Writing security test cases in Java involves ensuring that your application is protected against various security threats, such as SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF). Below is an example of how you can write security test cases using **JUnit 5** (for unit testing) along with **Spring Security** and **MockMvc** to simulate and test security-related scenarios in a Spring Boot application.

1. **SQL Injection Protection Test**
2. **XSS Protection Test**
3. **CSRF Protection Test**
4. **Authentication and Authorization Test**
5. **Secure Headers Test**

**1. SQL Injection Protection Test**

SQL injection can occur when user input is improperly handled, allowing attackers to execute arbitrary SQL queries.

Code sample git URL – <https://github.com/nikhildesale12/Java-Security-Testcases/blob/main/SqlInjectionTest.java>

In this test, the test simulates an SQL injection attempt by passing a malicious input in the username and password fields. The expected behavior is for the application to reject the request (status 400 Bad Request) because of invalid input.

**2. XSS Protection Test**

Cross-Site Scripting (XSS) occurs when malicious scripts are injected into content served to users. The following test ensures that any user input is sanitized properly.

Code sample git URL – <https://github.com/nikhildesale12/Java-Security-Testcases/blob/main/SqlInjectionTest.java>

In this case, we simulate a user submitting a comment with a script embedded. The expected outcome is that the script is sanitized, and the user is presented with the sanitized version of the input.

**3. CSRF Protection Test**

Cross-Site Request Forgery (CSRF) is a type of attack where a malicious user could trick a logged-in user into making unwanted requests. Spring Security provides CSRF protection by default.

Code sample git URL – <https://github.com/nikhildesale12/Java-Security-Testcases/blob/main/CsrfProtectionTest.java>

The first test checks for CSRF protection by making a request without a valid CSRF token, which should result in a 403 Forbidden response. The second test simulates a request with a valid CSRF token, which should succeed.

**4. Authentication and Authorization Test**

This test checks that users can only access certain endpoints if they are authenticated and authorized.

Code sample git URL – <https://github.com/nikhildesale12/Java-Security-Testcases/blob/main/AuthenticationAndAuthorizationTest.java>

In this test:

* The first case checks that an unauthenticated user cannot access a protected endpoint (/api/user/profile).
* The second case checks that an authenticated user can access the endpoint if they provide a valid JWT token.
* The third case ensures that a user with insufficient privileges (e.g., a regular user trying to access and admin-only endpoint) gets a 403 Forbidden response.

**5. Secure Headers Test**

This test ensures that security headers, like X-Content-Type-Options, X-Frame-Options, and others, are properly set.

Code sample git URL – <https://github.com/nikhildesale12/Java-Security-Testcases/blob/main/SecureHeadersText.java>

This test checks that specific security headers, such as X-Content-Type-Options and X-Frame-Options, are included in the HTTP response.