Question

Case Study: Library Management System

This case study details a simple web application designed to manage a library's books with two
distinct user roles: a regular user and an admin. The system is built using Spring MVC and Spring Security with XML configuration.

Project Components and Purpose

The application's logic is distributed across several key files, each with a specific responsibility:

• pom.xml: This file, a standard for Maven projects, lists all the necessary software libraries (dependencies) for the application to function. It includes the core Spring MVC framework and the Spring Security libraries, which provide the security features.

• web.xml: This is the application's deployment descriptor. It acts as the central router, telling the server to use Spring's DispatcherServlet to handle all incoming web requests and to apply the springSecurityFilterChainto enforce security rules on every request.

• spring-servlet.xml: This file configures the core of the Spring application. It tells Spring where to find the application's controller classes (the Java code that handles web requests) and how to resolve the names of JSP pages into full file paths. It also importantly imports the security configuration file.

• spring-security.xml: This is where all the security rules are defined. It's the central hub for access control, user authentication, and defining user roles.

• BookController.java: This is the Java class that contains the application's logic. It handles different URLs like /, /login, /books, and /admin, returning the name of the corresponding JSP page to be displayed.

• JSP Pages: These are the visual templates for the application. There are pages for the public
home page (home.jsp), the login form (login.jsp), the list of books (books.jsp), and the restricted admin panel (admin.jsp).

Security Rules and User Roles

The core of this case study is the different levels of access granted to each user role.

• ROLE_USER: This is the standard role for library patrons. A user with this role can log in and view the list of available books. Their username is user and their password is userpass.

• ROLE_ADMIN: This is the administrative role. An admin user can log in and view the list of books, but they also have access to a special "Admin Panel" page that regular users cannot
see. This user has both the ROLE_USER and ROLE_ADMIN authorities. The username is admin and the password is adminpass.

The spring-security.xml file implements these rules:

• The / (home) and /login URLs are open to everyone (permitAll).
• The /books URL is accessible to anyone with either ROLE_USER or ROLE_ADMIN (hasAnyRole).
• The /admin URL is strictly for users with ROLE_ADMIN (hasRole).

The system uses in-memory authentication, which means the user accounts and passwords are
hardcoded directly into the spring-security.xml file for this example.

Application Flow

1. Public Access: When an unauthenticated person first visits the library's main page (/), they
see the welcome page (home.jsp) and have options to view books or log in.

2. Navigating to Protected Pages: If they try to go directly to the /books or /admin page, Spring Security will intercept the request and automatically redirect them to the /login page.

3. Authentication:

◦ If they log in with the user/userpass credentials, they are successfully authenticated and redirected to the /books page. From there, they can see the list of books and an option to log out. If they try to access the /admin page manually, they will be blocked and receive a "403 Forbidden" error.

◦ If they log in with the admin/adminpass credentials, they are also redirected to the /books page. However, because they have the ROLE_ADMIN authority, they will also see an additional link to the "Admin Panel" on the books page. They can

access both the /books and /admin pages without any issues.

4. Logout: When a logged-in user clicks the "Logout" button, their session is invalidated, and they are redirected back to the public home page. This ensures that their access to protected
resources is revoked.

Code

```
// pom.xml (Dependencies)
<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.example</groupId>
  <artifactId>library-management</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>

  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.3.30</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-web</artifactId>
      <version>5.7.11</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-config</artifactId>
      <version>5.7.11</version>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>
  </dependencies>
```

```
</project>

// web.xml
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
      version="3.1">

  <display-name>Library Management</display-name>

  <servlet>
    <servlet-name>spring</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>spring</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

  <filter>
    <filter-name>springSecurityFilterChain</filter-name>
    <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>springSecurityFilterChain</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>

// spring-servlet.xml
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
     http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
     http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd
```

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

   <context:component-scan base-package="com.example.library.controller"/>
   <mvc:annotation-driven/>
   <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
      <property name="prefix" value="/WEB-INF/views/"/>
      <property name="suffix" value=".jsp"/>
   </bean>
   <import resource="spring-security.xml"/>
</beans>

// spring-security.xml
<beans:beans xmlns="http://www.springframework.org/schema/security"
         xmlns:beans="http://www.springframework.org/schema/beans"
         xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
         http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">

   <http auto-config="true">
      <intercept-url pattern="/" access="permitAll"/>
      <intercept-url pattern="/login" access="permitAll"/>
      <intercept-url pattern="/books" access="hasAnyRole('ROLE_USER','ROLE_ADMIN')"/>
      <intercept-url pattern="/admin" access="hasRole('ROLE_ADMIN')"/>
      <form-login login-page="/login" default-target-url="/books" authentication-failure-
url="/login?error=true"/>
      <logout logout-success-url="/"/>
   </http>

   <authentication-manager>
      <authentication-provider>
         <user-service>
            <user name="user" password="{noop}userpass" authorities="ROLE_USER"/>
            <user name="admin" password="{noop}adminpass"
authorities="ROLE_USER,ROLE_ADMIN"/>
         </user-service>
      </authentication-provider>
   </authentication-manager>
</beans:beans>

// BookController.java
package com.example.library.controller;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class BookController {

   @GetMapping("/")
   public String home() {
      return "home";
   }

   @GetMapping("/login")
   public String login() {
      return "login";
   }

   @GetMapping("/books")
   public String books() {
      return "books";
   }

   @GetMapping("/admin")
   public String admin() {
      return "admin";
   }
}

// home.jsp
<html>
<body>
<h2>Welcome to the Library</h2>
<a href="books">View Books</a> | <a href="login">Login</a>
</body>
</html>

// login.jsp
<html>
<body>
<h2>Login Page</h2>
<form method="post" action="login">
   Username: <input type="text" name="username"/> <br/>
   Password: <input type="password" name="password"/> <br/>
```

```
   <input type="submit" value="Login"/>
</form>
</body>
</html>

// books.jsp
<html>
<body>
<h2>Book List</h2>
<ul>
   <li>Book 1</li>
   <li>Book 2</li>
   <li>Book 3</li>
</ul>
<a href="logout">Logout</a>
</body>
</html>

// admin.jsp
<html>
<body>
<h2>Admin Panel</h2>
<p>Only admins can see this page.</p>
<a href="books">Back to Books</a> | <a href="logout">Logout</a>
</body>
</html>
```