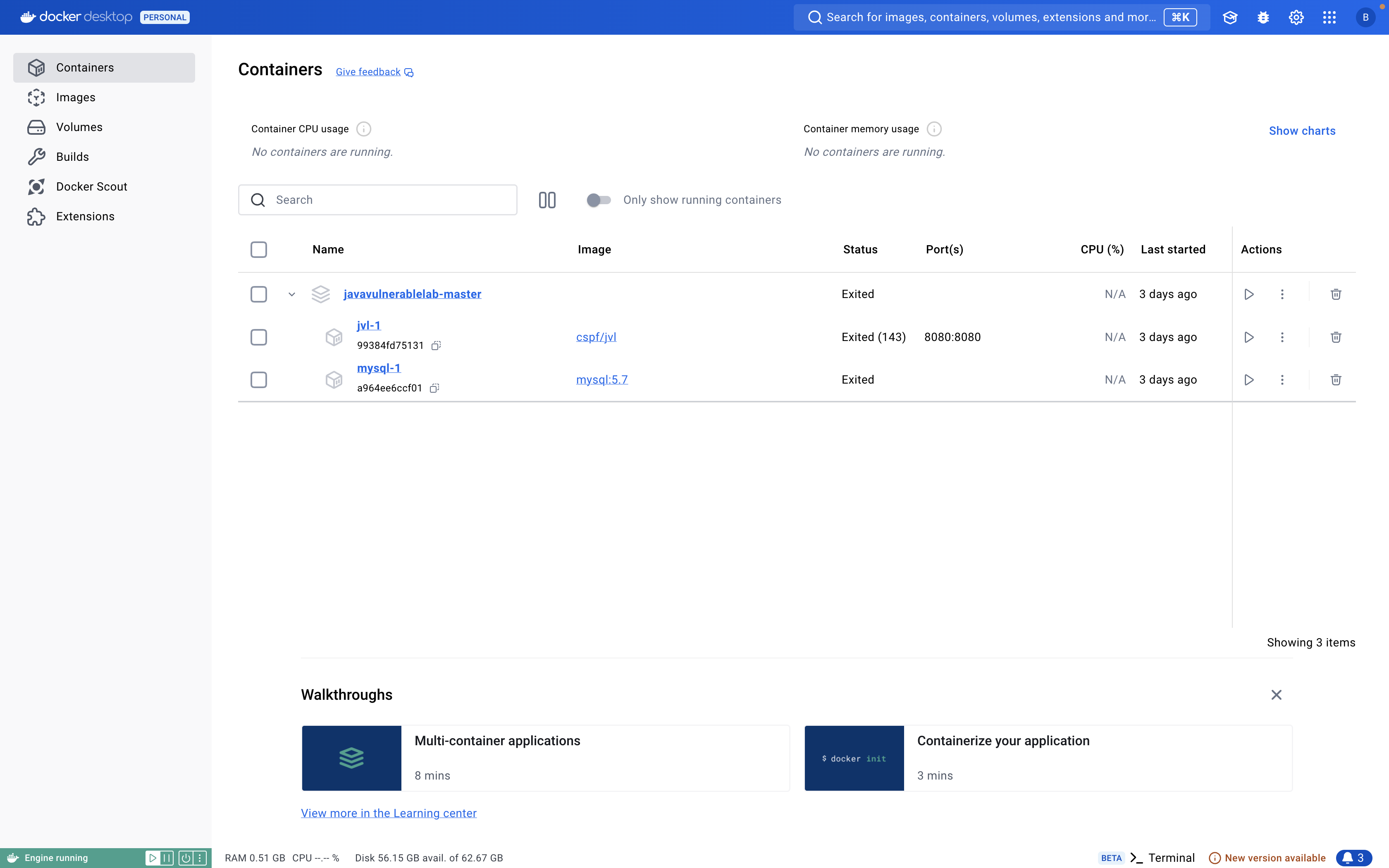
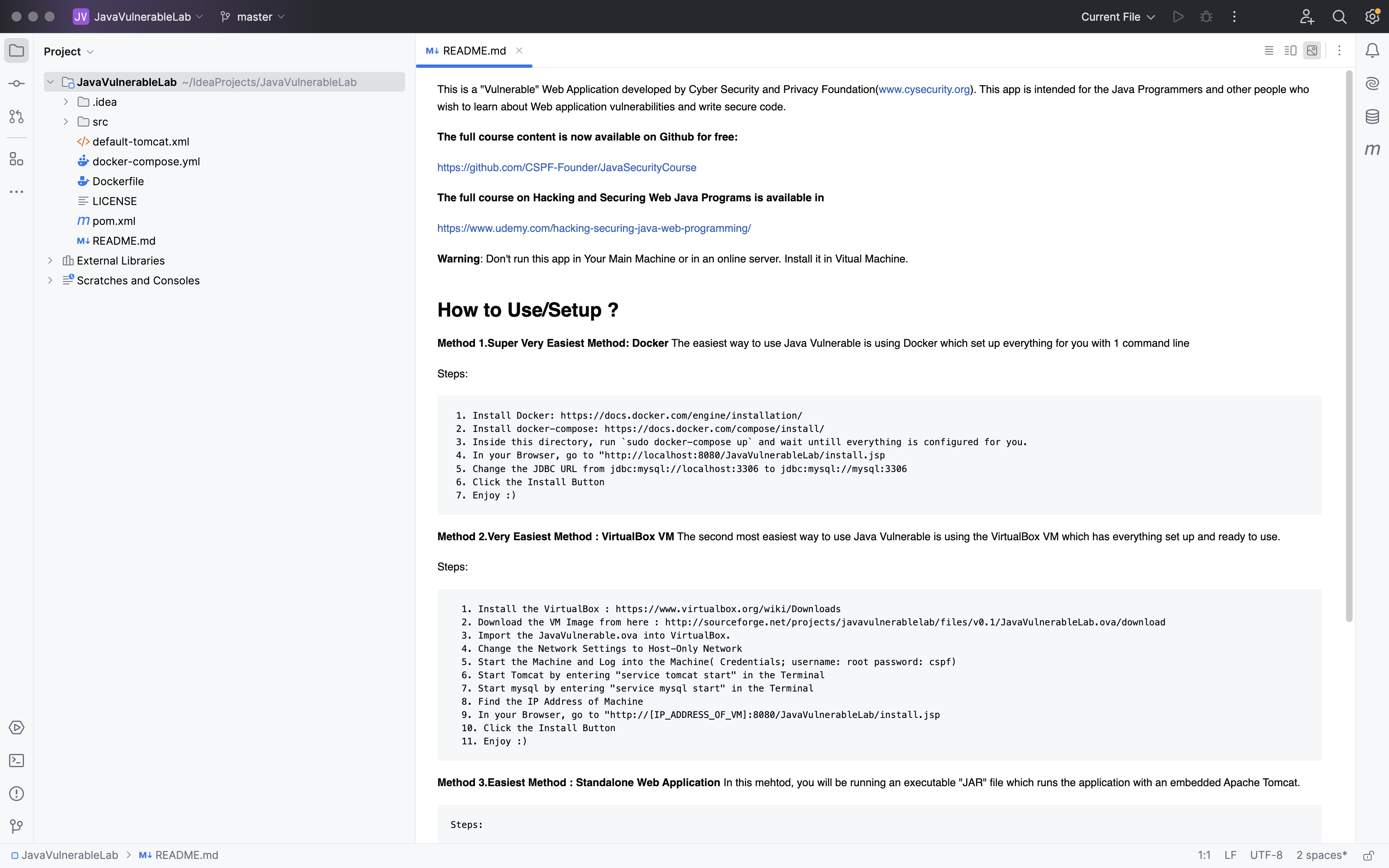
Brittney Karahalios

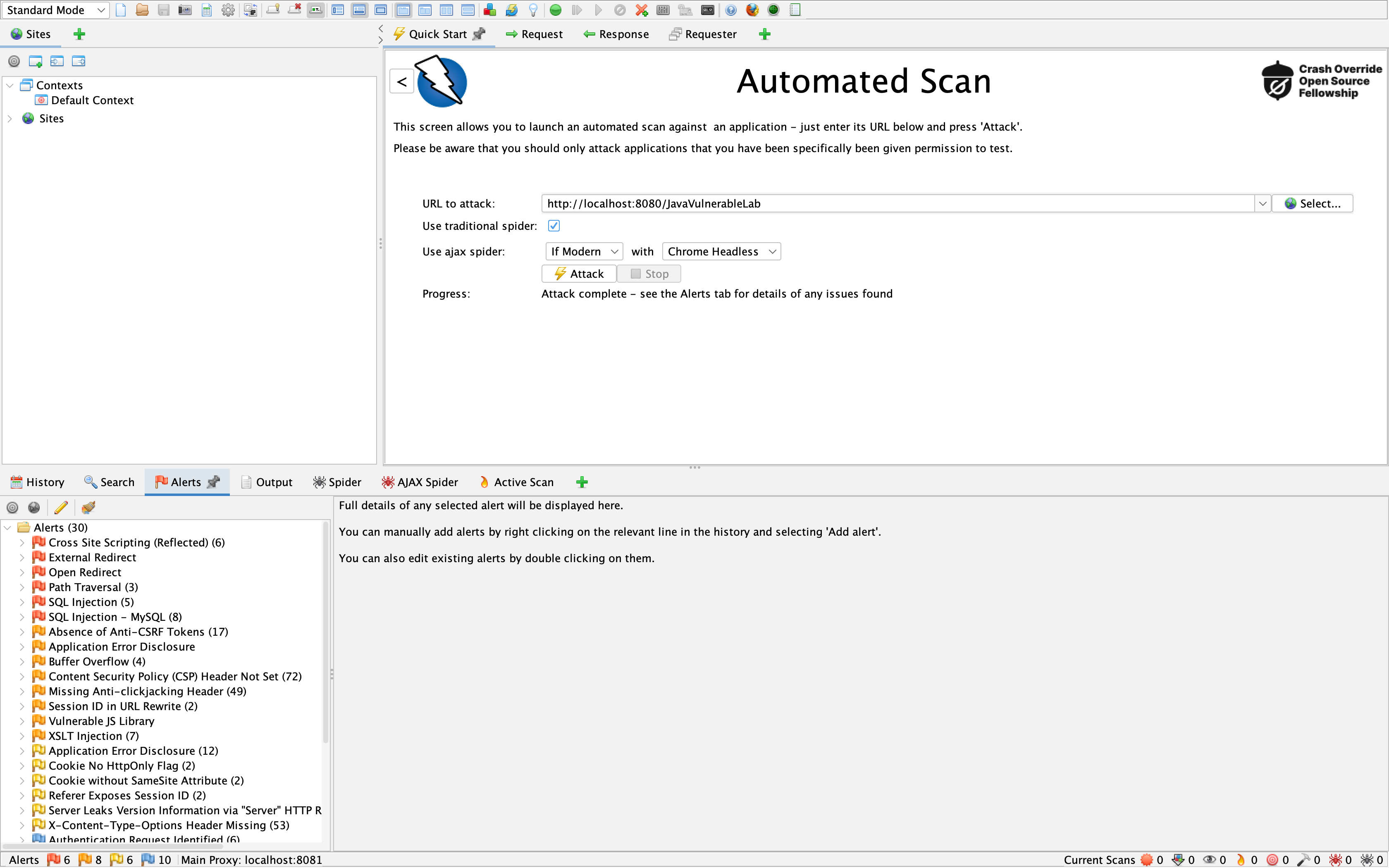
CSC1029

SQL Injection Project 2

10/07/24

1. Install Java Vulnerable Lab (JVL) in Docker.

2. Open the project using IntelliJ.

3. Use OWASP ZAP or similar tool to scan JVL for vulnerabilities/exploits. Generate a Report using ZAP.

4. Analyze the report and figure out how to mitigate the vulnerability/exploit.

One of the main vulnerabilities/exploits is that the code allows for SQL Injection to happen when username/password is incorrect. To mitigate this vulnerability I will rewrite the code in the Forgot Password file, utilizing PreparedStatement to prevent SQL injection from occurring.

Before:

{

Connection con=new DBConnect).connect(aetServletContext().aetRealpath(“/WEB-INE/confia.orooerties")):

ResultSet rs=null;

Statement stmt = con.createstatement();

rs=stmt.executeQuery("select \* from users where username=‘ "+request.getParameter("username") .trim()+”I and secret=+request.getParameter("secret")+"''):

if(rs != null && rs.next()){

out.print(“Hello+rs.getString("username")+ <bb class=" success’> Your Password is: +rs.getstring(“password));

}

else

{

out.print("<b class='fail'> Secret/Email is wrong</b>”)

}

}

After:

*//More secure PreparedStatement for username*

String username = request.getParameter("username");

String query = ("SELECT \* FROM users WHERE username = ? ");

PreparedStatement pstmt = connection.prepareStatement(query);

pstmt.setString(1, username);

ResultSet results = pstmt.executeQuery();

*//More secure PreparedStatement for secret*

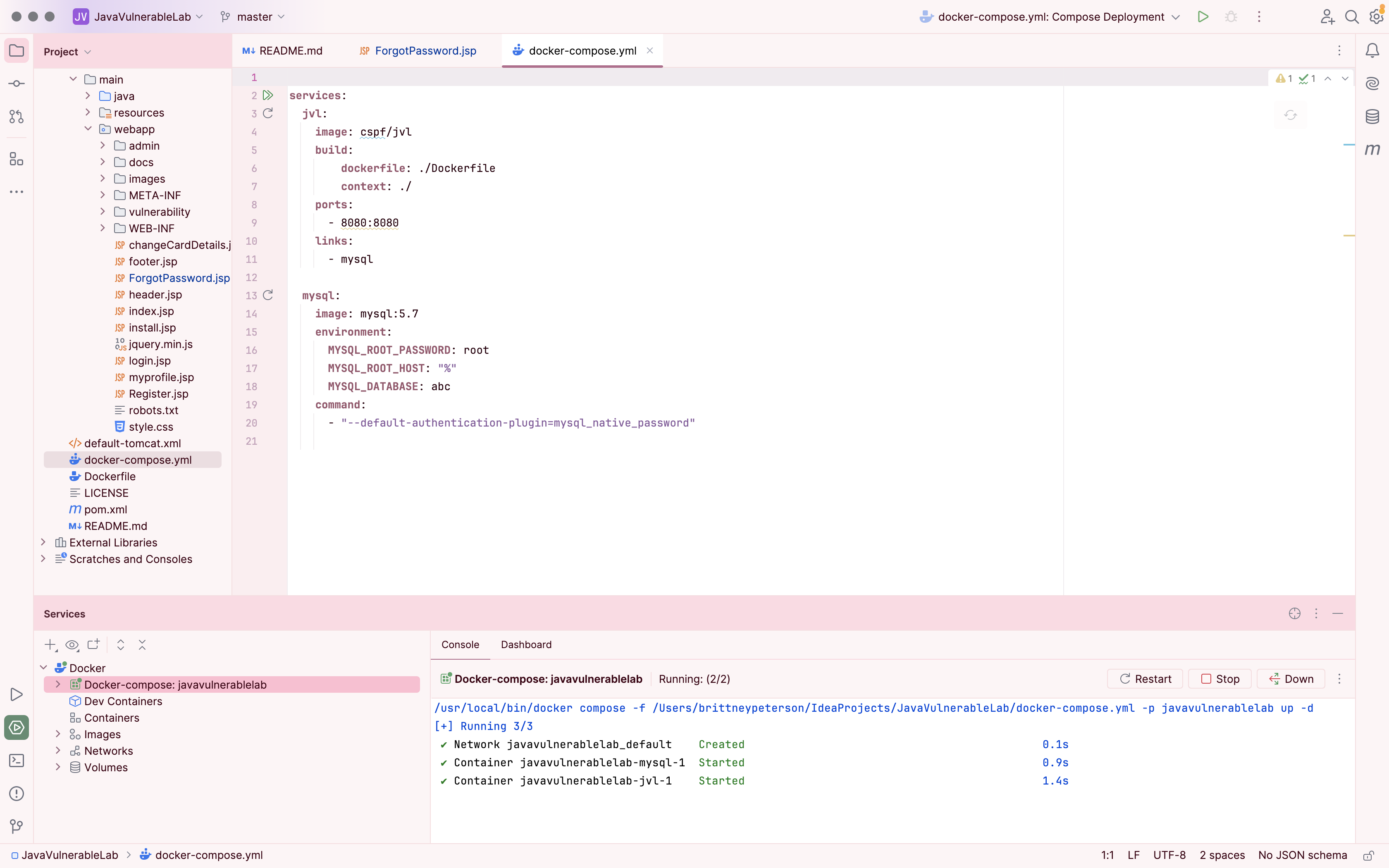
String secret = request.getParameter("secret");

String query2 = ("SELECT \* FROM secret WHERE secret = ? ");

PreparedStatement pstmt2 = connection.prepareStatement(query2);

pstmt2.setString(2, secret);

ResultSet results2 = pstmt2.executeQuery();

5. Make the required changes using IntelliJ and push the code back to Docker.

6. Run a scan of JVL and confirm the vulnerability/exploit has been mitigated (because it doesn't appear on the report).

Prepared statements were utilized in order to address the vulnerability as it relates to the ForgotPassword page. The vulnerability has been mitigated and the use of prepared statements prevents SQL injection attacks from being utilized in this particular area of the program.