**Exercise 6: Configuring Beans with Annotations**

Scenario:

**1. Introduction**

The Library Management System is built using the Spring Framework, which simplifies Java application development by providing comprehensive infrastructure support. In this exercise, the objective is to simplify the configuration of beans using annotations, which helps to reduce the XML configuration and makes the code more readable and maintainable.

**2. Updates Made**

**2.1. Enabling Component Scanning**

To enable component scanning, the applicationContext.xml file was updated to include the context:component-scan element. This instructs Spring to scan the specified package (com.library) for classes annotated with Spring's stereotype annotations such as @Component, @Service, @Repository, and @Controller.

**2.2. Annotating Classes**

To leverage Spring’s annotation-based configuration:

* **BookService Class:** The BookService class was annotated with @Service to indicate that it's a service component in the Spring context.
* **BookRepository Class:** The BookRepository class was annotated with @Repository to indicate that it's a repository component.

**2.3. Removing XML Bean Definitions**

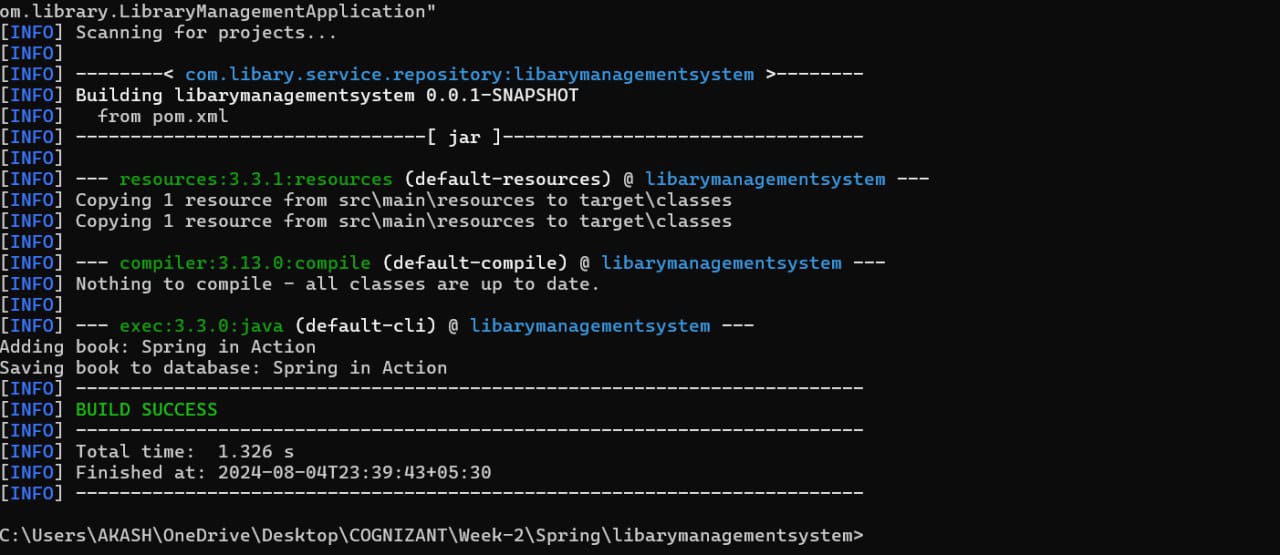
Since the classes are now annotated, the explicit bean definitions in applicationContext.xml for BookService and BookRepository were removed, simplifying the XML configuration.

**Testing the Configuration**

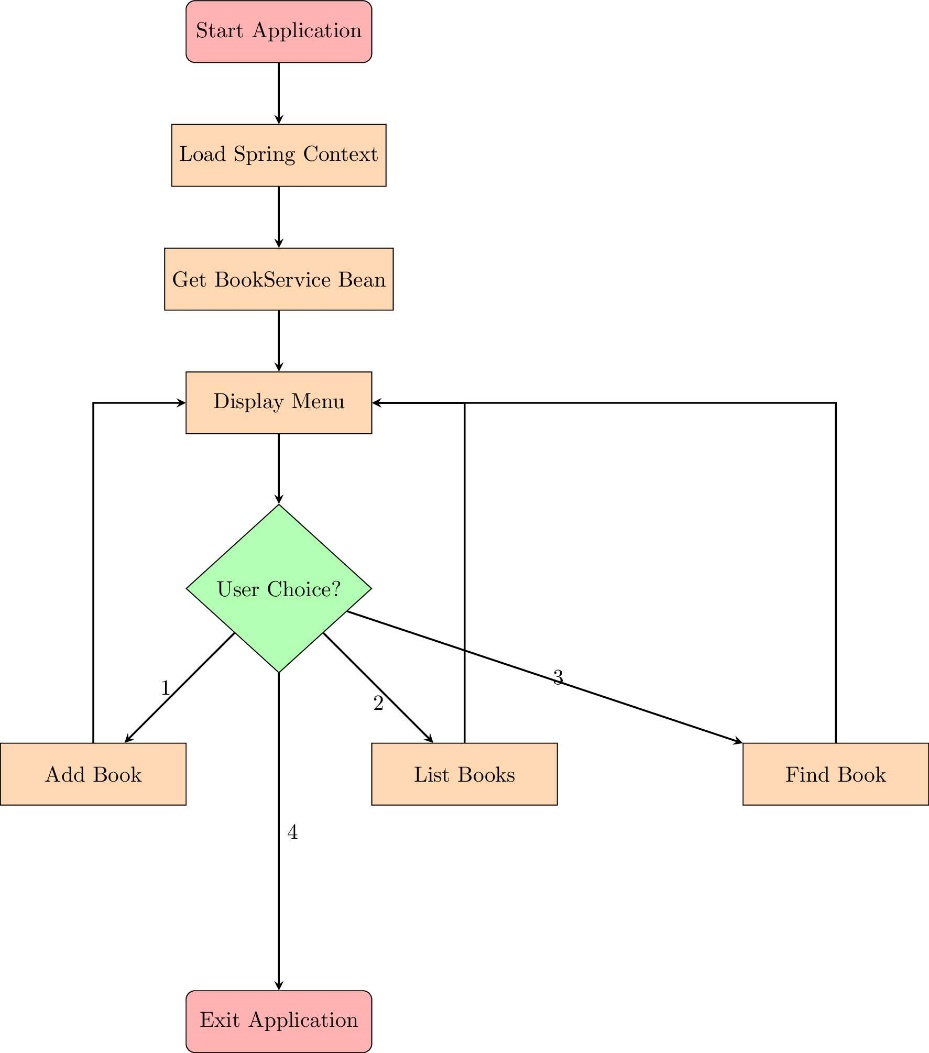
To test the updated configuration:

* Run the LibraryManagementApplication main class.

**OUTPUT :**



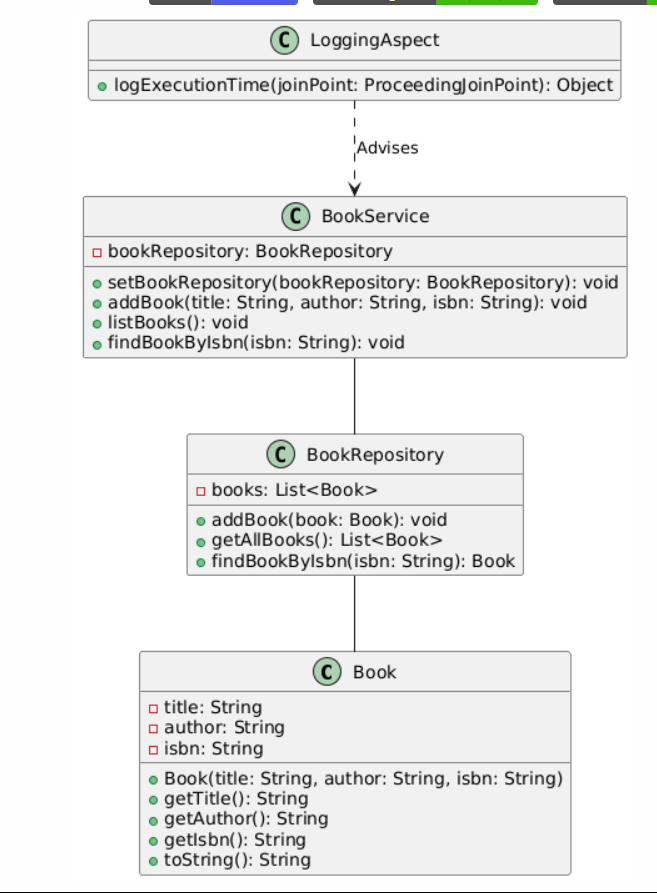
**FLOWCHART of the program :**



The flowchart visually represents the sequence of operations in the Library Management System:

* **Start Application:** The application starts and loads the Spring context using applicationContext.xml.
* **Load Spring Context:** Spring scans the com.library package and automatically wires the beans using annotations.
* **Get BookService Bean:** The BookService bean is retrieved from the Spring context.
* **Display Menu:** The application displays a menu to the user for various operations like adding a book, listing all books, or searching for a book by ISBN.
* **User Choice:** The user selects an option, which directs the flow to the corresponding action (Add Book, List Books, or Find Book).
* **Add Book:** If the user chooses to add a book, the book is added to the repository.
* **List Books:** If the user chooses to list books, all books in the repository are displayed.
* **Find Book:** If the user chooses to search for a book, the book is retrieved from the repository by ISBN.
* **Exit Application:** The application terminates if the user selects the exit option.

**CLASS DIAGRAM :**



The class diagram provides a high-level view of the Library Management System’s structure:

* **Book Class:** Represents a book with attributes like title, author, and ISBN.
* **BookRepository Class:** Manages a list of books and provides methods to add, retrieve, and search books.
* **BookService Class:** Acts as the service layer, handling business logic related to books and interacting with BookRepository.
* **LoggingAspect Class:** Implements an aspect that logs the execution time of methods in BookService. It leverages Spring AOP to intercept method calls and log performance data.

The relationships are depicted as:

* **BookRepository** depends on **Book**.
* **BookService** depends on **BookRepository**.
* **LoggingAspect** advises **BookService**.