Data Sanitization-Protection Used

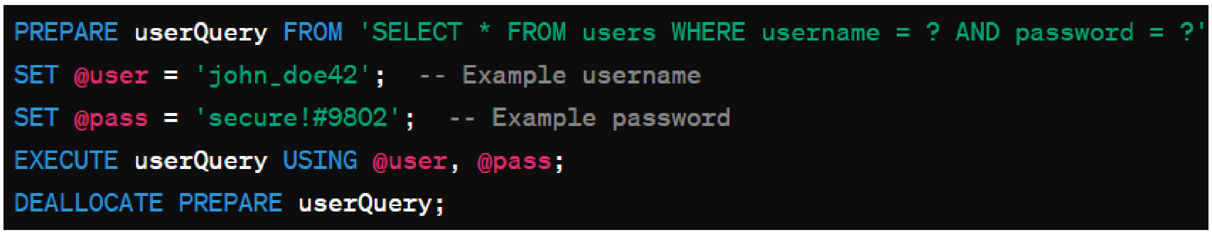
Rithvik Rajesh Krishnan (U14461770)

In Project 2:

Prepared Statements

Functionality: Employing prepared statements not only enhances performance but also greatly bolsters security by isolating SQL code from data. This segregation effectively thwarts attackers from injecting malevolent SQL code through user inputs.

Code Snippet:

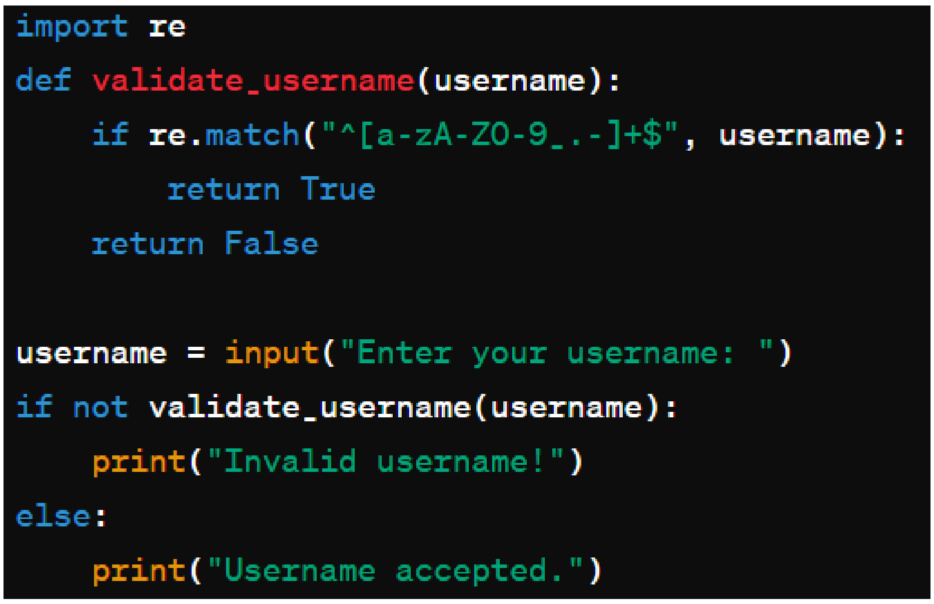


```sql
PREPARE userQuery FROM 'SELECT * FROM users WHERE username = ? AND password = ?'
SET @user = 'john_doe42';  -- Example username
SET @pass = 'secure!#9802';  -- Example password
EXECUTE userQuery USING @user, @pass;
DEALLOCATE PREPARE userQuery;
```

Input Validation

Functionality: The user input is checked against predefined criteria, such as type, format, and length. Any input that doesn't meet these criteria is rejected, a critical step in thwarting injection attacks and other potential data corruptions.

Code Snippet:

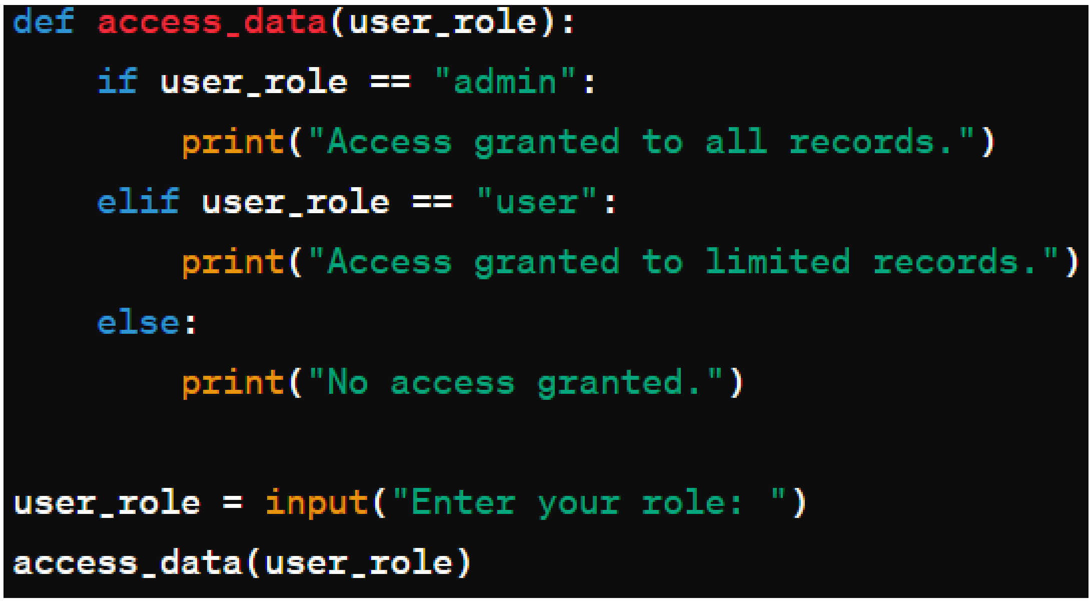


```python
import re
def validate_username(username):
    if re.match("^[a-zA-Z0-9_.-]+$", username):
        return True
    return False

username = input("Enter your username: ")
if not validate_username(username):
    print("Invalid username!")
else:
    print("Username accepted.")
```

Role-based Access Control

Functionality: It establishes protocols for managing database access permissions according to user roles, guaranteeing that users can solely access data pertinent to their assigned role.

Code Snippet:



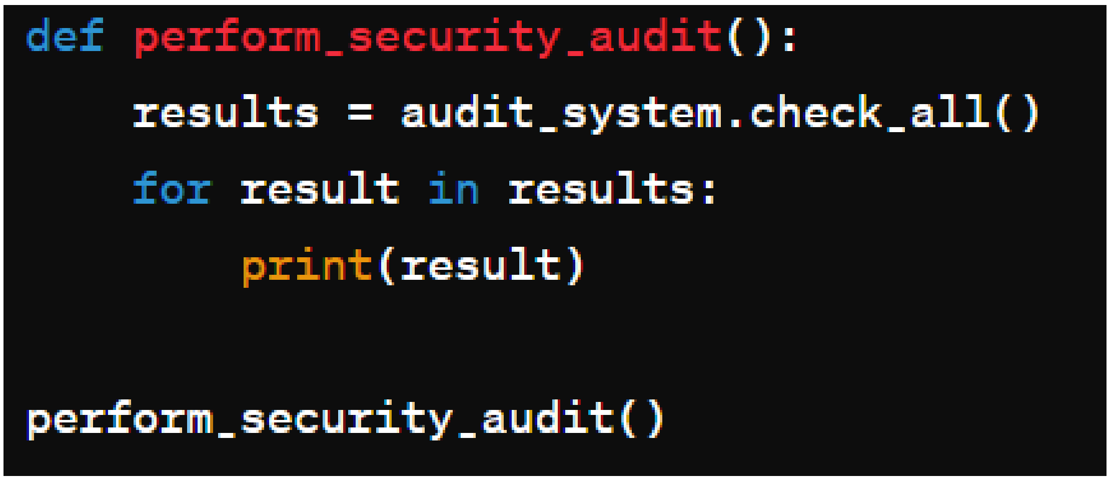
```python
def access_data(user_role):
    if user_role == "admin":
        print("Access granted to all records.")
    elif user_role == "user":
        print("Access granted to limited records.")
    else:
        print("No access granted.")

user_role = input("Enter your role: ")
access_data(user_role)
```

In Peer to Peer (P2P) Communication:

Regular Security Audits

Functionality: Consistently evaluates and upgrades security protocols to stay abreast of emerging threats, thereby ensuring ongoing safeguarding of data.

Code Snippets:



```python
def perform_security_audit():
    results = audit_system.check_all()
    for result in results:
        print(result)

perform_security_audit()
```