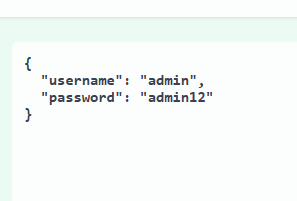
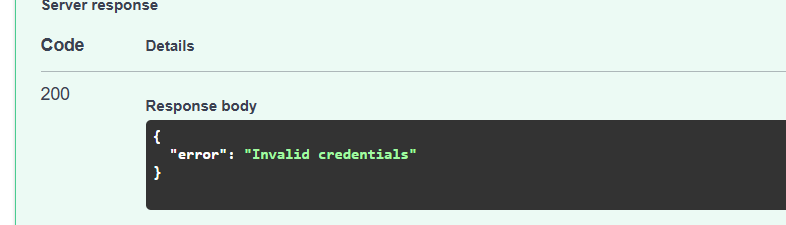
# SQL Injection Vulnerability - Demonstration Report

This report demonstrates a SQL Injection vulnerability in a FastAPI login system using SQLite. The vulnerability occurs because raw SQL queries are used with direct user input.

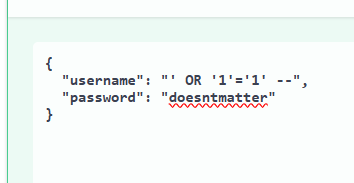
## 1. Normal Login Attempt with Incorrect Credentials



The user attempts to login using the username 'admin' and password 'admin12'. Since the credentials are invalid or do not match, the server responds with an error.

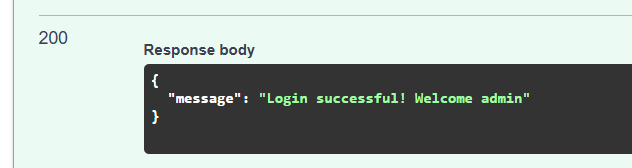


## 2. SQL Injection Attempt



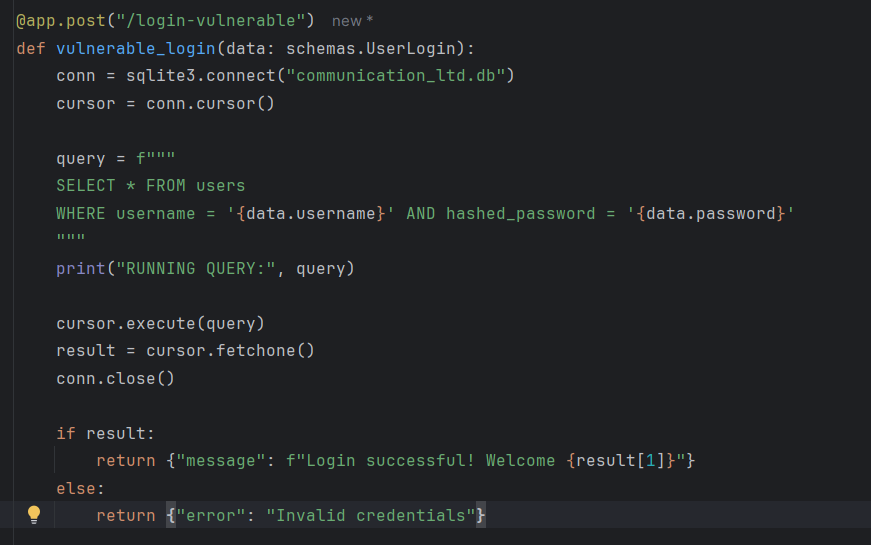
The user sends a malicious input: `' OR '1'='1' --` in the username field. This input manipulates the SQL query to always evaluate as true, bypassing authentication checks.

## 3. SQL Injection Succeeds



Despite entering incorrect credentials, the system returns a successful login message. This confirms that the system is vulnerable to SQL injection attacks.

## 4. Vulnerable Code



This code shows the vulnerable implementation. User input is inserted directly into the SQL query using an f-string, without any sanitization or parameterization. This is the root cause of the SQL injection vulnerability.

## Conclusion

- Directly interpolating user input into SQL queries is dangerous and insecure.  
- Using prepared statements or an ORM (Object Relational Mapper) like SQLAlchemy with parameterized queries is highly recommended.  
- This report highlights how attackers can gain unauthorized access through SQL Injection if inputs are not handled securely.