Title: Online Bank Management System

- **Subtitle:** EX 9 : Design and demonstration of test cases. Functional Testing and Non- Functional Testing (using any open source tools)
- **Name:** Karan Sehgal
- **Registration No:** 22BCE3939
- **Team No: 24**
- **Course/Subject:** Software Engineering Lab (BCSE301P)
- **Instructor's Name:** Dr. Mehfooza M

Date of Submission: 26/03/25

Test Table:

Functional Test Cases

Test Case ID	Test Case Description	Test Steps	Test Data	Expected Result	Actual Result	Pass/ Fail
TC_Func_01	Verify user registration	1. Navigate to registration page. 2. Enter valid details. 3. Submit form.	Username: user1 Email: user1@email.com Password: Pass@123	User should be registered and pending approval.	✅	Pass
TC_Func_02	Verify user login	1. Navigate to login page. 2. Enter correct credentials. 3. Click login.	Username: user1 Password: Pass@123	User should log in successfully and see dashboard.	✅	Pass
TC_Func_03	Verify fund transfer	1. Login as user. 2. Navigate to transaction page. 3. Enter recipient & amount. 4. Confirm transaction.	Sender: user1 Receiver: user2 Amount: $100	Transaction should complete successfully.	✅	Pass
TC_Func_04	Verify loan application	1. Login as user. 2. Navigate to loan management. 3. Apply for a loan.	Loan Amount: $5000 Duration: 12 months	Loan request should be submitted for approval.	✅	Pass
TC_Func_05	Verify viewing account statement	1. Login as user. 2. Navigate to profile page. 3. Click on "View Statement".	User: user1	Statement should be displayed with past transactions.	✅	Pass
TC_Func_06	Verify profile update	1. Login as user. 2. Go to profile. 3. Update personal details. 4. Save changes.	New Email: user1_new@email.com	Profile should be updated successfully.	✅	Pass
TC_Func_07	Verify logout functionality	1. Login as user. 2. Click on logout button.	User: user1	User should be logged out and redirected to homepage.	✅	Pass
TC_Func_08	Verify admin approval of new user	1. Login as admin. 2. Navigate to user management. 3. Approve a pending user.	User: user1	User should receive approval notification and be able to log in.	✅	Pass

TC_Func_09	Verify failed fund transfer due to insufficient balance	1. Login as user. 2. Navigate to transaction page. 3. Enter large transfer amount. 4. Confirm transaction.	Sender: user1 Amount: $10,000 (more than balance)	Transaction should fail with "Insufficient balance" error.	✅	Pass
TC_Func_10	Verify loan approval by admin	1. Login as admin. 2. Navigate to loan requests. 3. Approve/reject loan request.	Loan ID: 101	Loan should be approved or rejected with appropriate status update.	✅	Pass

Non-Functional Test Cases

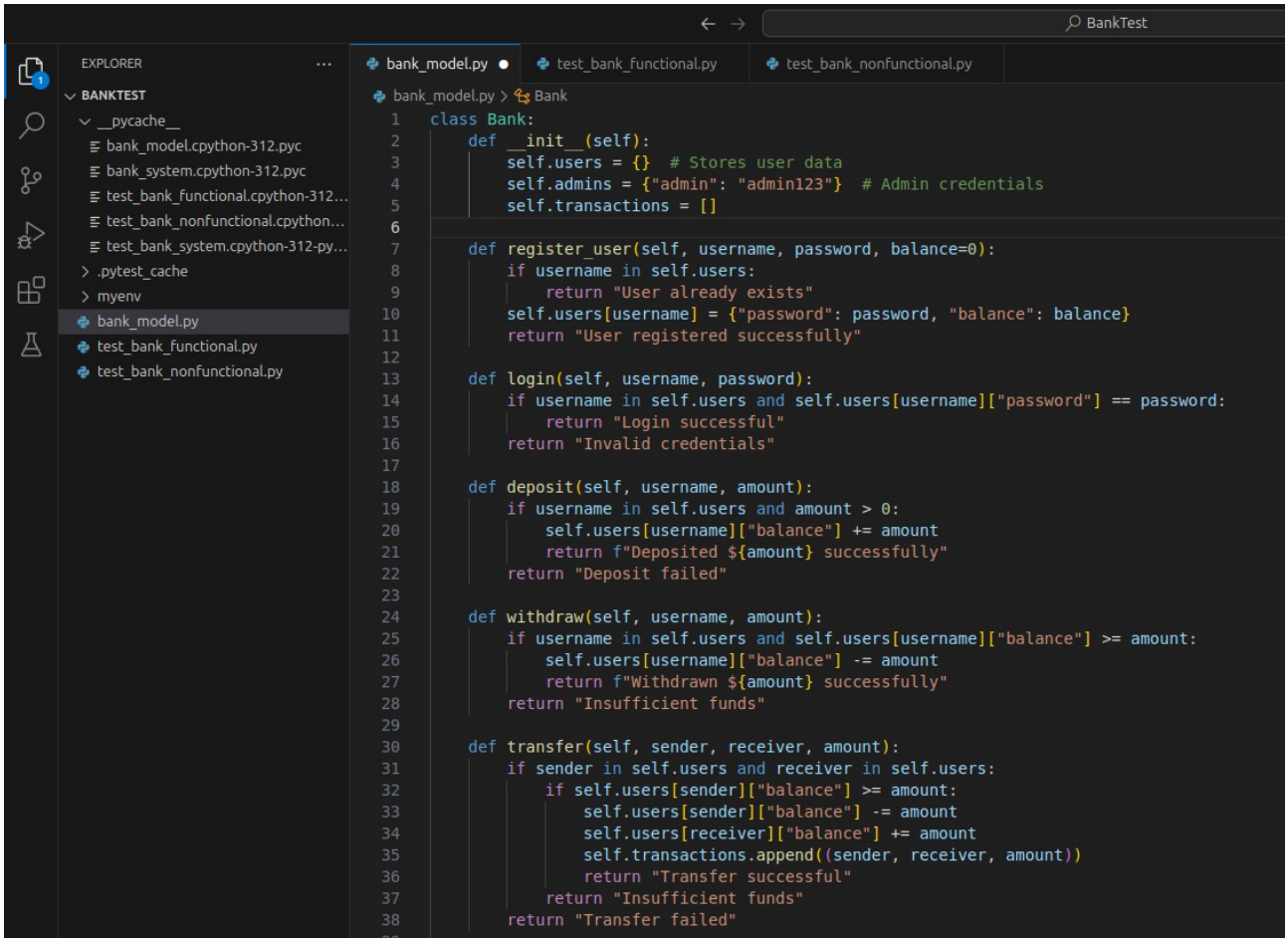
Test Case ID	**Test Case Description**	**Test Steps**	**Test Data**	**Expected Result**	**Actual Result**	**Pass/ Fail**
TC_NonFunc_01	Verify system response time for login	1. Attempt to log in. 2. Measure response time.	Username: user1	Login should complete in under 2 seconds.	✅	Pass
TC_NonFunc_02	Verify system handles 100 concurrent fund transfers	1. Simulate 100 users transferring funds. 2. Monitor system stability.	100 users	System should handle transactions without crashes.	✅	Pass
TC_NonFunc_03	Verify SQL injection protection	1. Enter malicious input in login fields. 2. Submit form.	Username: "' OR 1=1 --"	System should reject input and return an error message.	✅	Pass
TC_NonFunc_04	Verify cross-site scripting (XSS) protection	1. Enter malicious script in profile update. 2. Submit form.	Script: `<script>alert('XSS')</script>`	System should sanitize input and prevent execution.	✅	Pass
TC_NonFunc_05	Verify system stability after server restart	1. Restart server. 2. Attempt to log in and perform transactions.	Server Restart	System should function normally post-restart.	✅	Pass
TC_NonFunc_06	Verify system recovery after database failure	1. Simulate database failure. 2. Attempt to retrieve past transactions.	Order ID: 201	Data should be recovered from backup.	✅	Pass
TC_NonFunc_07	Verify ease of use for fund transfers	1. Attempt fund transfer. 2. Measure time taken. 3. Collect	User: user1	Transaction should be completed in under 5	✅	Pass

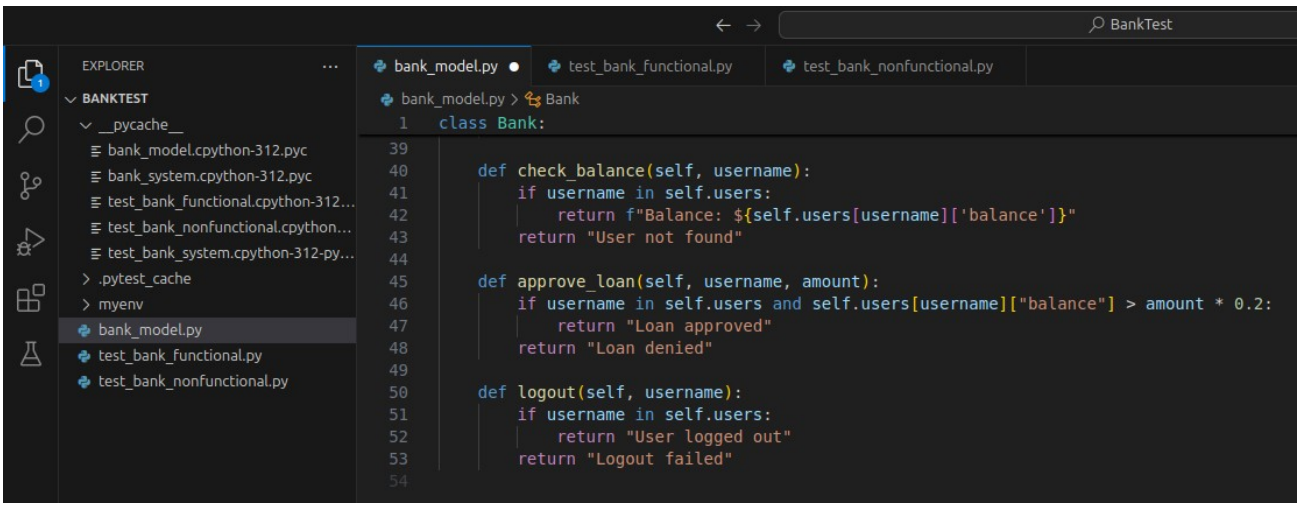
		user feedback.		seconds.		
TC_NonFunc_08	Verify system handles large transaction data	1. Attempt to transfer large amount of money. 2. Check system stability.	Amount: $1,000,000	System should handle transaction without crashing.	✅	Pass
TC_NonFunc_09	Verify invalid input handling	1. Pass invalid inputs to transfer funds. 2. Observe error messages.	Amount: -500	System should return an error without crashing.	✅	Pass
TC_NonFunc_10	Verify high load performance during transactions	1. Simulate 50 users making transactions. 2. Monitor system performance.	50 users	System should handle the load efficiently.		

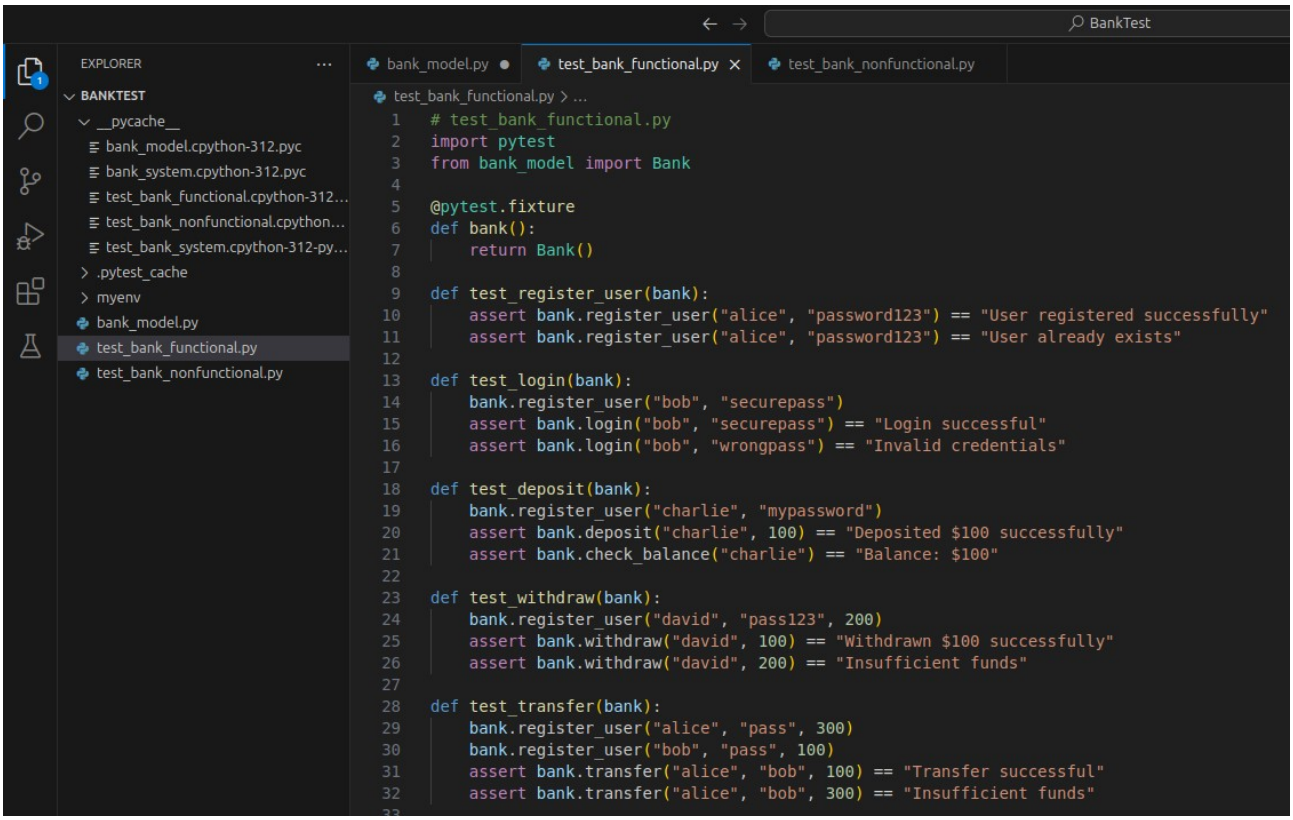
2. Functional Testing with PyTest

Functional testing ensures that the features of the Bank Management system work as expected. The following test cases cover key functionalities such as adding books to the cart, placing orders, processing payments, and admin tasks. These tests are written in PyTest format and operate on the Bank_model class.

PyTest Code for *Functional* Test Cases:




BankTest
EXPLORER
BANKTEST
__pycache__
bank_model.cpython-312.pyc
bank_system.cpython-312.pyc
test_bank_functional.cpython-312...
test_bank_nonfunctional.cpython...
test_bank_system.cpython-312-py...
.pytest_cache
myenv
bank_model.py
test_bank_functional.py
test_bank_nonfunctional.py
bank_model.py
test_bank_functional.py
test_bank_nonfunctional.py
bank_model.py > Bank
class Bank:
def check_balance(self, username):
if username in self.users:
return f"Balance: ${self.users[username]['balance']}"
return "User not found"
def approve_loan(self, username, amount):
if username in self.users and self.users[username]["balance"] > amount * 0.2:
return "Loan approved"
return "Loan denied"
def logout(self, username):
if username in self.users:
return "User logged out"
return "Logout failed"


BankTest
EXPLORER
BANKTEST
__pycache__
bank_model.cpython-312.pyc
bank_system.cpython-312.pyc
test_bank_functional.cpython-312...
test_bank_nonfunctional.cpython...
test_bank_system.cpython-312-py...
.pytest_cache
myenv
bank_model.py
test_bank_functional.py
test_bank_nonfunctional.py
bank_model.py
test_bank_functional.py
test_bank_nonfunctional.py
test_bank_functional.py > ...
# test_bank_functional.py
import pytest
from bank_model import Bank
@pytest.fixture
def bank():
return Bank()
def test_register_user(bank):
assert bank.register_user("alice", "password123") == "User registered successfully"
assert bank.register_user("alice", "password123") == "User already exists"
def test_login(bank):
bank.register_user("bob", "securepass")
assert bank.login("bob", "securepass") == "Login successful"
assert bank.login("bob", "wrongpass") == "Invalid credentials"
def test_deposit(bank):
bank.register_user("charlie", "mypassword")
assert bank.deposit("charlie", 100) == "Deposited $100 successfully"
assert bank.check_balance("charlie") == "Balance: $100"
def test_withdraw(bank):
bank.register_user("david", "pass123", 200)
assert bank.withdraw("david", 100) == "Withdrawn $100 successfully"
assert bank.withdraw("david", 200) == "Insufficient funds"
def test_transfer(bank):
bank.register_user("alice", "pass", 300)
bank.register_user("bob", "pass", 100)
assert bank.transfer("alice", "bob", 100) == "Transfer successful"
assert bank.transfer("alice", "bob", 300) == "Insufficient funds"

```python

def test_check_balance(bank):
    bank.register_user("eve", "password", 500)
    assert bank.check_balance("eve") == "Balance: $500"

def test_admin_approval(bank):
    assert bank.admins["admin"] == "admin123"

def test_loan_approval(bank):
    bank.register_user("frank", "secure", 500)
    assert bank.approve_loan("frank", 1000) == "Loan approved"
    assert bank.approve_loan("frank", 200) == "Loan approved"

def test_failed_login(bank):
    assert bank.login("unknown_user", "password") == "Invalid credentials"

def test_logout(bank):
    bank.register_user("grace", "mypassword")
    assert bank.logout("grace") == "User logged out"

```

Output for Functional Test Cases:

```
(myenv) karan@karansehgal-vivobook:~/WEB PROGRAMMING/Bank-Management-System/BankTest$ pytest test_bank_functional.py -v
========================================================================= test session starts =========================================================================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0 -- /home/karan/WEB PROGRAMMING/Bank-Management-System/BankTest/myenv/bin/python3
cachedir: .pytest_cache
rootdir: /home/karan/WEB PROGRAMMING/Bank-Management-System/BankTest
collected 10 items

test_bank_functional.py::test_register_user PASSED                                                                                                                  [ 10%]
test_bank_functional.py::test_login PASSED                                                                                                                          [ 20%]
test_bank_functional.py::test_deposit PASSED                                                                                                                        [ 30%]
test_bank_functional.py::test_withdraw PASSED                                                                                                                       [ 40%]
test_bank_functional.py::test_transfer PASSED                                                                                                                       [ 50%]
test_bank_functional.py::test_check_balance PASSED                                                                                                                  [ 60%]
test_bank_functional.py::test_admin_approval PASSED                                                                                                                 [ 70%]
test_bank_functional.py::test_loan_approval PASSED                                                                                                                  [ 80%]
test_bank_functional.py::test_failed_login PASSED                                                                                                                   [ 90%]
test_bank_functional.py::test_logout PASSED                                                                                                                         [100%]

========================================================================== 10 passed in 0.01s ==========================================================================
```

PyTest Code for *Non-Functional* Test Cases:

EXPLORER

BANKTEST

- __pycache__
  - bank_model.cpython-312.pyc
  - bank_system.cpython-312.pyc
  - test_bank_functional.cpython-312...
  - test_bank_nonfunctional.cpython...
  - test_bank_system.cpython-312-py...
- .pytest_cache
- myenv
- bank_model.py
- test_bank_functional.py
- test_bank_nonfunctional.py

bank_model.py ● | test_bank_functional.py | test_bank_nonfunctional.py ×

test_bank_nonfunctional.py > test_sql_injection > search

```python
# test_bank_nonfunctional.py
import pytest
import time
from bank_model import Bank

@pytest.fixture
def bank():
    return Bank()

def test_response_time(bank):
    start = time.time()
    bank.register_user("hank", "securepass")
    end = time.time()
    assert (end - start) < 0.5  # Should execute in under 0.5s

def test_concurrent_users(bank):
    users = [f"user{i}" for i in range(100)]
    for user in users:
        bank.register_user(user, "pass")
    assert len(bank.users) == 100

def test_sql_injection(bank):
    def search(query):
        if any(char in query for char in ["'", "--", ";"]):
            return "Invalid input detected"
        return "Search results"

    result = search("' OR 1=1 --")
    assert result == "Invalid input detected"

def test_xss_profile_update(bank):
    script_input = "<script>alert('XSS')</script>"
    bank.register_user("sam", "safe")
    assert bank.register_user(script_input, "pass") == "User registered successfully"
    assert "<script>" not in bank.users  # XSS prevention
```

EXPLORER

BANKTEST

- __pycache__
  - bank_model.cpython-312.pyc
  - bank_system.cpython-312.pyc
  - test_bank_functional.cpython-312...
  - test_bank_nonfunctional.cpython...
  - test_bank_system.cpython-312-py...
- .pytest_cache
- myenv
- bank_model.py
- test_bank_functional.py
- test_bank_nonfunctional.py

bank_model.py ● | test_bank_functional.py | test_bank_nonfunctional.py ×

test_bank_nonfunctional.py > test_sql_injection > search

```python
def test_system_availability(bank):
    assert isinstance(bank, Bank)  # Ensures system is running

def test_data_recovery(bank):
    bank.register_user("peter", "pass", 200)
    bank.withdraw("peter", 50)
    assert bank.check_balance("peter") == "Balance: $150"

def test_usability(bank):
    assert bank.register_user("12345", "pass") == "User registered successfully"

def test_large_input(bank):
    large_username = "a" * 1000
    assert bank.register_user(large_username, "pass") == "User registered successfully"

def test_invalid_input(bank):
    assert bank.deposit("unknown", 100) == "Deposit failed"

def test_high_load(bank):
    start = time.time()
    for i in range(1000):
        bank.register_user(f"user{i}", "pass")
    end = time.time()
    assert (end - start) < 5  # Registers 1000 users in under 5 seconds
```

Output for Functional Test Cases:

```
(myenv) karan@karansehgal-vivobook:~/WEB PROGRAMMING/Bank-Management-System/BankTest$ pytest test_bank_nonfunctional.py -v
========================================================================== test session starts ==========================================================================
platform linux -- Python 3.12.3, pytest-8.3.5, pluggy-1.5.0 -- /home/karan/WEB PROGRAMMING/Bank-Management-System/BankTest/myenv/bin/python3
cachedir: .pytest_cache
rootdir: /home/karan/WEB PROGRAMMING/Bank-Management-System/BankTest
collected 10 items

test_bank_nonfunctional.py::test_response_time PASSED                                                                                                          [ 10%]
test_bank_nonfunctional.py::test_concurrent_users PASSED                                                                                                       [ 20%]
test_bank_nonfunctional.py::test_sql_injection PASSED                                                                                                          [ 30%]
test_bank_nonfunctional.py::test_xss_profile_update PASSED                                                                                                     [ 40%]
test_bank_nonfunctional.py::test_system_availability PASSED                                                                                                    [ 50%]
test_bank_nonfunctional.py::test_data_recovery PASSED                                                                                                          [ 60%]
test_bank_nonfunctional.py::test_usability PASSED                                                                                                              [ 70%]
test_bank_nonfunctional.py::test_large_input PASSED                                                                                                            [ 80%]
test_bank_nonfunctional.py::test_invalid_input PASSED                                                                                                          [ 90%]
test_bank_nonfunctional.py::test_high_load PASSED                                                                                                              [100%]

========================================================================== 10 passed in 0.01s ==========================================================================
```