Определение типов тестирования
Юнит-тесты

Цель: Проверка функциональности отдельных модулей, таких как регистрация пользователей, добавление книг, управление отзывами.

Основные компоненты для тестирования:

Регистрация пользователя (проверка уникальности имени пользователя и сложности пароля).

Вход в систему (проверка аутентификации с верными и неверными учетными данными).

Добавление и просмотр книг.

Добавление и просмотр отзывов.

Логин и пароль
andrey
123

Интеграционные тесты

Цель: Проверка взаимодействия между компонентами, таких как соединение интерфейса пользователя с базой данных.

Основные модули для тестирования:

Взаимодействие между логикой приложения и базой данных.

Проверка корректного отображения данных в интерфейсе (например, список книг и отзывы).

Функциональные тесты

Цель: Проверка, что функциональность приложения работает согласно требованиям.

Функциональные проверки:

Успешное завершение регистрации, входа в систему, добавления книг и отзывов.

Проверка доступности всех функций и корректного отображения сообщений об ошибках.

Создание тест-кейсов для каждой функциональной единицы

Тест-кейсы

Регистрация пользователя

Шаги: ввести уникальное имя пользователя и надежный пароль; нажать "Регистрация".

Ожидаемый результат: появляется сообщение об успешной регистрации; данные сохраняются в базе.

Вход пользователя

Шаги: ввести зарегистрированное имя пользователя и пароль; нажать "Вход".

Ожидаемый результат: переход в главное окно приложения; корректная загрузка данных пользователя.

Добавление новой книги

Шаги: ввести название и автора книги; нажать "Добавить книгу".

Ожидаемый результат: появляется сообщение об успешном добавлении книги; книга отображается в общем списке.

Просмотр списка книг

Шаги: открыть главное окно и нажать "Просмотреть книги".

Ожидаемый результат: в списке отображаются все добавленные книги с корректными данными.

Добавление отзыва на книгу

Шаги: ввести ID книги, текст отзыва и оценку; нажать "Добавить отзыв".

Ожидаемый результат: появляется сообщение об успешном добавлении отзыва; отзыв сохраняется в базе данных.

Просмотр отзывов на книгу

Шаги: ввести ID книги и нажать "Просмотреть отзывы".

Ожидаемый результат: в списке отзывов отображаются все отзывы к выбранной книге с правильными данными.

Критерии успешного тестирования

Все тест-кейсы выполняются успешно без критических ошибок.

Данные корректно сохраняются и отображаются, как в базе данных, так и в интерфейсе.

Ошибки и исключения обрабатываются корректно (например, сообщения об ошибках при неверных учетных данных или пустых полях).

Все элементы интерфейса работают в соответствии с функциональными требованиями, и переход между экранами происходит плавно.

Интерфейс приложения остается стабильным и не закрывается самопроизвольно при выполнении действий.

Настройка среды для автоматизированного тестирования

Пошаговая настройка среды автоматизированного тестирования:

Установка инструментов для тестирования

Для начала нужно установить необходимые библиотеки:

Pytest: основной инструмент для запуска юнит- и интеграционных тестов.

Selenium: для автоматизированного тестирования интерфейса.

Pytest-cov: для покрытия кода тестами, чтобы отслеживать, сколько логики приложения покрыто тестами.

Сперва нужно прописать в терминале PyCharm:
pip install pytest pytest-cov selenium pyodbc

Конфигурация драйвера Selenium

Настройка базы данных для тестирования

Создается тестовая копию моей базы данных (например, BookExchangeDB_Test).

Нужно будет обновить строку подключения в database.py, добавив возможность переключения между главной и тестовой базами с помощью переменной окружения.

```
import os

import pyodbc

def connect_to_db():

    db_name = os.getenv("DB_NAME", "BookExchangeDB")

    conn = pyodbc.connect(f'DRIVER={{SQL
Server}};SERVER=localhost\\SQLEXPRESS;DATABASE={db_name
};Trusted_Connection=yes')

    return conn
```

Теперь для запуска тестов используем тестовую базу данных, установив переменную окружения:

```
export DB_NAME=BookExchangeDB_Test
```

Создание тестов с Pytest

Пример юнит-теста для функции добавления пользователя

В папке tests создается файл test_database.py и добавляется тест функции register_user

```
import pytest

from database import connect_to_db, register_user

@pytest.fixture

def db_connection():
```

```
    conn = connect_to_db()
    yield conn
    conn.close()

def test_register_user(db_connection):
    result = register_user(db_connection, "test_user", "test_password")
    assert result is True
```

Интеграционный тест для добавления книги

Создается тест для добавления функции add_book в том же файле

```
from database import add_book

def test_add_book(db_connection):
    result = add_book(db_connection, "Test Book", "Test Author")
    assert result is True
```

Тестирование интерфейса с Selenium

Для начала нужно будет прописать в терминале

pip install pytest selenium

Для того чтобы установились библиотеки pytest и selenium

В папке tests создается файл test_ui.py

```
from selenium import webdriver
import pytest

@pytest.fixture
def browser():
    driver = webdriver.Chrome()
    driver.get("http://localhost:5000")  # Убедитесь, что ваше приложение работает локально
```

```
    yield driver
    driver.quit()

def test_login_page(browser):
    username_input = browser.find_element_by_name("username")
    password_input = browser.find_element_by_name("password")
    login_button = browser.find_element_by_name("login_button")

    username_input.send_keys("test_user")
    password_input.send_keys("test_password")
    login_button.click()

    assert "Главное окно" in browser.page_source
```

После чего соответственно производится тест

В целом на этом все, вроде бы ничего не забыл.