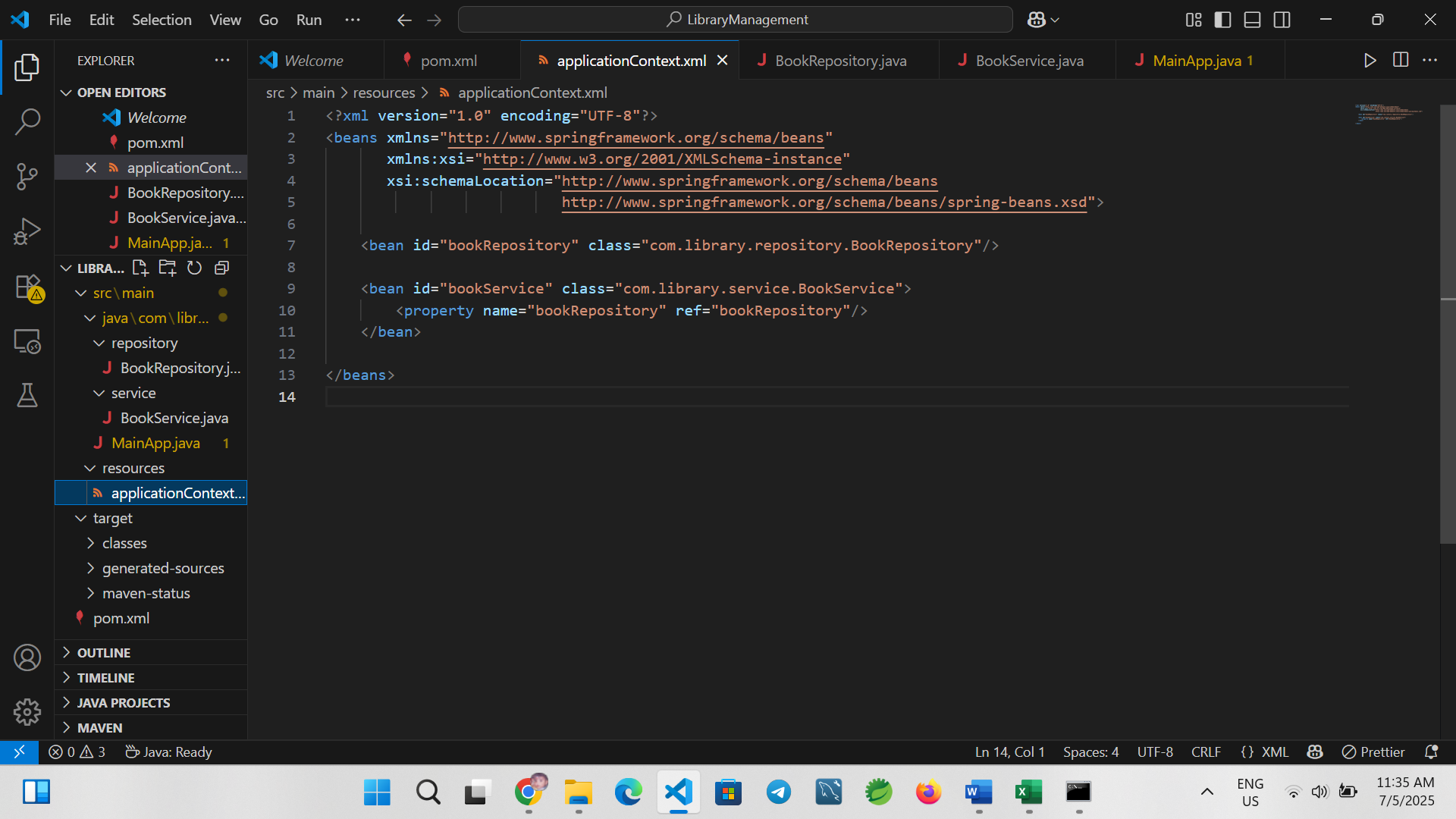


Run:

mvn clean install

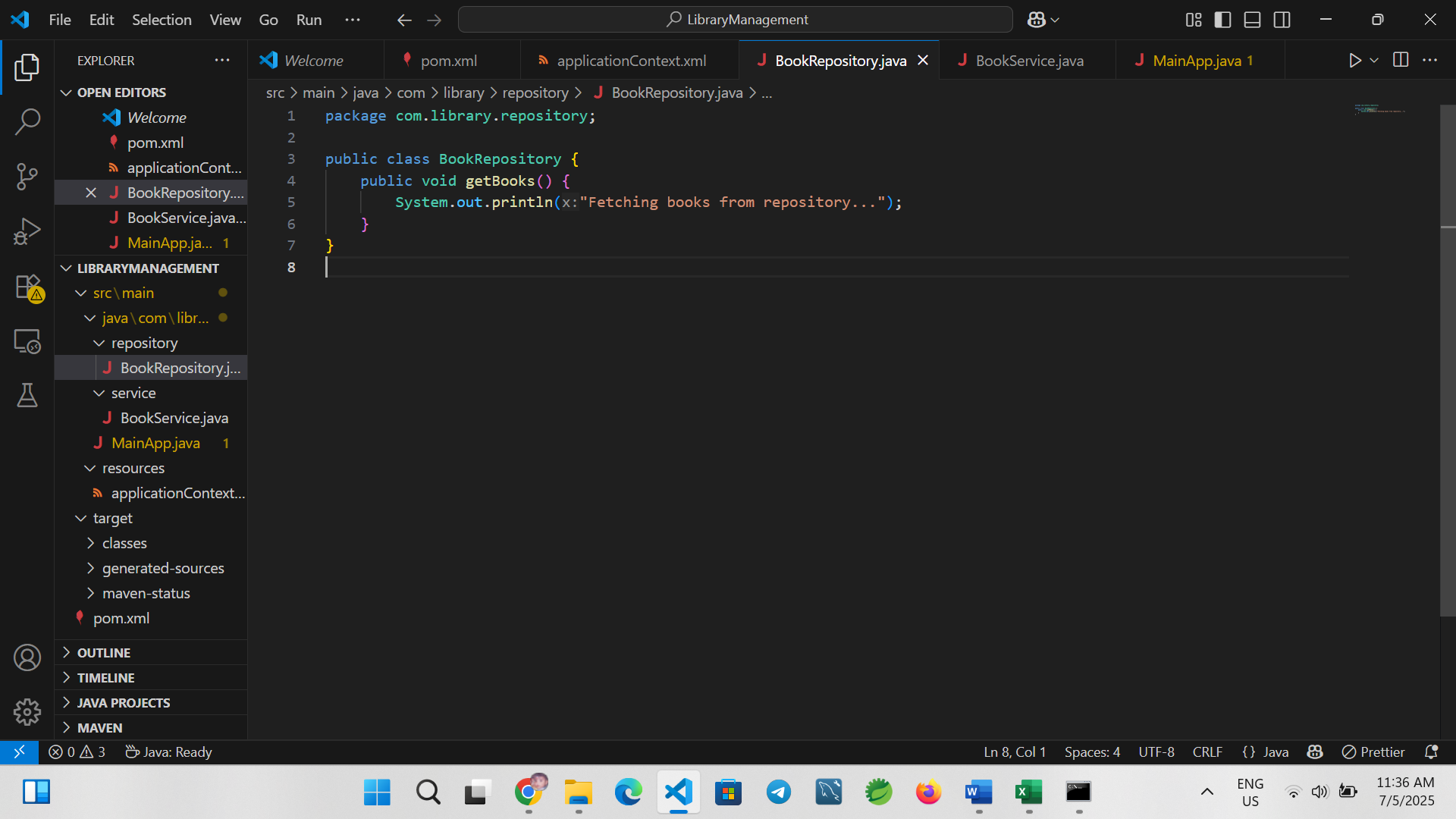
**5. Spring Configuration**

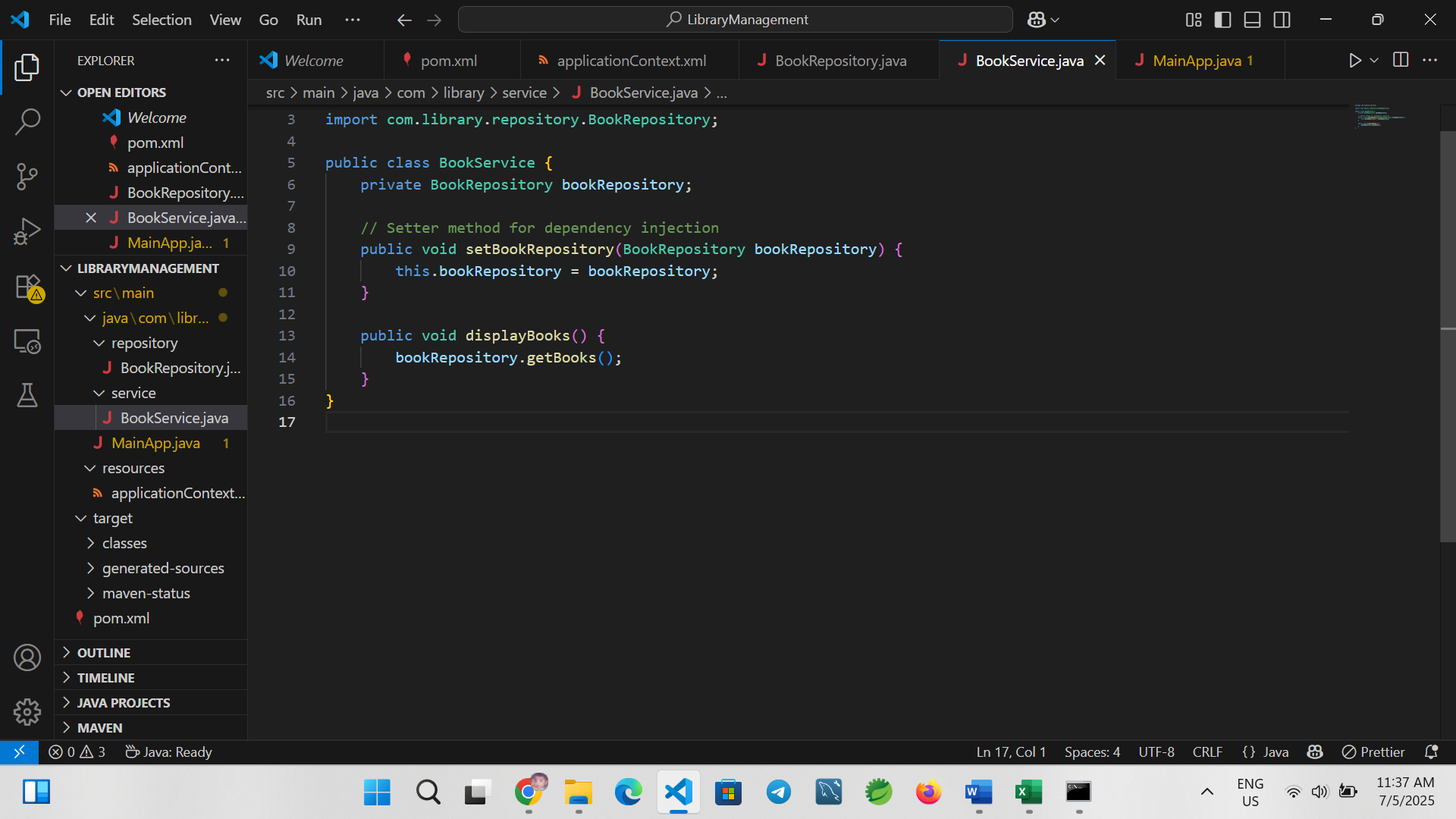
**Step 3: Create applicationContext.xml** Location: src/main/resources/applicationContext.xml

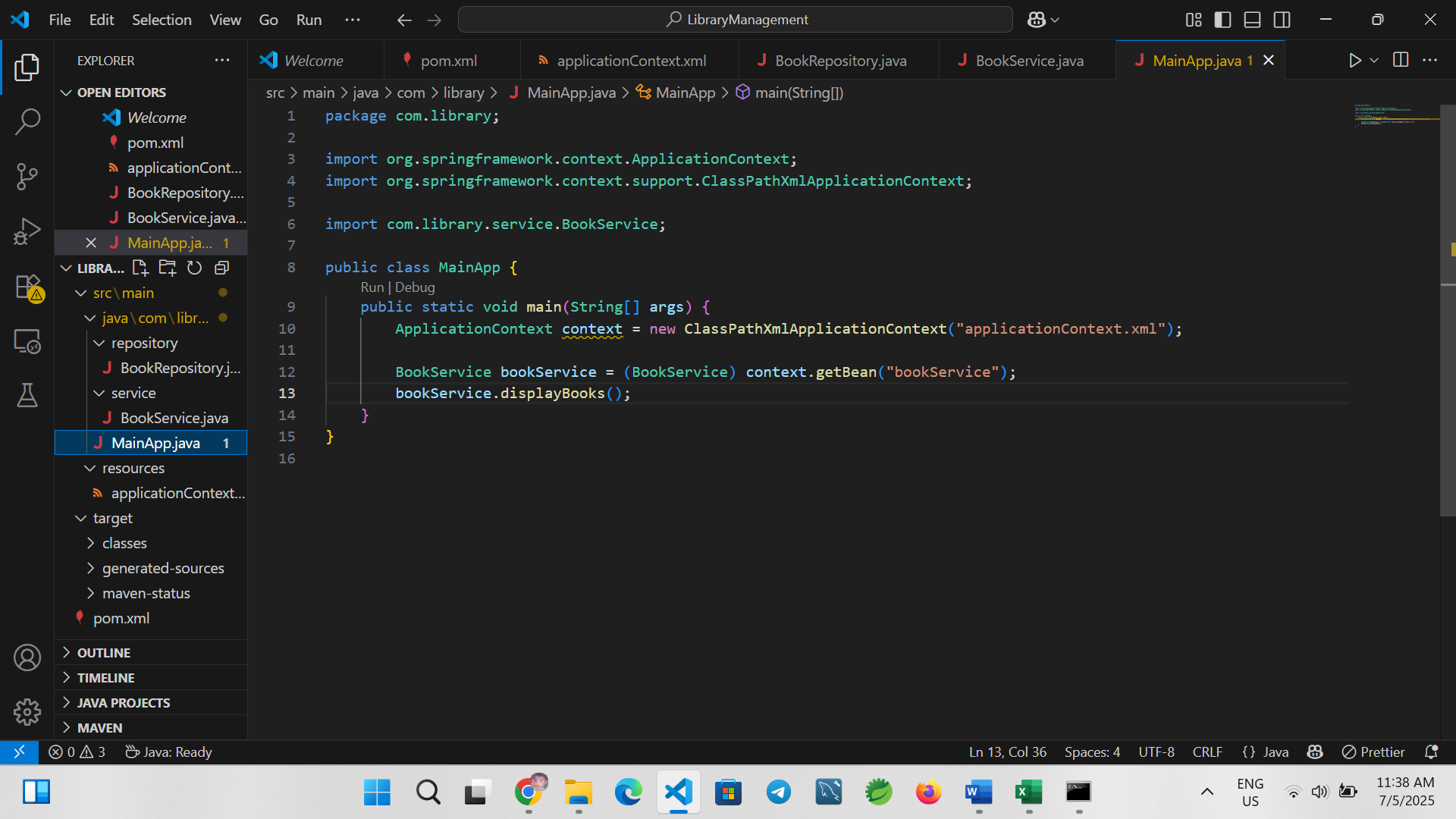


**6. Java Classes**

**BookRepository.java**  
Location: src/main/java/com/library/repository/BookRepository.java

**BookService.java**  
Location: src/main/java/com/library/service/BookService.java

**MainApp.java**  
Location: src/main/java/com/library/MainApp.java



**7. Build and Run the Application**

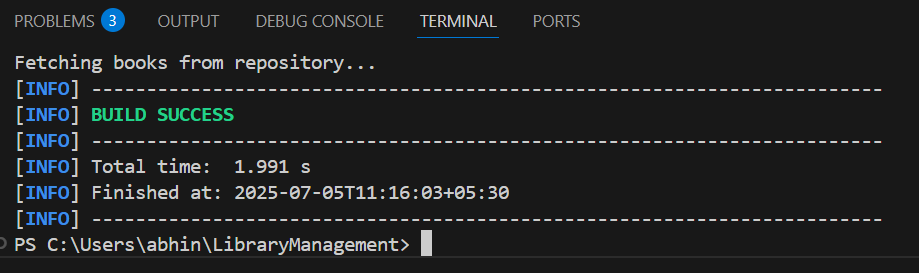
To compile:

mvn clean compile

To run:

mvn exec:java

**Expected Output:**



**8. Conclusion**

In this exercise, we successfully created a simple Spring-based application that demonstrates XML configuration, dependency injection, and Maven build management. This foundation can be extended with features like annotation-based configuration, Spring Boot, database integration, and REST APIs.

**Exercise 2: Implementing Dependency Injection in Spring**

**1. Introduction**

This document explains the implementation of Dependency Injection (DI) in a Spring-based Library Management application. The goal of this exercise is to demonstrate how Spring’s Inversion of Control (IoC) container manages dependencies between components using setter-based dependency injection. We use XML configuration to wire the BookService and BookRepository beans.

**2. Objective**

To configure Spring to inject the BookRepository into BookService using Spring’s IoC container and validate the injection through a simple application run.

**3. Tools and Technologies**

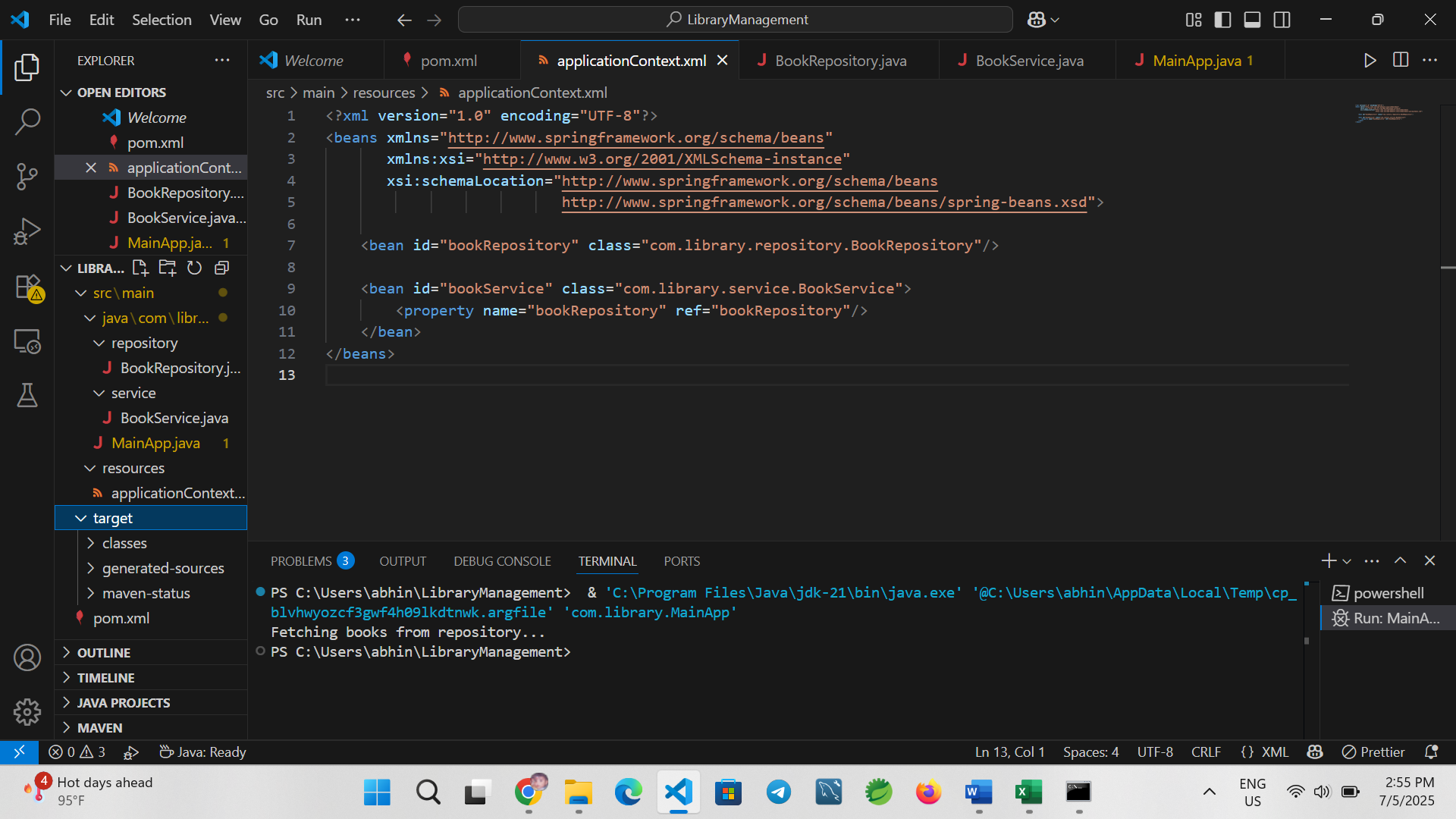
* Java (JDK 8 or above)
* Spring Framework (spring-context)
* Apache Maven
* Visual Studio Code (VS Code)
* Windows Command Prompt

**4. Steps to Implement Dependency Injection**

**Step 1: Modify the XML Configuration**

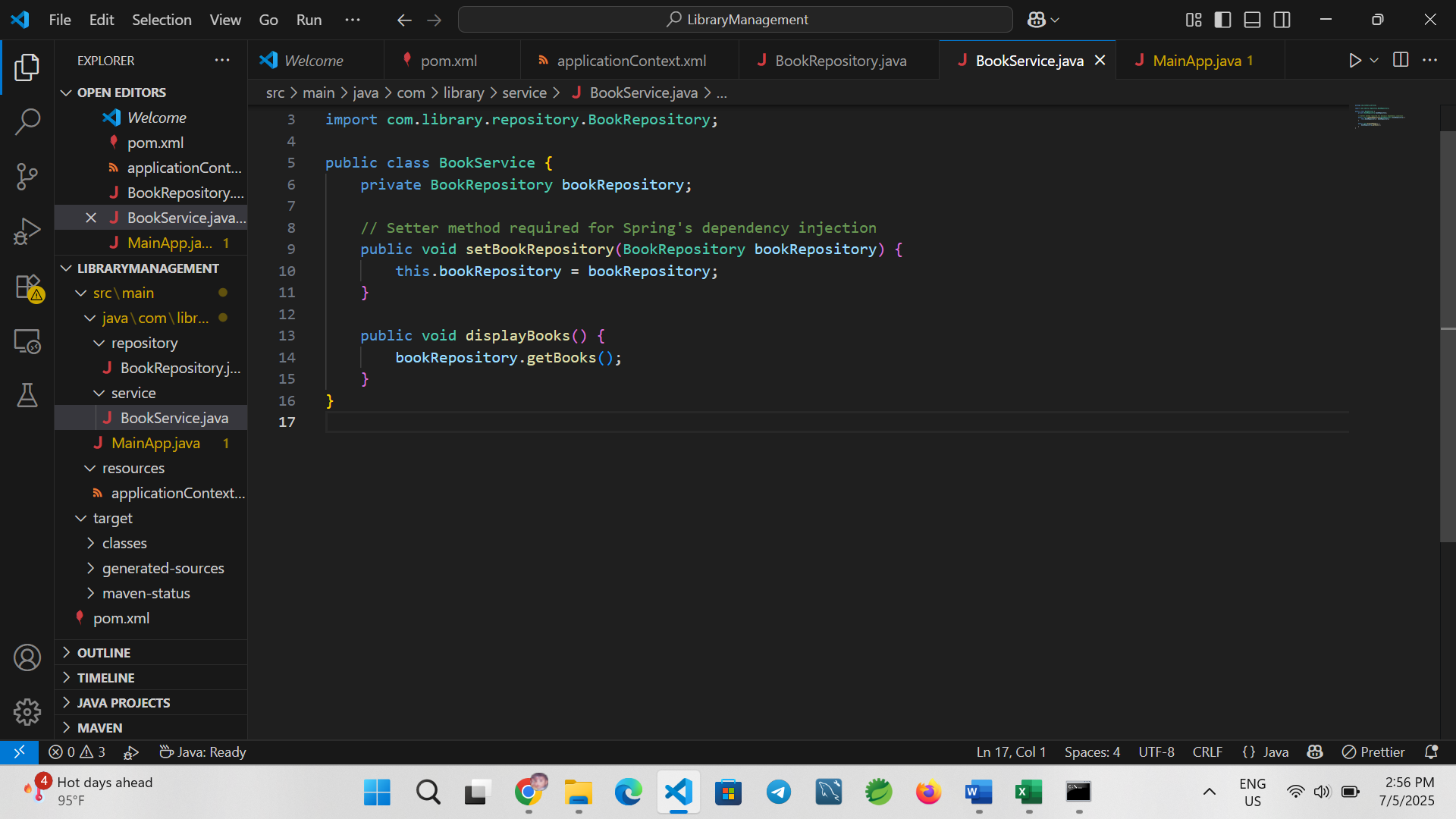
Location: src/main/resources/applicationContext.xml

Update the content to:

This configuration defines two beans and uses setter injection to inject BookRepository into BookService.

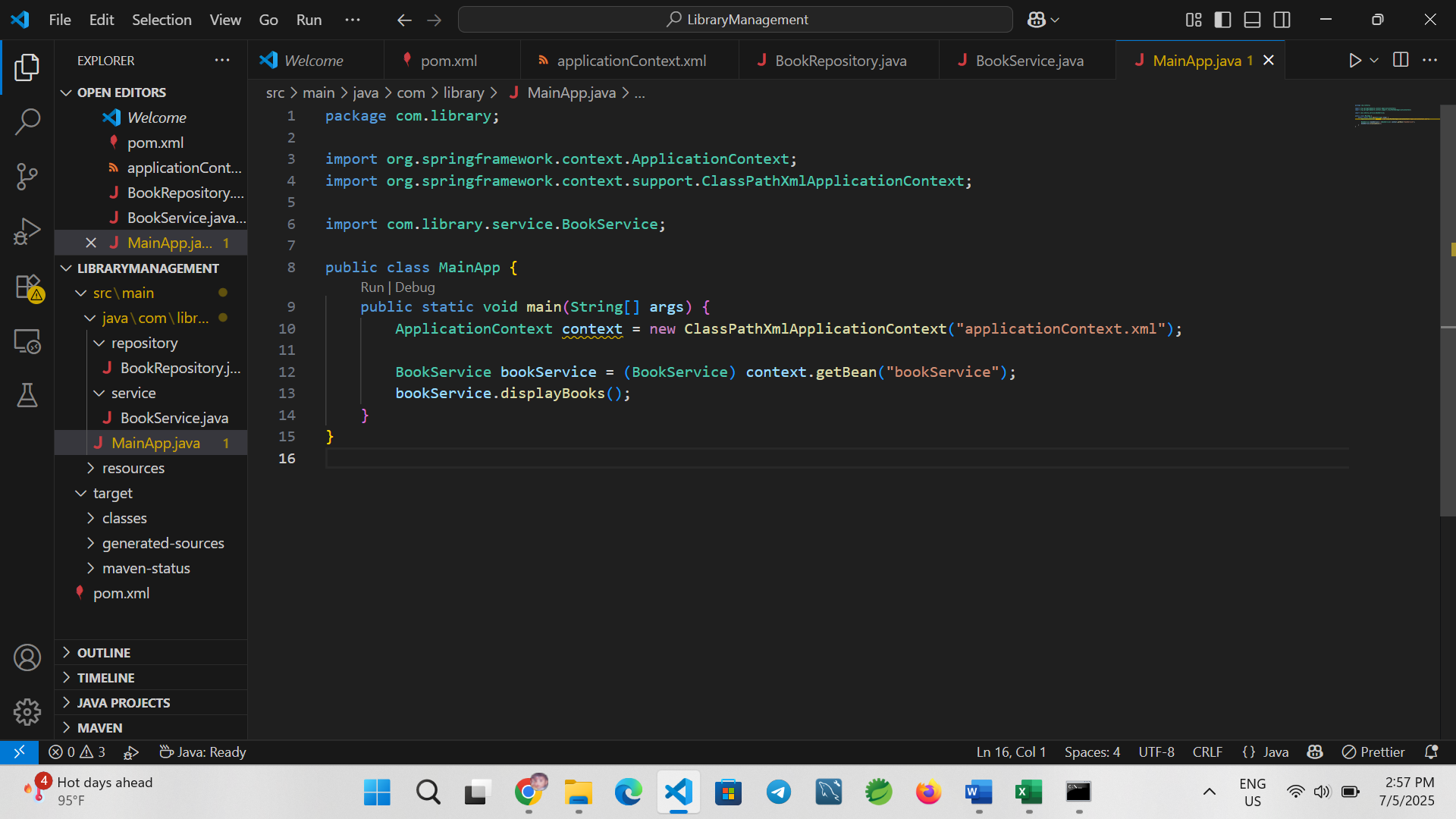
**Step 2: Ensure BookService Has a Setter Method**

Location: src/main/java/com/library/service/BookService.java

Ensure the setBookRepository() method is present so Spring can inject the dependency via setter.

**Step 3: Test the Configuration**

Location: src/main/java/com/library/MainApp.java

This main class loads the Spring context and retrieves the bookService bean to verify the injection.

**Step 4: Compile and Run the Application**

Use the following Maven commands:

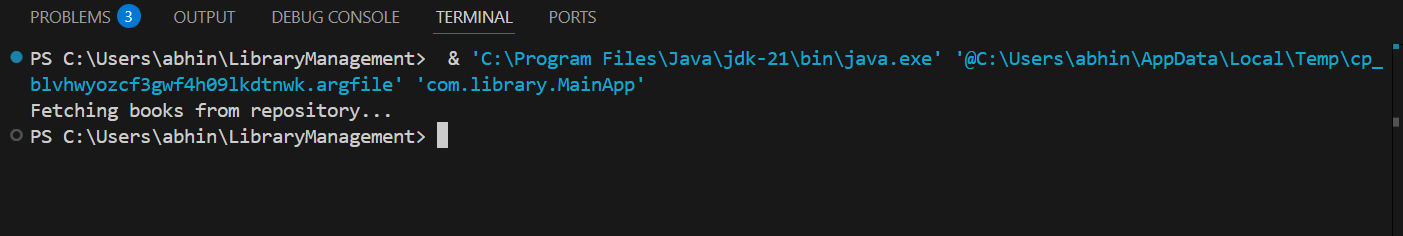
To compile the code:

mvn clean compile

To run the application:

mvn exec:java

**Expected Output:**



**5. Conclusion**

In this exercise, we implemented setter-based dependency injection using Spring’s XML configuration. The application demonstrated how Spring can manage object dependencies through the IoC container without the need for manual instantiation. This approach enhances modularity, testability, and maintainability in enterprise applications.

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

**1. Introduction**

This document outlines the process for creating and configuring a Maven project for a Spring-based Library Management application. The goal is to set up a new Maven project, add necessary Spring dependencies, and configure the Maven Compiler Plugin for Java 1.8. The project will serve as a foundation for future development using the Spring Framework.

**2. Objective**

* Create a Maven project named LibraryManagement
* Add dependencies for Spring Context, Spring AOP, and Spring WebMVC
* Configure the Maven Compiler Plugin for Java 1.8

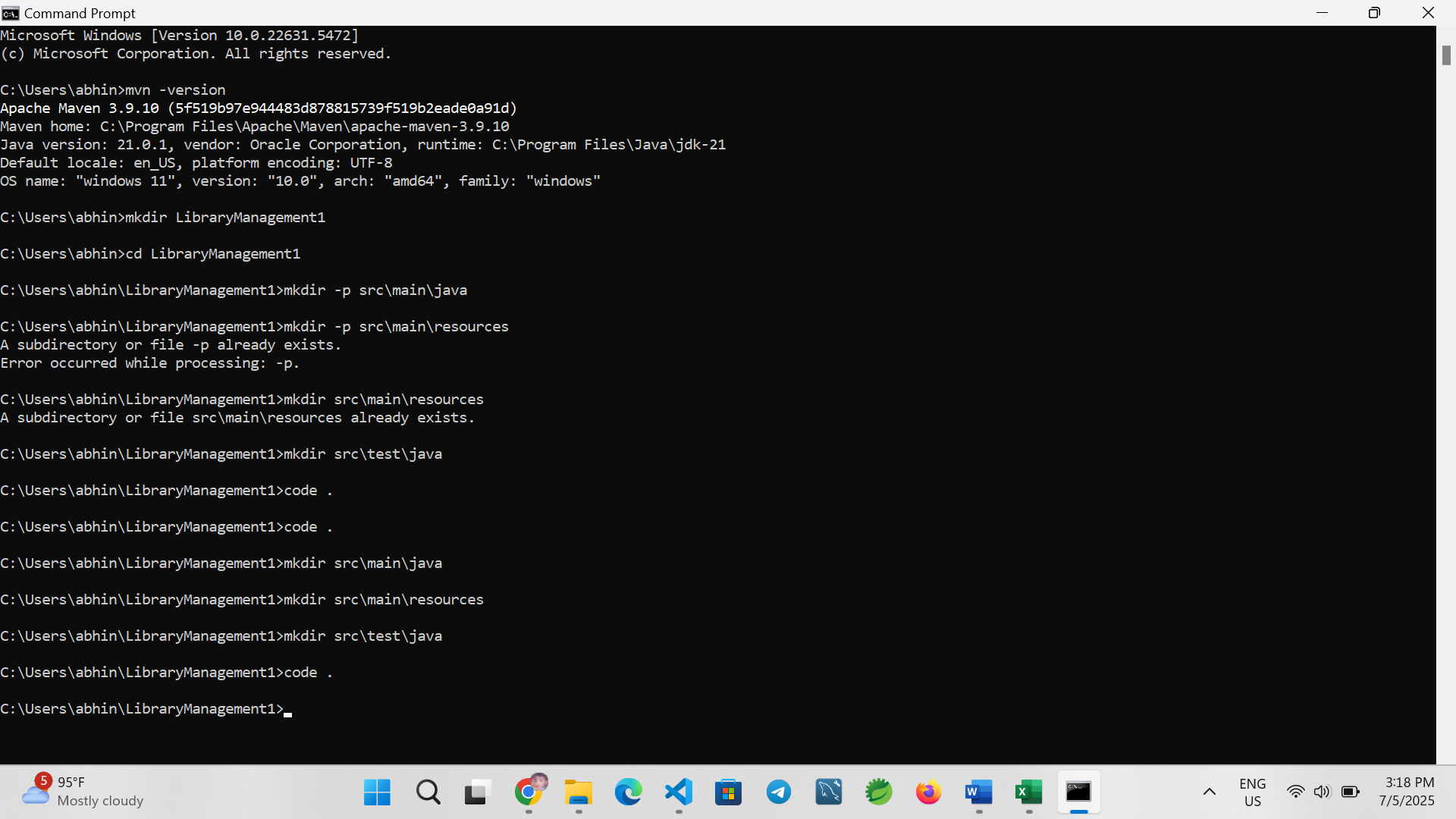
**3. Tools and Technologies**

* Java JDK 8 or higher
* Apache Maven
* Visual Studio Code (VS Code)
* Windows Command Prompt

**4. Project Setup Steps**

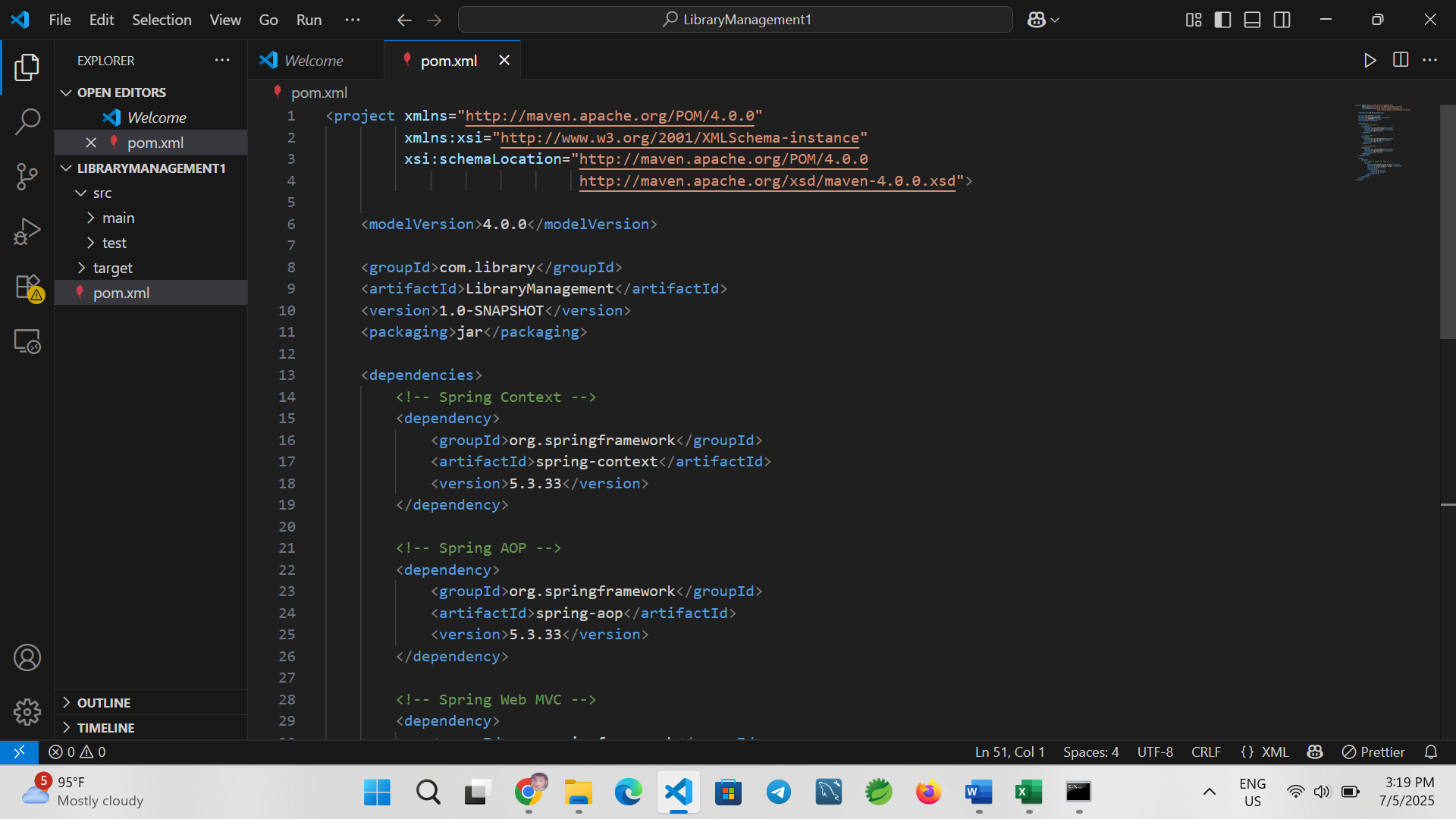
**Step 1: Create the Project Directory Structure**

Open Command Prompt and execute the following commands:

This sets up the standard Maven directory layout.

**Step 2: Create and Configure** ``

Create a file named pom.xml in the root directory of the project and add the following configuration:



**Step 3: Build the Maven Project**

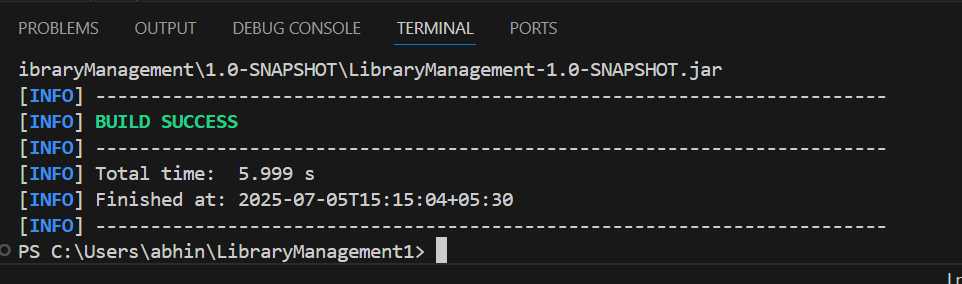
In the terminal, navigate to the project root and run:

mvn clean install

This command will:

* Download the specified dependencies
* Compile the code (if present)
* Verify that the Maven structure and configuration are correct

**Expected Output:**



**5. Conclusion**

In this exercise, we successfully created a new Maven project configured for a Spring-based application. We added dependencies for Spring Context, AOP, and WebMVC modules and configured the Maven Compiler Plugin to use Java 1.8. This setup provides a robust foundation for building a scalable Java application using the Spring Framework.