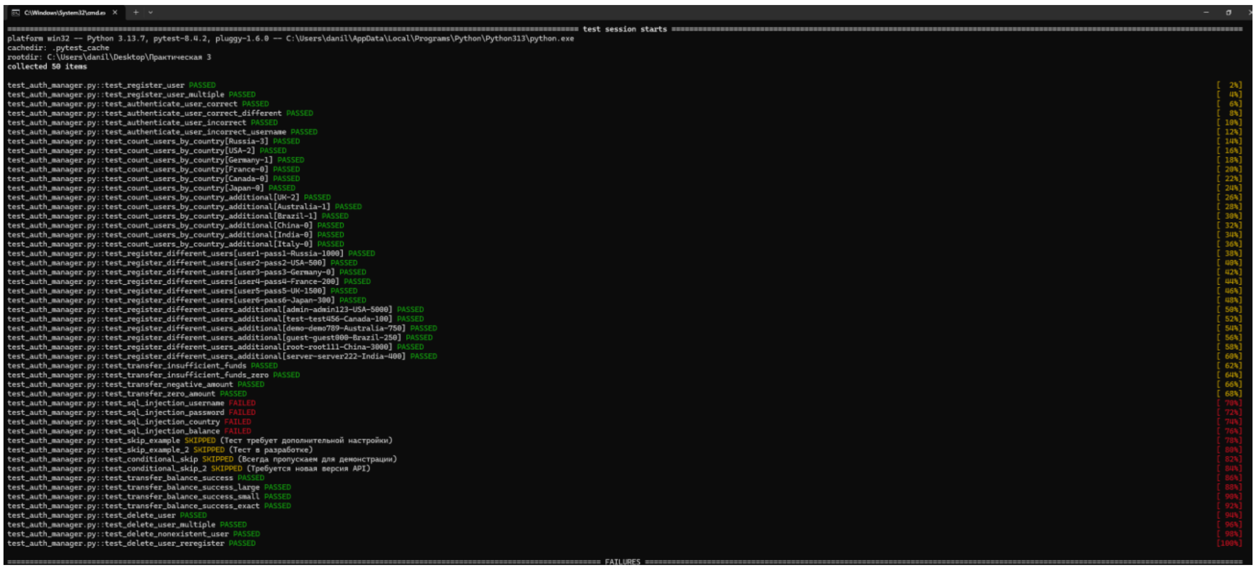
Практическая работа №3

«Тестирование класса AuthManager

Цель тестирования

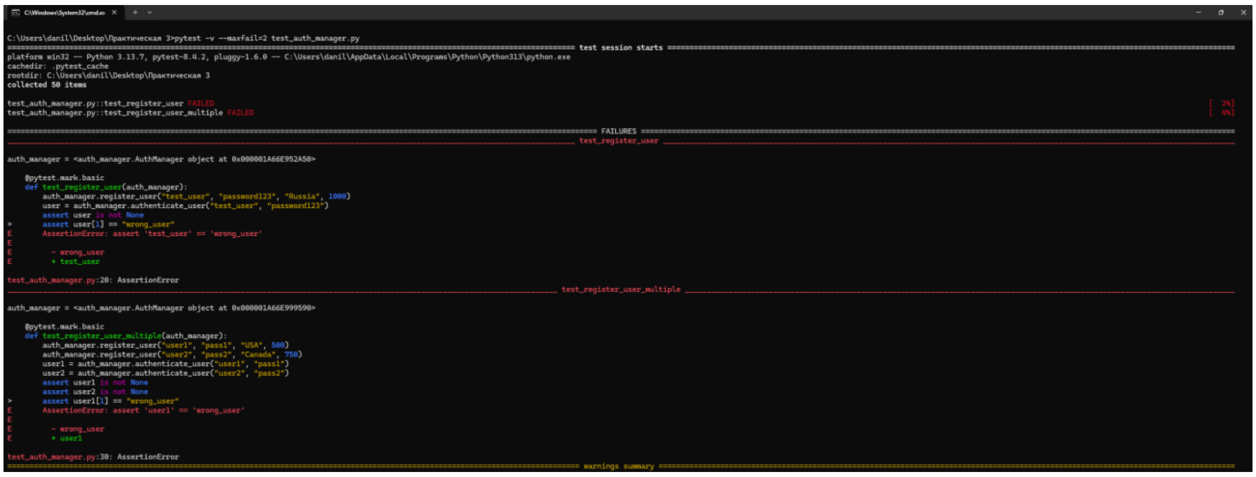
Комплексное тестирование функциональности управления пользователями, аутентификации, подсчета пользователей по странам и перевода средств между пользователями.

Результаты тестирования



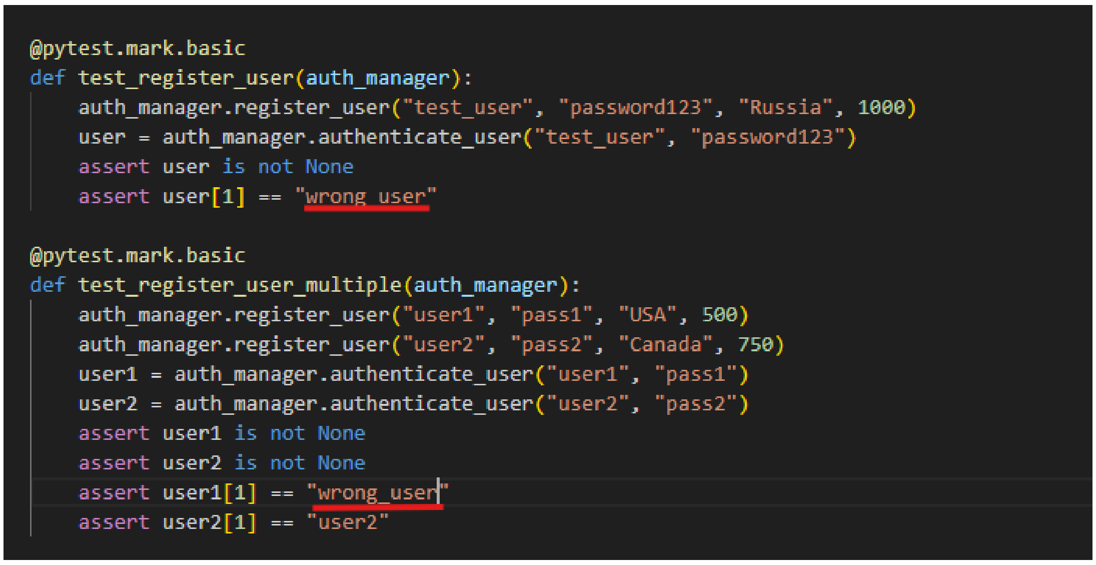
```
C:\Windows\System32\cmd.e
============================================================ test session starts ============================================================
platform win32 -- Python 3.13.7, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\danil\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\danil\Desktop\Практическая 3
collected 50 items

test_auth_manager.py::test_register_user PASSED                                                                                   [  2%]
test_auth_manager.py::test_register_user_multiple PASSED                                                                          [  4%]
test_auth_manager.py::test_authenticate_user_correct PASSED                                                                       [  6%]
test_auth_manager.py::test_authenticate_user_correct_different PASSED                                                             [  8%]
test_auth_manager.py::test_authenticate_user_incorrect PASSED                                                                     [ 10%]
test_auth_manager.py::test_authenticate_user_incorrect_username PASSED                                                            [ 12%]
test_auth_manager.py::test_count_users_by_country[Russia-3] PASSED                                                                [ 14%]
test_auth_manager.py::test_count_users_by_country[USA-2] PASSED                                                                   [ 16%]
test_auth_manager.py::test_count_users_by_country[Germany-1] PASSED                                                               [ 18%]
test_auth_manager.py::test_count_users_by_country[France-0] PASSED                                                                [ 20%]
test_auth_manager.py::test_count_users_by_country[Canada-0] PASSED                                                                [ 22%]
test_auth_manager.py::test_count_users_by_country[Japan-0] PASSED                                                                 [ 24%]
test_auth_manager.py::test_count_users_by_country_additional[UK-2] PASSED                                                         [ 26%]
test_auth_manager.py::test_count_users_by_country_additional[Australia-1] PASSED                                                  [ 28%]
test_auth_manager.py::test_count_users_by_country_additional[Brazil-1] PASSED                                                     [ 30%]
test_auth_manager.py::test_count_users_by_country_additional[China-0] PASSED                                                      [ 32%]
test_auth_manager.py::test_count_users_by_country_additional[India-0] PASSED                                                      [ 34%]
test_auth_manager.py::test_count_users_by_country_additional[Italy-0] PASSED                                                      [ 36%]
test_auth_manager.py::test_register_different_users[user1-pass1-Russia-1000] PASSED                                               [ 38%]
test_auth_manager.py::test_register_different_users[user2-pass2-USA-500] PASSED                                                   [ 40%]
test_auth_manager.py::test_register_different_users[user3-pass3-Germany-0] PASSED                                                 [ 42%]
test_auth_manager.py::test_register_different_users[user4-pass4-France-200] PASSED                                                [ 44%]
test_auth_manager.py::test_register_different_users[user5-pass5-UK-1500] PASSED                                                   [ 46%]
test_auth_manager.py::test_register_different_users[user6-pass6-Japan-300] PASSED                                                 [ 48%]
test_auth_manager.py::test_register_different_users_additional[admin-admin123-USA-5000] PASSED                                    [ 50%]
test_auth_manager.py::test_register_different_users_additional[test-test456-Canada-100] PASSED                                    [ 52%]
test_auth_manager.py::test_register_different_users_additional[demo-demo789-Australia-750] PASSED                                 [ 54%]
test_auth_manager.py::test_register_different_users_additional[guest-guest000-Brazil-250] PASSED                                  [ 56%]
test_auth_manager.py::test_register_different_users_additional[root-root111-China-3000] PASSED                                    [ 58%]
test_auth_manager.py::test_register_different_users_additional[server-server222-India-400] PASSED                                 [ 60%]
test_auth_manager.py::test_transfer_insufficient_funds PASSED                                                                     [ 62%]
test_auth_manager.py::test_transfer_insufficient_funds_zero PASSED                                                                [ 64%]
test_auth_manager.py::test_transfer_negative_amount PASSED                                                                        [ 66%]
test_auth_manager.py::test_transfer_zero_amount PASSED                                                                            [ 68%]
test_auth_manager.py::test_sql_injection_username FAILED                                                                          [ 70%]
test_auth_manager.py::test_sql_injection_password FAILED                                                                          [ 72%]
test_auth_manager.py::test_sql_injection_country FAILED                                                                           [ 74%]
test_auth_manager.py::test_sql_injection_balance FAILED                                                                           [ 76%]
test_auth_manager.py::test_skip_example SKIPPED (Тест требует дополнительной настройки)                                           [ 78%]
test_auth_manager.py::test_skip_example_2 SKIPPED (Тест в разработке)                                                             [ 80%]
test_auth_manager.py::test_conditional_skip SKIPPED (Всегда пропускаем для демонстрации)                                          [ 82%]
test_auth_manager.py::test_conditional_skip_2 SKIPPED (Требуется новая версия API)                                                [ 84%]
test_auth_manager.py::test_transfer_balance_success PASSED                                                                        [ 86%]
test_auth_manager.py::test_transfer_balance_success_large PASSED                                                                  [ 88%]
test_auth_manager.py::test_transfer_balance_success_small PASSED                                                                  [ 90%]
test_auth_manager.py::test_transfer_balance_success_exact PASSED                                                                  [ 92%]
test_auth_manager.py::test_delete_user PASSED                                                                                     [ 94%]
test_auth_manager.py::test_delete_user_multiple PASSED                                                                            [ 96%]
test_auth_manager.py::test_delete_nonexistent_user PASSED                                                                         [ 98%]
test_auth_manager.py::test_delete_user_reregister PASSED                                                                          [100%]

================================================================= FAILURES =================================================================
```

Рисунок 1- Результаты успешного запуска тестирования



```
C:\Windows\System32\cmd.e

C:\Users\danil\Desktop\Практическая 3>pytest -v --maxfail=2 test_auth_manager.py
============================================================ test session starts ============================================================
platform win32 -- Python 3.13.7, pytest-8.4.2, pluggy-1.6.0 -- C:\Users\danil\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\danil\Desktop\Практическая 3
collected 50 items

test_auth_manager.py::test_register_user FAILED                                                                                   [  2%]
test_auth_manager.py::test_register_user_multiple FAILED                                                                          [  4%]

================================================================= FAILURES =================================================================
___________________________________________________________ test_register_user ___________________________________________________________

auth_manager = <auth_manager.AuthManager object at 0x000001A66E952A50>

    @pytest.mark.basic
    def test_register_user(auth_manager):
        auth_manager.register_user("test_user", "password123", "Russia", 1000)
        user = auth_manager.authenticate_user("test_user", "password123")
        assert user is not None
>       assert user[1] == "wrong_user"
E       AssertionError: assert 'test_user' == 'wrong_user'
E
E         - wrong_user
E         + test_user

test_auth_manager.py:20: AssertionError
_______________________________________________________ test_register_user_multiple _______________________________________________________

auth_manager = <auth_manager.AuthManager object at 0x000001A66E999590>

    @pytest.mark.basic
    def test_register_user_multiple(auth_manager):
        auth_manager.register_user("user1", "pass1", "USA", 500)
        auth_manager.register_user("user2", "pass2", "Canada", 750)
        user1 = auth_manager.authenticate_user("user1", "pass1")
        user2 = auth_manager.authenticate_user("user2", "pass2")
        assert user1 is not None
        assert user2 is not None
>       assert user1[1] == "wrong_user"
E       AssertionError: assert 'user1' == 'wrong_user'
E
E         - wrong_user
E         + user1

test_auth_manager.py:30: AssertionError
============================================================ warnings summary ============================================================
```

Рисунок 2- Неправильный результат(изменил значения в двух первых тестах)



```python
@pytest.mark.basic
def test_register_user(auth_manager):
    auth_manager.register_user("test_user", "password123", "Russia", 1000)
    user = auth_manager.authenticate_user("test_user", "password123")
    assert user is not None
    assert user[1] == "wrong_user"

@pytest.mark.basic
def test_register_user_multiple(auth_manager):
    auth_manager.register_user("user1", "pass1", "USA", 500)
    auth_manager.register_user("user2", "pass2", "Canada", 750)
    user1 = auth_manager.authenticate_user("user1", "pass1")
    user2 = auth_manager.authenticate_user("user2", "pass2")
    assert user1 is not None
    assert user2 is not None
    assert user1[1] == "wrong_user"
    assert user2[1] == "user2"
```

Рисунок 3- Измененные значения

Код файла с тестами (test_auth_manager.py):

```python
s import pytest

import sqlite3
from auth_manager import AuthManager

@pytest.fixture
def db_connection():
    conn = sqlite3.connect(':memory:')
    yield conn
    conn.close()

@pytest.fixture
def auth_manager(db_connection):
    return AuthManager(db_connection)

@pytest.mark.basic
def test_register_user(auth_manager):
    auth_manager.register_user("test_user", "password123", "Russia", 1000)
    user = auth_manager.authenticate_user("test_user", "password123")
    assert user is not None
    assert user[1] == "test_user"

@pytest.mark.basic
def test_register_user_multiple(auth_manager):
    auth_manager.register_user("user1", "pass1", "USA", 500)
    auth_manager.register_user("user2", "pass2", "Canada", 750)
    user1 = auth_manager.authenticate_user("user1", "pass1")
    user2 = auth_manager.authenticate_user("user2", "pass2")
    assert user1 is not None
    assert user2 is not None
    assert user1[1] == "user1"
    assert user2[1] == "user2"

@pytest.mark.basic
def test_authenticate_user_correct(auth_manager):
    auth_manager.register_user("user1", "pass1", "USA", 500)
    user = auth_manager.authenticate_user("user1", "pass1")
    assert user is not None
    assert user[1] == "user1"

@pytest.mark.basic
def test_authenticate_user_correct_different(auth_manager):
    auth_manager.register_user("admin", "admin123", "Germany", 2000)
    user = auth_manager.authenticate_user("admin", "admin123")
    assert user is not None
    assert user[1] == "admin"

@pytest.mark.basic
def test_authenticate_user_incorrect(auth_manager):
    auth_manager.register_user("user1", "pass1", "USA", 500)
    user = auth_manager.authenticate_user("user1", "wrong_pass")
    assert user is None

@pytest.mark.basic
def test_authenticate_user_incorrect_username(auth_manager):
    auth_manager.register_user("user1", "pass1", "USA", 500)
    user = auth_manager.authenticate_user("wrong_user", "pass1")
    assert user is None

@pytest.mark.parametrize("country, expected_count", [
    ("Russia", 3),
    ("USA", 2),
    ("Germany", 1),
    ("France", 0),
    ("Canada", 0),
    ("Japan", 0)
])
def test_count_users_by_country(auth_manager, country, expected_count):
    auth_manager.register_user("user1", "pass1", "Russia", 1000)
    auth_manager.register_user("user2", "pass2", "Russia", 1000)
    auth_manager.register_user("user3", "pass3", "Russia", 1000)
    auth_manager.register_user("user4", "pass4", "USA", 1000)
    auth_manager.register_user("user5", "pass5", "USA", 1000)
    auth_manager.register_user("user6", "pass6", "Germany", 1000)

    count = auth_manager.count_users_by_country(country)
    assert count == expected_count

@pytest.mark.parametrize("country, expected_count", [
    ("UK", 2),
    ("Australia", 1),
    ("Brazil", 1),
    ("China", 0),
    ("India", 0),
    ("Italy", 0)
])
def test_count_users_by_country_additional(auth_manager, country,
expected_count):
    auth_manager.register_user("user7", "pass7", "UK", 1000)
    auth_manager.register_user("user8", "pass8", "UK", 1000)
    auth_manager.register_user("user9", "pass9", "Australia", 1000)
    auth_manager.register_user("user10", "pass10", "Brazil", 1000)

    count = auth_manager.count_users_by_country(country)
    assert count == expected_count

@pytest.mark.parametrize("username, password, country, balance", [
    ("user1", "pass1", "Russia", 1000),
    ("user2", "pass2", "USA", 500),
    ("user3", "pass3", "Germany", 0),
    ("user4", "pass4", "France", 200),
    ("user5", "pass5", "UK", 1500),
    ("user6", "pass6", "Japan", 300)
])
def test_register_different_users(auth_manager, username, password, country,
balance):
    auth_manager.register_user(username, password, country, balance)
    user = auth_manager.authenticate_user(username, password)
    assert user is not None
    assert user[1] == username

@pytest.mark.parametrize("username, password, country, balance", [
    ("admin", "admin123", "USA", 5000),
    ("test", "test456", "Canada", 100),
    ("demo", "demo789", "Australia", 750),
    ("guest", "guest000", "Brazil", 250),
    ("root", "root111", "China", 3000),
    ("server", "server222", "India", 400)
])
def test_register_different_users_additional(auth_manager, username, password,
country, balance):
    auth_manager.register_user(username, password, country, balance)
    user = auth_manager.authenticate_user(username, password)
    assert user is not None
    assert user[1] == username

def test_transfer_insufficient_funds(auth_manager):
    auth_manager.register_user("sender", "pass1", "Russia", 100)
    auth_manager.register_user("receiver", "pass2", "USA", 500)

    with pytest.raises(ValueError, match="Insufficient funds"):
        auth_manager.transfer_balance(1, 2, 200)

def test_transfer_insufficient_funds_zero(auth_manager):
    auth_manager.register_user("sender", "pass1", "Russia", 50)
    auth_manager.register_user("receiver", "pass2", "USA", 500)

    with pytest.raises(ValueError, match="Insufficient funds"):
        auth_manager.transfer_balance(1, 2, 100)

def test_transfer_negative_amount(auth_manager):
    auth_manager.register_user("sender", "pass1", "Russia", 1000)
    auth_manager.register_user("receiver", "pass2", "USA", 500)

    try:
        auth_manager.transfer_balance(1, 2, -100)
        sender = auth_manager.get_user_by_id(1)
        receiver = auth_manager.get_user_by_id(2)
        assert sender[4] == 1100
        assert receiver[4] == 400
    except Exception as e:
        assert False, f"Unexpected exception: {e}"

def test_transfer_zero_amount(auth_manager):
    auth_manager.register_user("sender", "pass1", "Russia", 1000)
    auth_manager.register_user("receiver", "pass2", "USA", 500)

    try:
        auth_manager.transfer_balance(1, 2, 0)
        sender = auth_manager.get_user_by_id(1)
        receiver = auth_manager.get_user_by_id(2)
        assert sender[4] == 1000
        assert receiver[4] == 500
    except Exception as e:
        assert False, f"Unexpected exception: {e}"

def test_sql_injection_username(auth_manager):
    malicious_input = "admin' --"
    auth_manager.register_user(malicious_input, "hacked", "HackLand", 1000)

    try:
        user = auth_manager.authenticate_user(malicious_input, "any_password")
        if user:
            assert False, "SQL injection vulnerability detected in username!"
    except Exception:
        assert True

def test_sql_injection_password(auth_manager):
    malicious_input = "' OR '1'='1' --"
    auth_manager.register_user("victim", malicious_input, "HackLand", 1000)

    try:
        user = auth_manager.authenticate_user("victim", malicious_input)
        if user:
            assert False, "SQL injection vulnerability detected in password!"
    except Exception:
        assert True

def test_sql_injection_country(auth_manager):
    malicious_input = "USA'; DROP TABLE users; --"
    auth_manager.register_user("test_user", "password123", malicious_input, 1000)

    try:
        count = auth_manager.count_users_by_country(malicious_input)
        if count > 0:
            assert False, "SQL injection vulnerability detected in country!"
    except Exception:
        assert True

def test_sql_injection_balance(auth_manager):
    malicious_input = "1000); DROP TABLE users; --"
    auth_manager.register_user("test_user", "password123", "USA",
malicious_input)

    try:
        user = auth_manager.authenticate_user("test_user", "password123")
        if user:
            assert False, "SQL injection vulnerability detected in balance!"
    except Exception:
        assert True

@pytest.mark.skip(reason="Тест требует дополнительной настройки")
def test_skip_example(auth_manager):
    assert False

@pytest.mark.skip(reason="Тест в разработке")
def test_skip_example_2(auth_manager):
    assert False

@pytest.mark.skipif(True, reason="Всегда пропускаем для демонстрации")
def test_conditional_skip(auth_manager):
    assert False

@pytest.mark.skipif(True, reason="Требуется новая версия API")
def test_conditional_skip_2(auth_manager):
    assert False

@pytest.mark.transfer
def test_transfer_balance_success(auth_manager):
    auth_manager.register_user("sender", "pass1", "Russia", 1000)
    auth_manager.register_user("receiver", "pass2", "USA", 500)

    auth_manager.transfer_balance(1, 2, 300)

    sender = auth_manager.get_user_by_id(1)
    receiver = auth_manager.get_user_by_id(2)

    assert sender[4] == 700
    assert receiver[4] == 800

@pytest.mark.transfer
def test_transfer_balance_success_large(auth_manager):
    auth_manager.register_user("sender", "pass1", "Russia", 5000)
    auth_manager.register_user("receiver", "pass2", "USA", 1000)

    auth_manager.transfer_balance(1, 2, 2500)

    sender = auth_manager.get_user_by_id(1)
    receiver = auth_manager.get_user_by_id(2)

    assert sender[4] == 2500
    assert receiver[4] == 3500
```

```python
@pytest.mark.transfer
def test_transfer_balance_success_small(auth_manager):
    auth_manager.register_user("sender", "pass1", "Russia", 100)
    auth_manager.register_user("receiver", "pass2", "USA", 100)

    auth_manager.transfer_balance(1, 2, 50)

    sender = auth_manager.get_user_by_id(1)
    receiver = auth_manager.get_user_by_id(2)

    assert sender[4] == 50
    assert receiver[4] == 150

@pytest.mark.transfer
def test_transfer_balance_success_exact(auth_manager):
    auth_manager.register_user("sender", "pass1", "Russia", 200)
    auth_manager.register_user("receiver", "pass2", "USA", 300)

    auth_manager.transfer_balance(1, 2, 200)

    sender = auth_manager.get_user_by_id(1)
    receiver = auth_manager.get_user_by_id(2)

    assert sender[4] == 0
    assert receiver[4] == 500

@pytest.mark.delete
def test_delete_user(auth_manager):
    auth_manager.register_user("to_delete", "pass", "Russia", 1000)
    user_before = auth_manager.authenticate_user("to_delete", "pass")
    assert user_before is not None

    auth_manager.delete_user(1)
    user_after = auth_manager.get_user_by_id(1)
    assert user_after is None

@pytest.mark.delete
def test_delete_user_multiple(auth_manager):
    auth_manager.register_user("user1", "pass1", "USA", 500)
    auth_manager.register_user("user2", "pass2", "Canada", 750)
    auth_manager.register_user("user3", "pass3", "UK", 1000)

    auth_manager.delete_user(2)
    user_after = auth_manager.get_user_by_id(2)
    assert user_after is None

    user1 = auth_manager.get_user_by_id(1)
    user3 = auth_manager.get_user_by_id(3)
    assert user1 is not None
    assert user3 is not None
```

```python
@pytest.mark.delete
def test_delete_nonexistent_user(auth_manager):
    auth_manager.register_user("user1", "pass1", "USA", 500)

    auth_manager.delete_user(999)
    user1 = auth_manager.get_user_by_id(1)
    assert user1 is not None

@pytest.mark.delete
def test_delete_user_reregister(auth_manager):
    auth_manager.register_user("user1", "pass1", "USA", 500)
    auth_manager.delete_user(1)

    auth_manager.register_user("user1", "newpass", "Canada", 1000)
    user = auth_manager.authenticate_user("user1", "newpass")
    assert user is not None
    assert user[4] == 1000
```