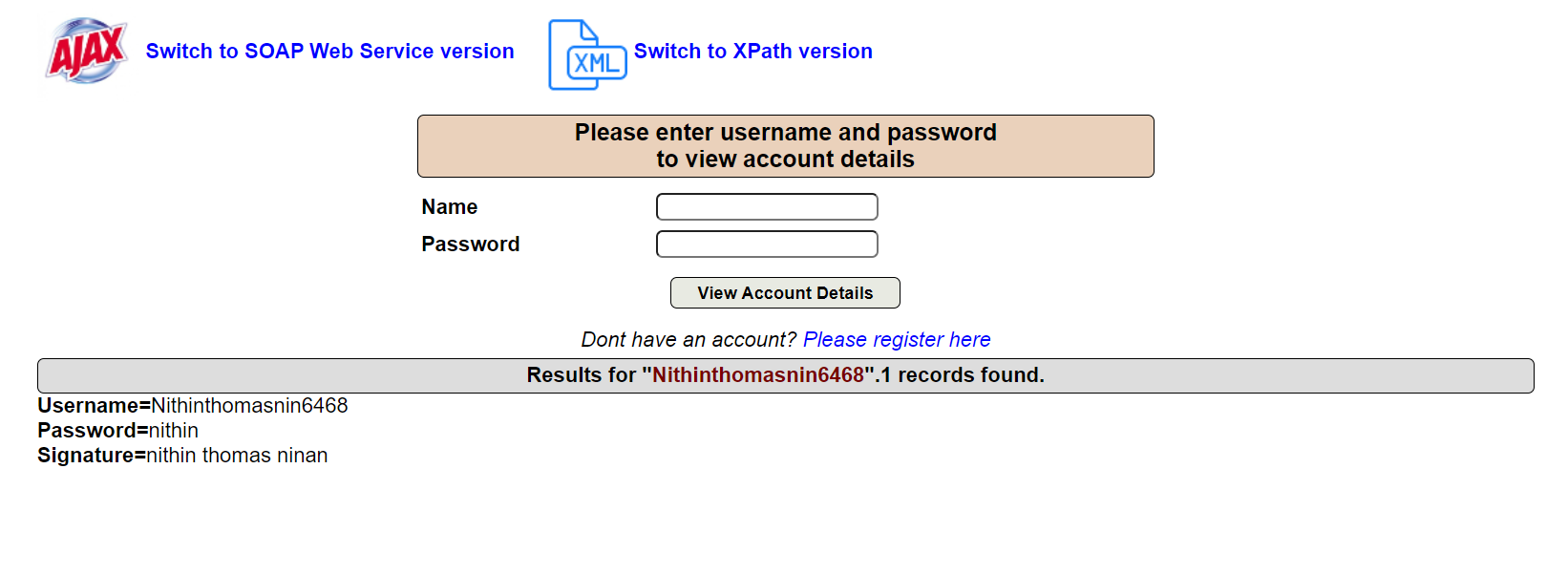
**NITHIN THOMAS NINAN ERUTHICKAL**

**8866468**

**Assignment #2**

**28-Jan-2024**

**Task1**

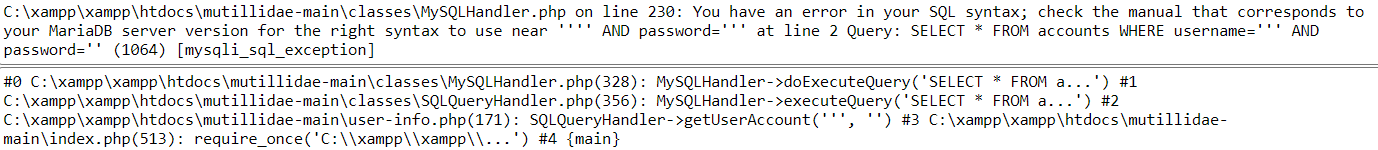
We have created an account in Multillidae and signed in to the account using our name and password. We can see the created account's Username, password, and Signature.

**Task2**

We put an (') in the name field to break the query. Next, we click on "view account details" to see the error message.

**Task3**

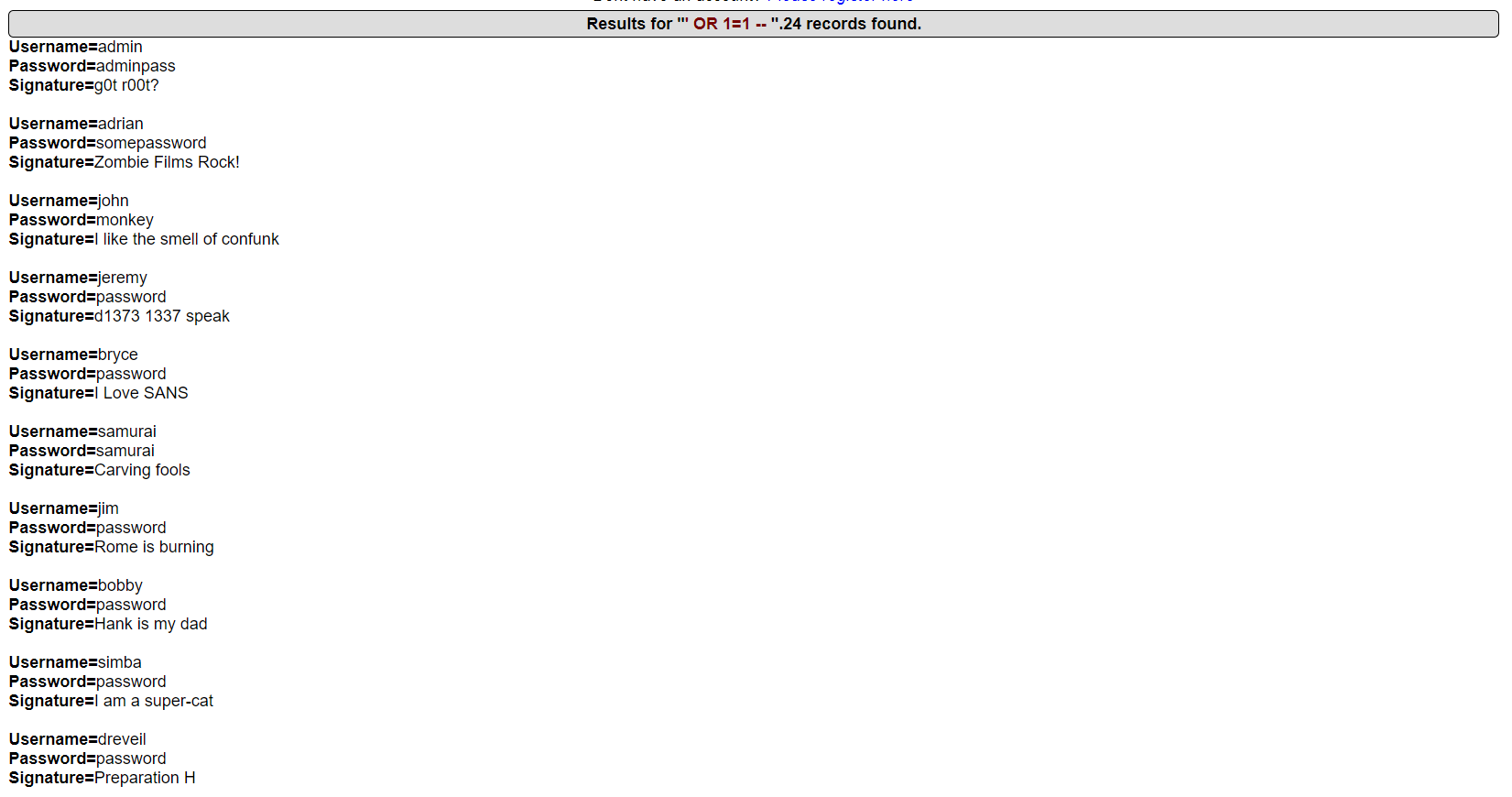
The table name and SQL query are visible.

"SELECT \* FROM accounts WHERE username=''' AND password=' '" is the query. "Accounts" is the table's name referenced in the query.

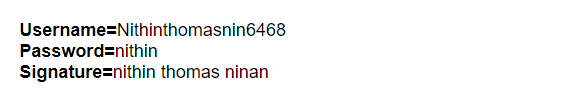
**Task4**

In this task, we execute an SQL injection attack using just the username field. And the name of the table from the previous task. All user account details are displayed in this table once the query has been executed.

The SQL injection attack uses the query "'OR 1=1 ––."

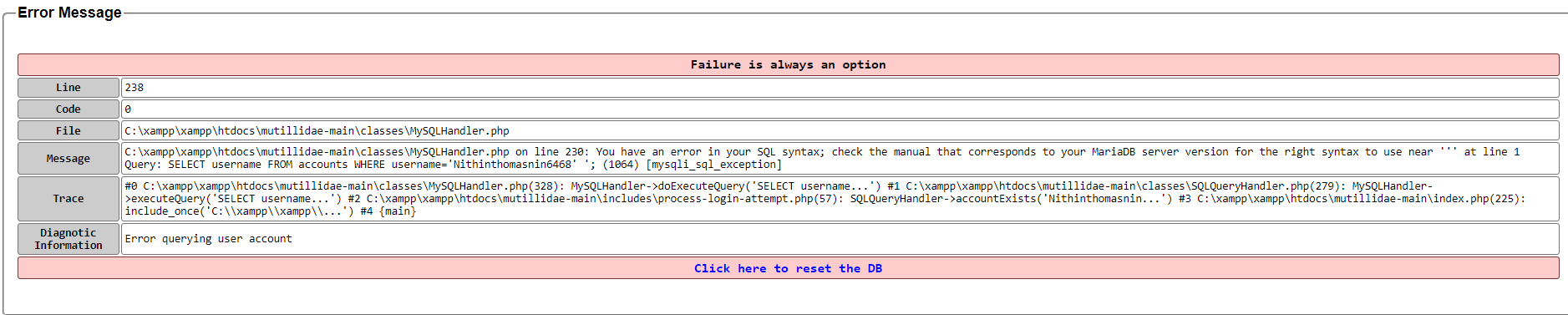
24 records have been found

Specific account highlighted

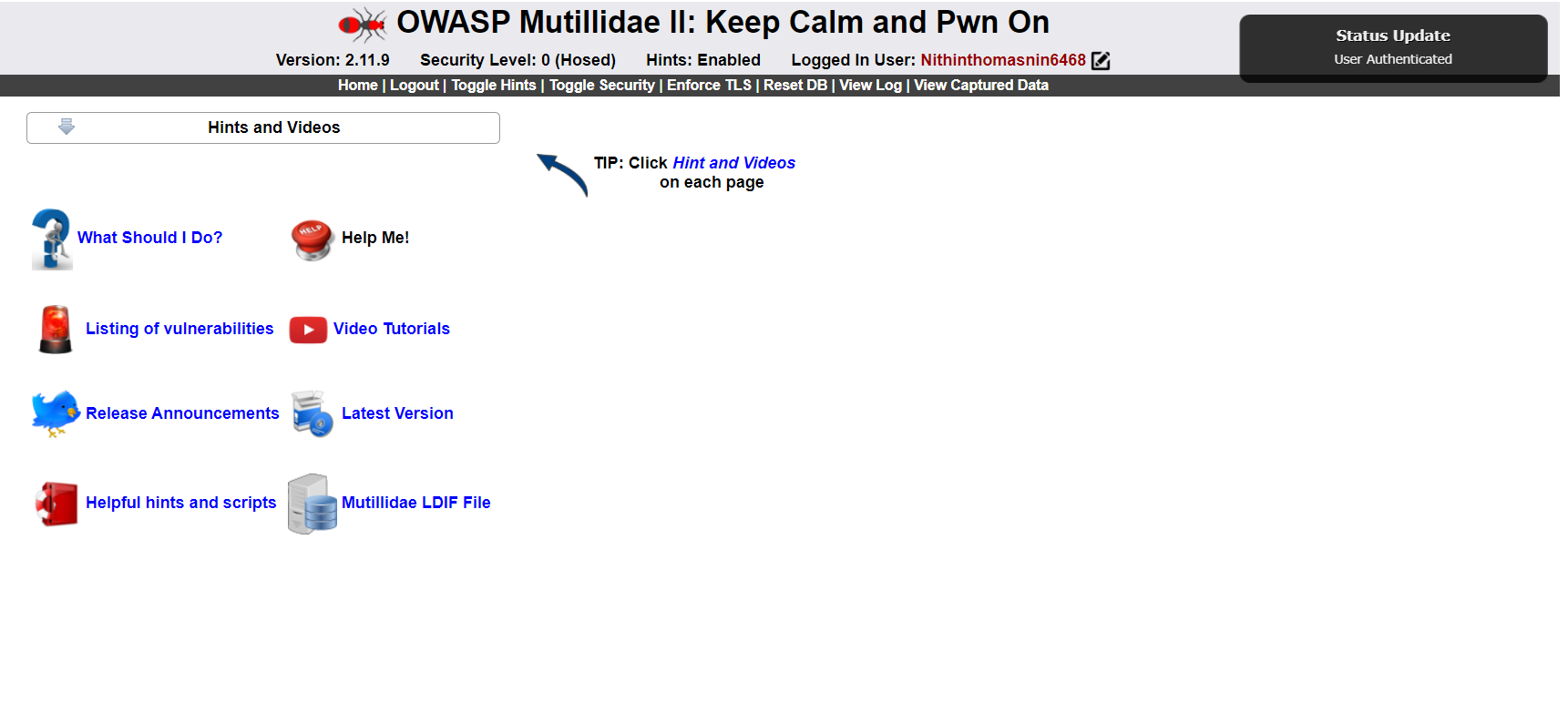


**Task5**

Go into OWASP 2017 | A1 - Injection (SQL) | SQLi - Bypass Authentication | Login: and enter only Username and apostrophe (').

This is the error message that is seen.

After entering the username "Nithinthomasnin6468 -" in the username field and signing in again, we can see the Logged User name and a pop-up message indicating "user Authenticated" in the upper right corner.

****

**Task 6**

What worked well:

* Creating an account in Mutillidae and signing in using the provided credentials worked well.
* Displaying error messages correctly indicating the affected query

What didn’t work well and how I resolved it:

In task 4, the initial SQL injection attempt using the query 'OR 1=1 -- did not successfully bypass authentication, resulting in an error message. The issue was due to the spacing at the end of the query. It worked correctly when I added the space.

Thing I learned which I didn’t know prior to starting

* In task 3, I gained knowledge of the format of the SQL query that the system ran, including the table name and the columns utilized for the authentication procedure.
* Importance of using different SQL injection techniques.

What can be done to prevent such attacks?

By employing parameterized queries, input validation, and safe coding practices, applications can lower their risk of SQL injection vulnerabilities and prevent such attacks.