**Цель лабораторной работы**

Заключается в приобретении навыков создания автотестов для контроля функциональности программных систем. Автоматизированное тестирование позволяет увеличить скорость контроля качества в процессе разработки программных продуктов. Для выполнения работы необходимо использовать ранее разработанную тестовую документацию, включая тест-кейсы.

**Описание реализованных автотестов**

Автотесты написаны на языке Python с использованием модуля **unittest**. В данном случае используется подход модульного тестирования.

**Инструменты:**

* Python: язык программирования, на котором написаны тесты.
* unittest: стандартный модуль тестирования в Python.

**Подходы:**

* **Модульное тестирование:** Каждый тест проверяет отдельную часть функционала (функцию **register\_user**), изолируя ее от других частей системы. Это позволяет выявить дефекты и проверить правильность работы отдельных компонентов.
* **Тестирование на основе методов:** Для каждого тестового случая используется метод **assertTrue()** или **assertFalse()** для проверки ожидаемого результата (True/False).
* **Тестирование с использованием assert-утверждений:** assert-утверждения используются для проверки фактического результата выполнения функции **register\_user**. Например, **self.assertTrue(result, "Valid name should allow registration")** проверяет, что при передаче валидных данных функция **register\_user** возвращает True.
* **Разделение тестов на категории:** Тесты разделены на категории в соответствии с полями формы регистрации, что позволяет более структурированно организовать тестовый набор и легче находить проблемы в конкретных областях функционала.

Эти подходы обеспечивают хорошую организацию и структурированность тестов, что упрощает их поддержку и расширение в будущем.

**Код автотестов**

import unittest  
  
  
def register\_user(name, email, password, confirm\_password):  
 # Проверка наличия пустых значений  
 if not name or not email or not password or not confirm\_password:  
 return False  
  
 # Проверка валидности email  
 if '@' not in email or '.' not in email:  
 return False  
  
 # Проверка соответствия паролей  
 if password != confirm\_password:  
 return False  
  
 # Все проверки пройдены, пользователь успешно зарегистрирован  
 return True  
  
  
class UserRegistrationTests(unittest.TestCase):  
  
 def test\_empty\_name\_field(self):  
 result = register\_user("", "example@example.com", "Password123", "Password123")  
 self.assertFalse(result, "Empty name field should not allow registration")  
  
 def test\_valid\_name(self):  
 result = register\_user("Иванов Иван Иванович", "example@example.com", "Password123", "Password123")  
 self.assertTrue(result, "Valid name should allow registration")  
  
 def test\_invalid\_name(self):  
 result = register\_user("123", "example@example.com", "Password123", "Password123")  
 self.assertFalse(result, "Invalid name should not allow registration")  
  
 def test\_empty\_email\_field(self):  
 result = register\_user("Иванов Иван Иванович", "", "Password123", "Password123")  
 self.assertFalse(result, "Empty email field should not allow registration")  
  
 def test\_valid\_email(self):  
 result = register\_user("Иванов Иван Иванович", "example@example.com", "Password123", "Password123")  
 self.assertTrue(result, "Valid email should allow registration")  
  
 def test\_invalid\_email(self):  
 result = register\_user("Иванов Иван Иванович", "example.com", "Password123", "Password123")  
 self.assertFalse(result, "Invalid email should not allow registration")  
  
 def test\_empty\_password\_field(self):  
 result = register\_user("Иванов Иван Иванович", "example@example.com", "", "")  
 self.assertFalse(result, "Empty password field should not allow registration")  
  
 def test\_valid\_password(self):  
 result = register\_user("Иванов Иван Иванович", "example@example.com", "Password123", "Password123")  
 self.assertTrue(result, "Valid password should allow registration")  
  
 def test\_invalid\_password(self):  
 result = register\_user("Иванов Иван Иванович", "example@example.com", "123", "123")  
 self.assertFalse(result, "Invalid password should not allow registration")  
  
 def test\_empty\_confirm\_password\_field(self):  
 result = register\_user("Иванов Иван Иванович", "example@example.com", "Password123", "")  
 self.assertFalse(result, "Empty confirm password field should not allow registration")  
  
 def test\_matching\_passwords(self):  
 result = register\_user("Иванов Иван Иванович", "example@example.com", "Password123", "Password123")  
 self.assertTrue(result, "Matching passwords should allow registration")  
  
 def test\_non\_matching\_passwords(self):  
 result = register\_user("Иванов Иван Иванович", "example@example.com", "Password123", "Password321")  
 self.assertFalse(result, "Non-matching passwords should not allow registration")

**Отчет о тестировании**

**Выполненные тест-кейсы**

В ходе тестирования были выполнены следующие тест-кейсы:

1. **Тестирование поля "Имя"**
   * Проверка пустого значения: ОШИБКА
   * Проверка валидного имени: УСПЕШНО
   * Проверка невалидного имени: УСПЕШНО
2. **Тестирование поля "Email"**
   * Проверка пустого значения: УСПЕШНО
   * Проверка валидного email: УСПЕШНО
   * Проверка невалидного email: УСПЕШНО
3. **Тестирование поля "Пароль"**
   * Проверка пустого значения: УСПЕШНО
   * Проверка валидного пароля: УСПЕШНО
   * Проверка невалидного пароля: ОШИБКА
4. **Тестирование поля "Подтверждение пароля"**
   * Проверка пустого значения: УСПЕШНО
   * Проверка пароля, совпадающего с введенным: УСПЕШНО
   * Проверка пароля, не совпадающего с введенным: УСПЕШНО

**Результат**

Из 12 выполненных тестов, 10 завершились успешно, а 2 завершились с ошибкой.

**Выявленные дефекты**

1. **Тестирование невалидного имени:**
   * При вводе невалидного имени (например, "123") ожидалось, что регистрация пользователя не будет выполнена, но функция **register\_user** вернула **True**, что привело к ошибке. Требуется доработка функции **register\_user** для корректной обработки невалидных имен.
2. **Тестирование невалидного пароля:**
   * При вводе невалидного пароля (например, "123") ожидалось, что регистрация пользователя не будет выполнена, но функция **register\_user** вернула **True**, что привело к ошибке. Требуется доработка функции **register\_user** для корректной обработки невалидных паролей.

**Выводы по работе**

В ходе выполнения данной лабораторной работы были приобретены навыки создания автотестов для контроля функциональности на основе разработанной ранее тестовой документации, тест-кейсах. Приобретен опыт применения таких инструментов тестирования как: python, unittest

**Список использованных источников**

ПИ21-7\_Очиров\_Алдар\_Управление\_разработкой\_04.04.2024\_лаб\_5.docx