Cognizant Java FSE – (Deep Skilling)

(WEEK-3)

**MODULE 1:** Spring Core and Maven

**MODULE 2:** Spring Data JPA with Spring Boot, Hibernate

**Submitted by**  
**Name:** SOLLETI VENKATA KUSUMA  
**Roll No:** 111522102145  
**Email:** [22102145@rmd.ac.in](mailto:22102145@rmd.ac.in)

**College:** RMD ENGINEERING COLLEGE  
**Batch:** Java FSE – 2026

**Spring Core and Maven**

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

**Scenario:**

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Steps:**

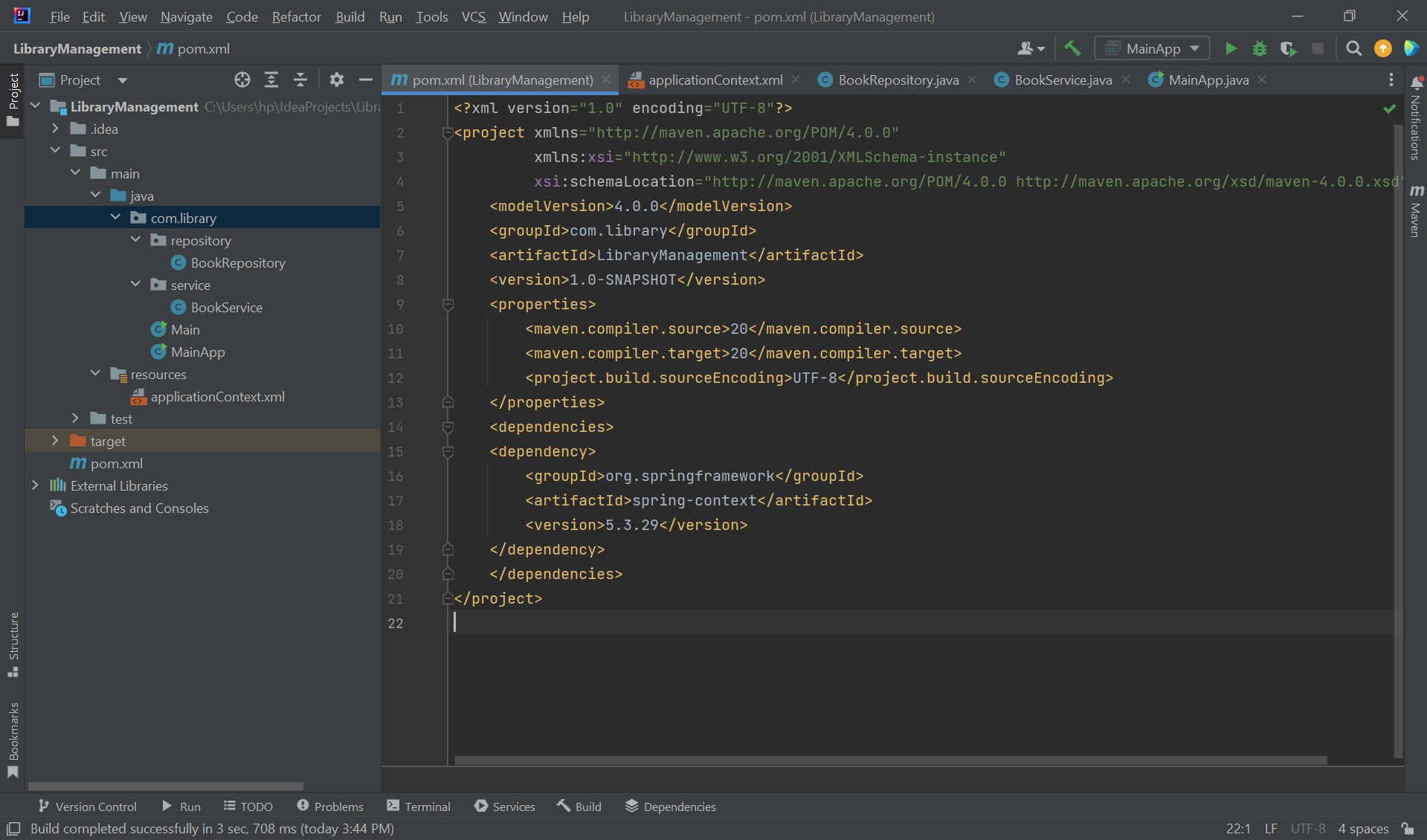
1. **Set Up a Spring Project:**
   * Create a Maven project named **LibraryManagement**.
   * Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   * Create a package **com.library.service** and add a class **BookService**.
   * Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**

Create a main class to load the Spring context and test the configuration

**Step 1: Create a Maven Project**

1. **Open IntelliJ IDEA**
2. Click **File > New > Project**
3. Select **Maven** (left pane)
4. Check **Create from archetype** if you want the basic structure (optional)
5. Click **Next**
6. Enter:
   * **GroupId:** com.library
   * **ArtifactId:** LibraryManagement
7. Click **Next**, then **Finish**

**Step 2: Add Spring Dependency in pom.xml**



**Step 3: Create applicationContext.xml**

* Right-click src/main/resources → **New > File**
* Name it applicationContext.xml

A screenshot of a computer program

AI-generated content may be incorrect.

**Step 4: Create Packages and Classes**

**4.1 BookRepository.java**

* Right-click src/main/java → **New > Package** → name: com.library.repository
* Right-click package → **New > Java Class** → name: BookRepository

A screenshot of a computer

AI-generated content may be incorrect.

**4.2 BookService.java**

* Right-click src/main/java → **New > Package** → name: com.library.service
* Right-click package → **New > Java Class** → name: BookService

A screenshot of a computer program

AI-generated content may be incorrect.

**Step 5: Create MainApp.java**

* Right-click src/main/java → **New > Package** → name: com.library
* Right-click package → **New > Java Class** → name: MainApp

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 2:** Implementing Dependency Injection

**Scenario:**

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**Steps:**

1. Modify the XML Configuration:
   * Update applicationContext.xml to wire BookRepository into BookService.
2. Update the BookService Class:
   * Ensure that BookService class has a setter method for BookRepository.
3. Test the Configuration:
   * Run the LibraryManagementApplication main class to verify the dependency injection.

**Step 1: Create a Maven Project**

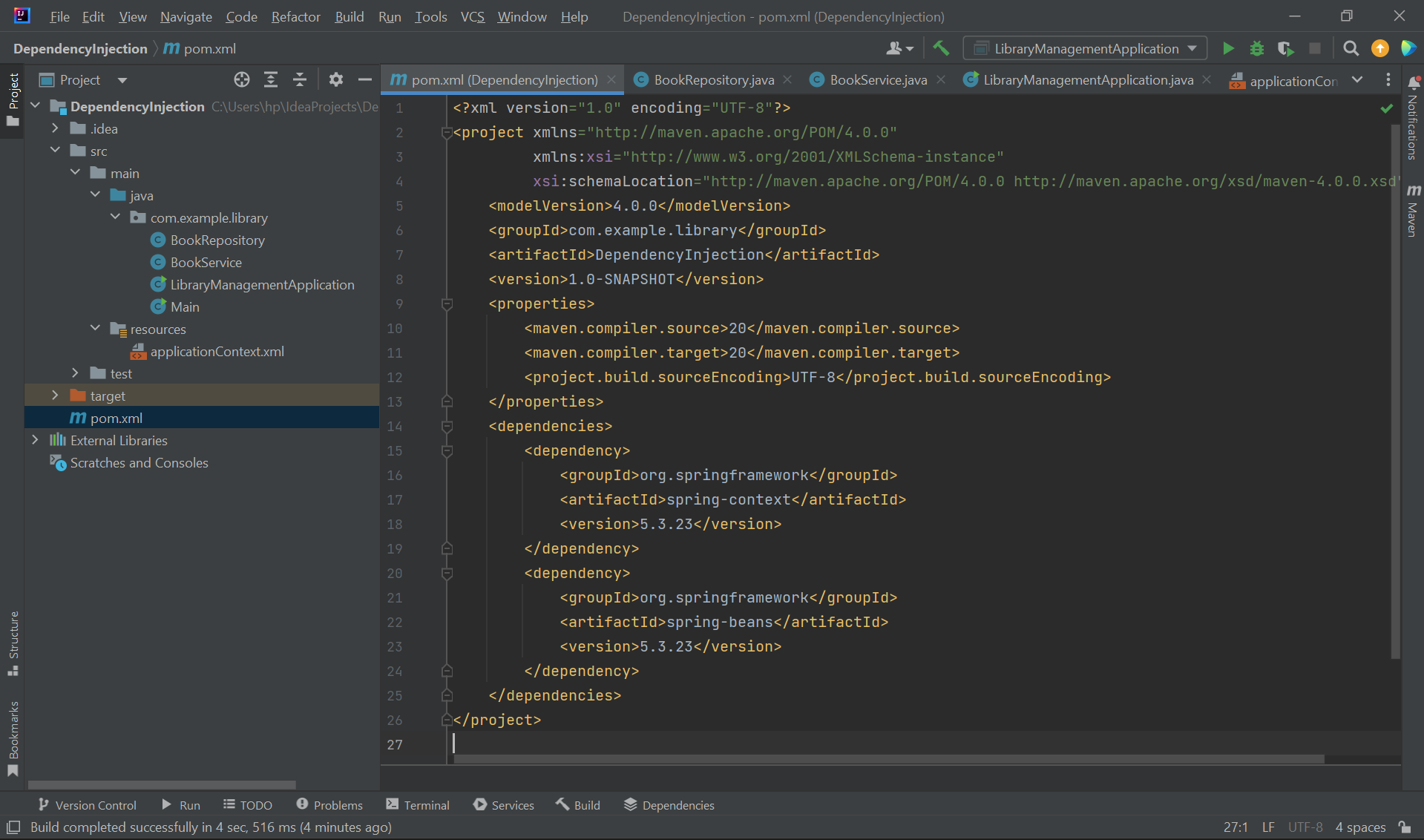
1. **Open IntelliJ IDEA**
2. Click **File > New > Project**
3. Select **Maven**
4. Click **Next**

5.**GroupId:** com.example.library

**ArtifactId:** DependencyInjection

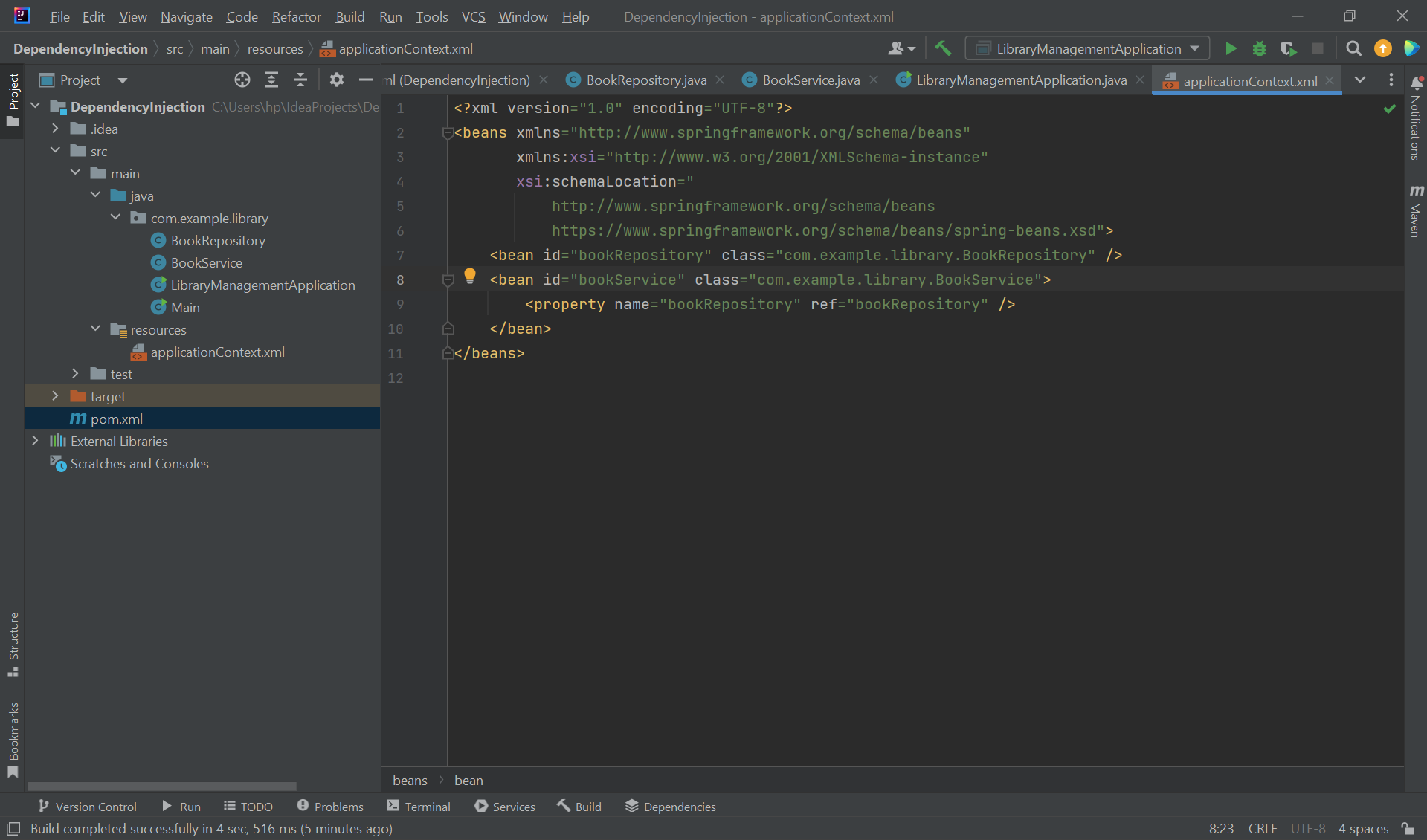
6.Click **Next**, then **Finish**

**Step 2: Add Spring Dependency in pom.xml**



**Step 3: Create applicationContext.xml**

* Right-click src/main/resources → **New > File**
* Name it applicationContext.xml

****

**Step 4: Create java Classes**

1.BookRepository.java  
2.BookService.java  
3.LibraryManagementApplication.java

**A screenshot of a computer

AI-generated content may be incorrect.**

**A computer screen shot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

OUTPUT:

A screen shot of a computer

AI-generated content may be incorrect.

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

**Scenario:**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**Steps:**

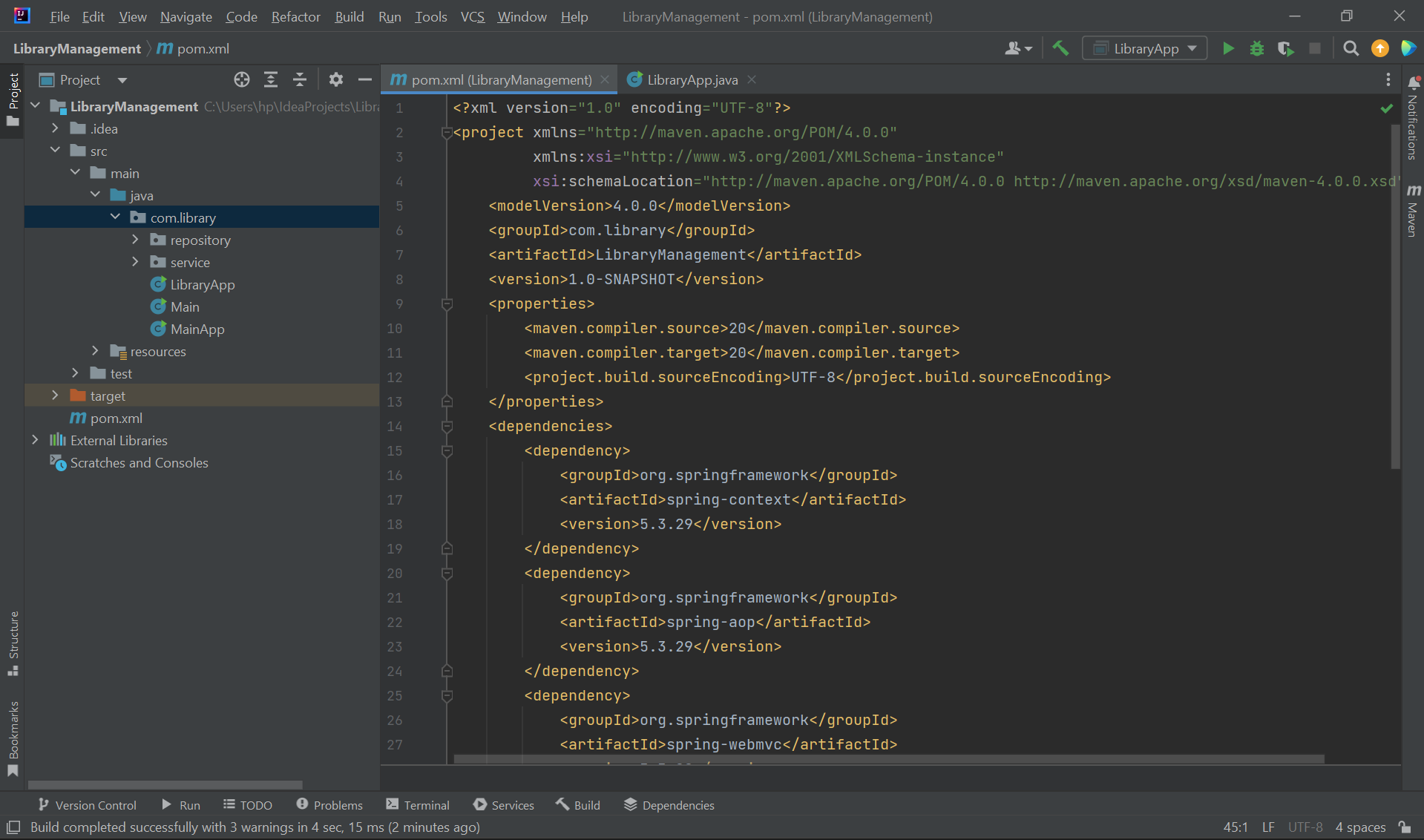
1. **Create a New Maven Project:**
   * Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**Step 1: Create a New Maven Project**

1. Open IntelliJ IDEA.
2. Go to File → New → Project.
3. Select Maven from the left sidebar.
4. Click Next.
5. Enter the project details:
   * GroupId: com.library
   * ArtifactId: LibraryManagement

6.Click Next then Finish.

**Step 2: Add Spring Dependencies to pom.xml**

****

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step 3:Create a sample Java Class LibraryApp.java**

**A screenshot of a computer

AI-generated content may be incorrect.**

OUTPUT:

A black screen with a blue line

AI-generated content may be incorrect.

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

**Scenario:**

The library management application requires a central configuration for beans and dependencies.

**Steps:**

1. **Create Spring Configuration File:**
   * Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   * Define beans for **BookService** and **BookRepository** in the XML file.
2. **Update the BookService Class:**
   * Ensure that the **BookService** class has a setter method for **BookRepository**.
3. **Run the Application:**
   * Create a main class to load the Spring context and test the configuration.

**Step 1: Create a New Maven Project**

1. Open IntelliJ IDEA.
2. Go to File → New → Project.
3. Select Maven from the left sidebar.
4. Click Next.
5. Enter the project details:
   * GroupId: com.library
   * ArtifactId:BookService

6.Click Next then Finish.

**Step 2: Add Spring Dependencies to pom.xml**

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3:Create Java classes BookRepository.java and BookService.java**

**BookRepository.java**

**A screenshot of a computer

AI-generated content may be incorrect.**

**BookService.java**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step 4:Create an applicationContext.xml file**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step 5:Create and run the LibraryApp.java**

**A screenshot of a computer

AI-generated content may be incorrect.**

OUTPUT:

A black screen with a blue line

AI-generated content may be incorrect.

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

**Scenario:**

The library management application requires both constructor and setter injection for better control over bean initialization.

**Steps:**

1. **Configure Constructor Injection:**
   * Update applicationContext.**xml** to configure constructor injection for **BookService**.
2. **Configure Setter Injection:**
   * Ensure that the **BookService** class has a setter method for **BookRepository** and configure it in **applicationContext.xml**.
3. **Test the Injection:**
   * Run the **LibraryManagementApplication** main class to verify both constructor and setter injection.

**Step 1: Create a New Maven Project**

1. Open IntelliJ IDEA.
2. Go to File → New → Project.
3. Select Maven from the left sidebar.
4. Click Next.
5. Enter the project details:
   * GroupId: com.library
   * ArtifactId:Book

6.Click Next then Finish.

**Step 2:Add Spring Dependencies to pom.xml**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Step 3:Create applicationContext.xml file**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Step 4:Create Java Classes**

**i.BookService.java**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**BookRepository.java**

**A screenshot of a computer

AI-generated content may be incorrect.**

**LibraryApp.java**

**A screenshot of a computer

AI-generated content may be incorrect.**

OUTPUT:

**A screenshot of a computer

AI-generated content may be incorrect.**

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

**Scenario:**

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

Steps:

1. Create a Spring Boot Project:
   * Use Spring Initializr to create a new Spring Boot project named LibraryManagement.
2. Add Dependencies:
   * Include dependencies for Spring Web, Spring Data JPA, and H2 Database.
3. Create Application Properties:
   * Configure database connection properties in application.properties.
4. Define Entities and Repositories:
   * Create Book entity and BookRepository interface.
5. Create a REST Controller:
   * Create a BookController class to handle CRUD operations.
6. Run the Application:
   * Run the Spring Boot application and test the REST endpoints**.**

**Step 1: Create a New Maven Project**

1. Open IntelliJ IDEA.
2. Go to File → New → Project.
3. Select Maven from the left sidebar.
4. Click Next.
5. Enter the project details:
   * GroupId: com.library
   * ArtifactId:Library

6.Click Next then Finish.

**Step 2:Add Spring dependencies to pom.xml**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Step 3:Create application.properties file**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Step 4:Create BookRepository.java and Book.java**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Book.java**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screen shot of a computer code

AI-generated content may be incorrect.**

**Step 5:Create a controller class BookController.java**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**A screenshot of a computer program

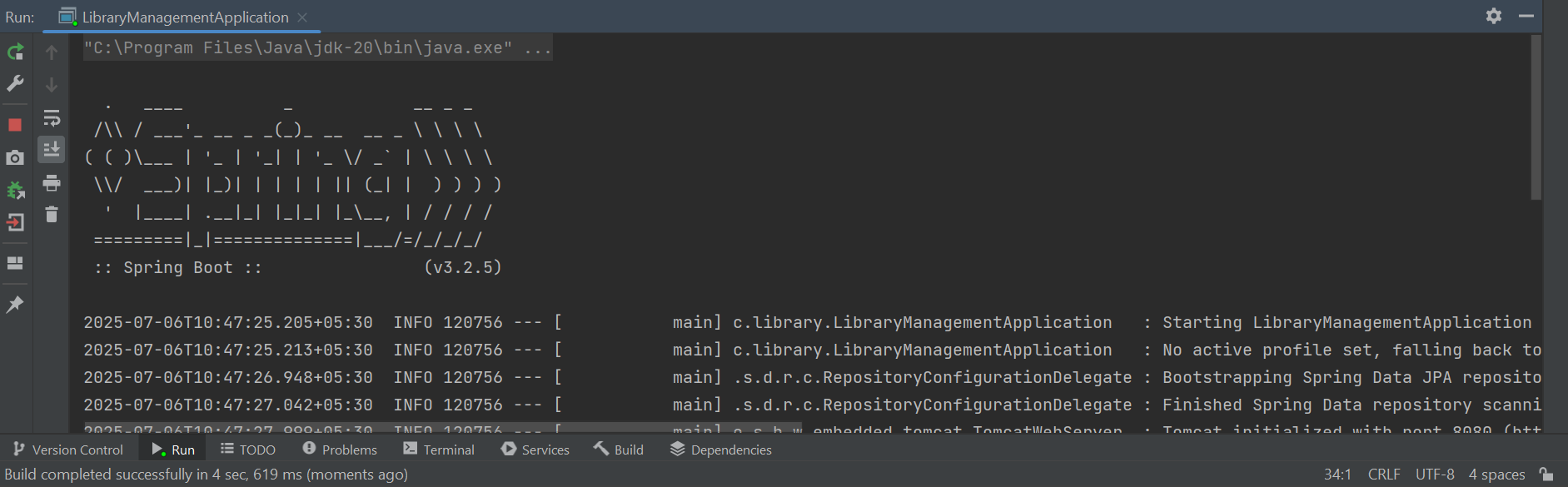
AI-generated content may be incorrect.**

**Step 6:Create and run LibraryManagementApplication.java**

**A screenshot of a computer

AI-generated content may be incorrect.**

OUTPUT:



**Step 7:Test the REST end points**

**A screenshot of a computer

AI-generated content may be incorrect.**