EX.NO.: 03
DATE: 02.07.2025

SECURE USER AUTHENTICATION MODULE USING PYTEST

AIM

To design and test a secure user authentication system that validates login credentials, tracks failed login attempts, blocks users after multiple failures, and generates secure session tokens using Python and Pytest.

ALGORITHM

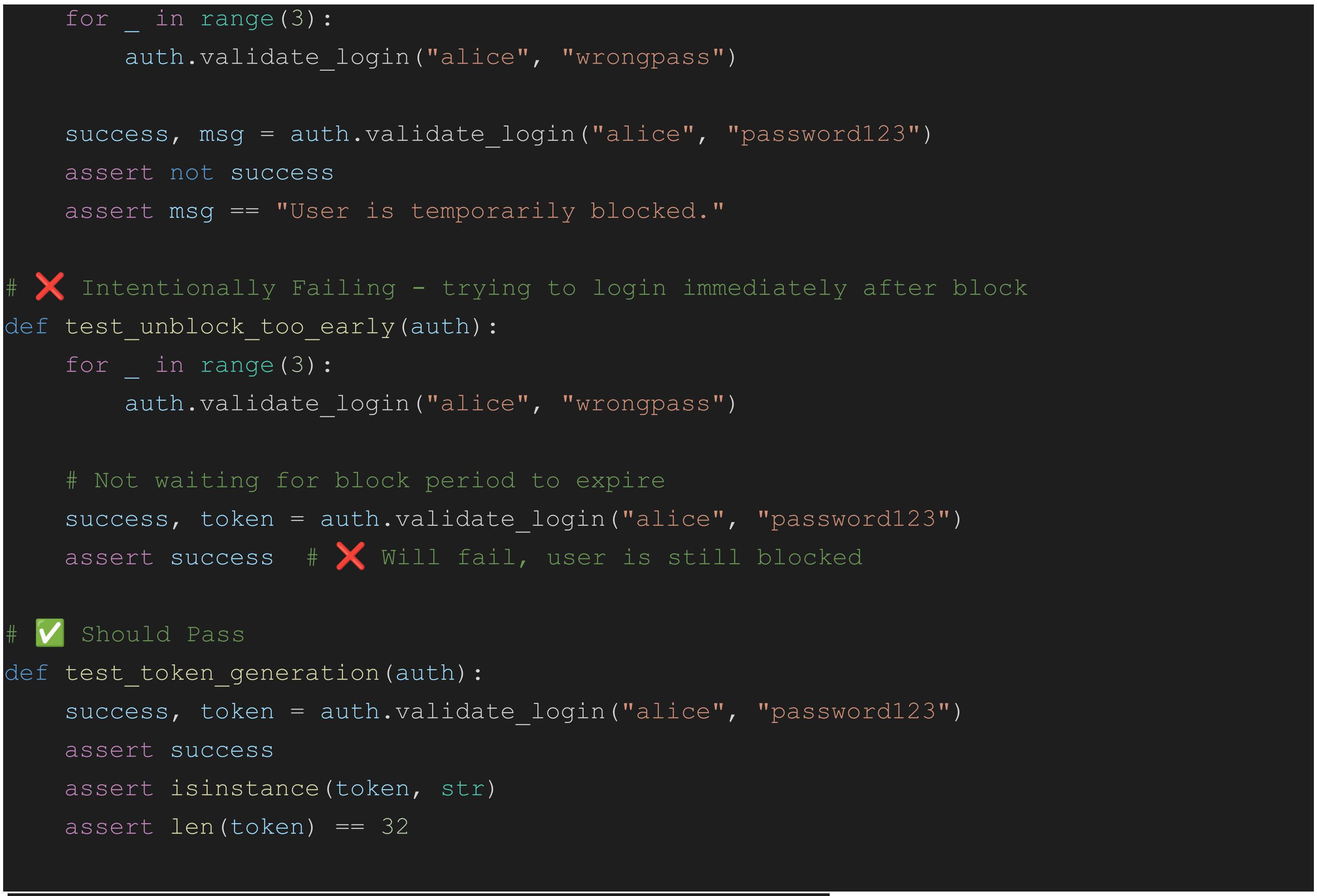
1. Initialize user storage, failed attempt counter, blocked users, and session store.
2. Register users by storing securely hashed passwords (using SHA-256).
3. On login attempt:
   - If user is blocked, deny access.
   - If password matches, generate and return a secure session token.
   - If incorrect, increment failed attempt count.
4. Block users for 60 seconds after 3 failed attempts.
5. Generate session tokens using a cryptographically secure random generator.
6. Validate session tokens by checking token store.

CODE AND OUTPUT

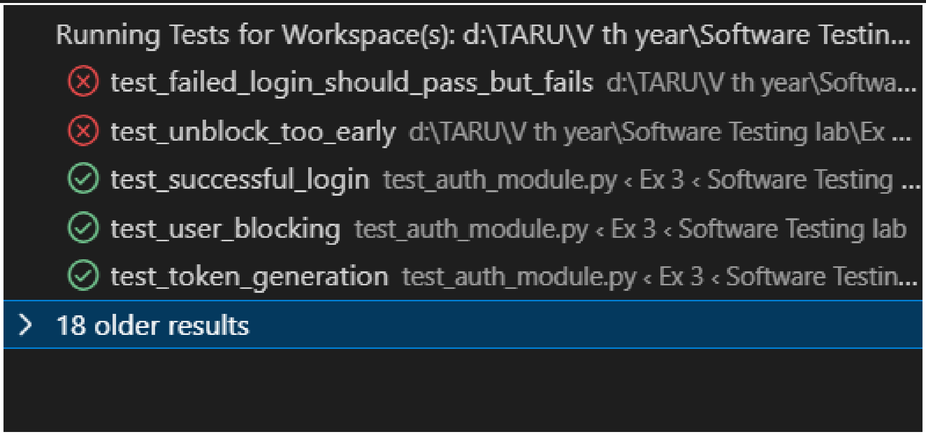
```
import hashlib
import secrets
import time

class UserAuth:
    def __init__(self):
        self.users = {}  # Format: {username: hashed_password}
        self.failed_attempts = {}  # Format: {username: [count, last_failed_time]}
        self.blocked_users = {}  # Format: {username: unblock_time}
        self.sessions = {}  # Format: {session_token: username}

    def _hash_password(self, password):
        return hashlib.sha256(password.encode()).hexdigest()

    def register_user(self, username, password):
        self.users[username] = self._hash_password(password)

    def is_blocked(self, username):
        unblock_time = self.blocked_users.get(username)
        if unblock_time and time.time() < unblock_time:
            return True
        elif unblock_time:
            del self.blocked_users[username]  # Unblock if time expired
        return False

    def validate_login(self, username, password):
        if self.is_blocked(username):
            return False, "User is temporarily blocked."

        hashed_input = self._hash_password(password)
```

```python
        if self.users.get(username) == hashed_input:
            self.failed_attempts.pop(username, None)  # Reset failures
            token = self._generate_session_token(username)
            return True, token
        else:
            self._handle_failed_attempt(username)
            return False, "Invalid credentials."

    def _handle_failed_attempt(self, username):
        count, last_time = self.failed_attempts.get(username, (0, 0))
        count += 1
        self.failed_attempts[username] = (count, time.time())

        if count >= 3:
            self.blocked_users[username] = time.time() + 60  # Block for 1 min
            self.failed_attempts.pop(username)

    def _generate_session_token(self, username):
        token = secrets.token_hex(16)
        self.sessions[token] = username
        return token

    def is_token_valid(self, token):
        return token in self.sessions
```

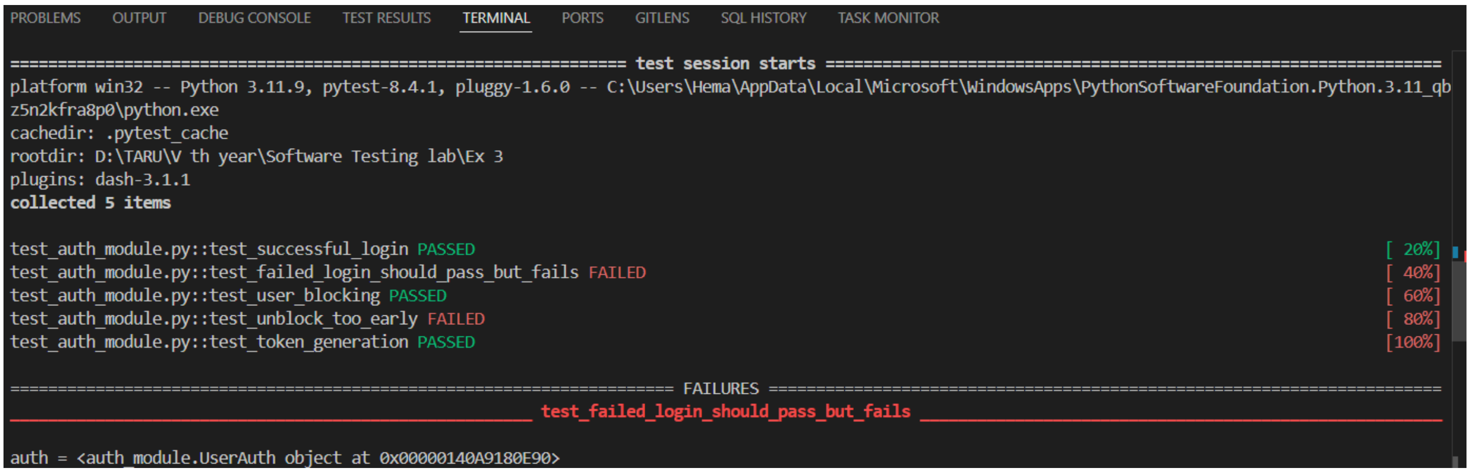
```python
import time
import pytest
from auth_module import UserAuth

@pytest.fixture
def auth():
    auth = UserAuth()
    auth.register_user("alice", "password123")
    return auth

# ✅ Should Pass
def test_successful_login(auth):
    success, token = auth.validate_login("alice", "password123")
    assert success
    assert auth.is_token_valid(token)

# ❌ Intentionally Failing - expects wrong password to be valid
def test_failed_login_should_pass_but_fails(auth):
    success, message = auth.validate_login("alice", "wrongpass")
    assert success  # ❌ This will fail because login should fail

# ✅ Should Pass
def test_user_blocking(auth):
```



```python
    for _ in range(3):
        auth.validate_login("alice", "wrongpass")

    success, msg = auth.validate_login("alice", "password123")
    assert not success
    assert msg == "User is temporarily blocked."

# ❌ Intentionally Failing - trying to login immediately after block
def test_unblock_too_early(auth):
    for _ in range(3):
        auth.validate_login("alice", "wrongpass")

    # Not waiting for block period to expire
    success, token = auth.validate_login("alice", "password123")
    assert success  # ❌ Will fail, user is still blocked

# ✅ Should Pass
def test_token_generation(auth):
    success, token = auth.validate_login("alice", "password123")
    assert success
    assert isinstance(token, str)
    assert len(token) == 32
```



```
Running Tests for Workspace(s): d:\TARU\V th year\Software Testin...
  ⓧ test_failed_login_should_pass_but_fails d:\TARU\V th year\Softwa...
  ⓧ test_unblock_too_early d:\TARU\V th year\Software Testing lab\Ex ...
  ✓ test_successful_login test_auth_module.py ‹ Ex 3 ‹ Software Testing ...
  ✓ test_user_blocking test_auth_module.py ‹ Ex 3 ‹ Software Testing lab
  ✓ test_token_generation test_auth_module.py ‹ Ex 3 ‹ Software Testin...
> 18 older results
```



```
============================================================ test session starts ============================================================
platform win32 -- Python 3.11.9, pytest-8.4.1, pluggy-1.6.0 -- C:\Users\Hema\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qb
z5n2kfra8p0\python.exe
cachedir: .pytest_cache
rootdir: D:\TARU\V th year\Software Testing lab\Ex 3
plugins: dash-3.1.1
collected 5 items

test_auth_module.py::test_successful_login PASSED                                                                                      [ 20%]
test_auth_module.py::test_failed_login_should_pass_but_fails FAILED                                                                    [ 40%]
test_auth_module.py::test_user_blocking PASSED                                                                                         [ 60%]
test_auth_module.py::test_unblock_too_early FAILED                                                                                     [ 80%]
test_auth_module.py::test_token_generation PASSED                                                                                      [100%]

================================================================= FAILURES =================================================================
_________________________________________________ test_failed_login_should_pass_but_fails _________________________________________________

auth = <auth_module.UserAuth object at 0x00000140A9180E90>
```

INFERENCE

The authentication module works as intended by securely handling login, blocking after multiple failed attempts, and generating valid session tokens. The test results confirm both correct functionality and failure handling through Pytest.