Лабораторная работа №4

**Цель** данной лабораторной работы заключается в освоении методов тест-дизайна для проектирования эффективных наборов тестов, которые позволяют минимизировать количество тестов и их прогонов при максимальном контроле качества реализации программных продуктов. Основное внимание уделяется методам анализа эквивалентных классов и граничных значений.

**1. Выделение эквивалентных классов для формы регистрации**

* **Имя пользователя**
  + Валидный класс: строки длиной от 3 до 20 символов, содержащие только буквы и цифры.
  + Не валидный класс: строки короче 3 символов, длиннее 20 символов, содержащие специальные символы или пробелы.
* **Электронная почта**
  + Валидный класс: строки в формате "адрес@домен".
  + Не валидный класс: строки без символа "@", без домена, с пробелами.
* **Пароль**
  + Валидный класс: строки длиной от 8 символов, содержащие как минимум одну букву и одну цифру.
  + Не валидный класс: строки короче 8 символов, строки без цифр, строки без букв.
* **Подтверждение пароля**
  + Валидный класс: строка, идентичная введенному паролю.
  + Не валидный класс: строка, не совпадающая с введенным паролем.

**2. Расчет количества тестов**

Для минимизации количества тестов, необходимо провести тестирование хотя бы одного представителя из каждого валидного класса и хотя бы одного представителя из каждого не валидного класса, а также проверить граничные условия.

* Имя пользователя: 2 валидных теста (минимальная и максимальная допустимая длина) + 3 не валидных (слишком короткое, слишком длинное, с недопустимыми символами) = 5 тестов.
* Электронная почта: 1 валидный тест + 3 не валидных (без @, без домена, с пробелами) = 4 теста.
* Пароль: 2 валидных теста (минимально допустимая длина с буквами и цифрами, сложный пароль) + 3 не валидных (слишком короткий, без цифр, без букв) = 5 тестов.
* Подтверждение пароля: 1 валидный тест (совпадение с паролем) + 1 не валидный (несовпадение) = 2 теста.

Итого, минимальное количество тестов для проверки формы: **16 тестов**.

#### Список используемых тест-кейсов

1. **Имя пользователя**
   * Тест на минимально допустимую длину (3 символа).
   * Тест на максимально допустимую длину (20 символов).
   * Тест на длину менее минимально допустимой (2 символа).
   * Тест на длину более максимально допустимой (21 символ).
   * Тест с использованием недопустимых символов.
2. **Электронная почта**
   * Тест на корректный формат адреса.
   * Тест без символа "@".
   * Тест без домена.
   * Тест с пробелами в адресе.
3. **Пароль**
   * Тест на минимально допустимую длину с буквами и цифрами (8 символов).
   * Тест на сложный пароль (буквы, цифры, спецсимволы).
   * Тест на длину менее 8 символов.
   * Тест без цифр.
   * Тест без букв.
4. **Подтверждение пароля**
   * Тест на совпадение с паролем.
   * Тест на несовпадение с паролем.

**Ответы на контрольные вопросы**

1. **Методика выделения эквивалентных классов** заключается в разделении всего множества возможных входных данных на классы, в пределах которых все данные считаются эквивалентными и вызывают одинаковую реакцию системы. Цель - минимизировать количество тестов, выбирая для проверки лишь некоторые представительные данные из каждого класса.
2. **Цель тестирования граничных значений** - проверка реакции системы на данные, находящиеся вблизи границ допустимых значений входных данных. Часто ошибки возникают именно на границах или за пределами ожидаемого диапазона значений, поэтому этот метод позволяет эффективно идентифицировать потенциальные проблемы.
3. **Методика черного ящика** предполагает тестирование функциональности программы без знания внутреннего устройства программного кода. Тестировщик сосредоточен на том, чтобы входные данные приводили к ожидаемым результатам, основываясь на спецификациях функционала, не углубляясь в то, как именно программа достигает этих результатов.
4. **Разница между методиками черного, белого и серого ящиков** заключается в уровне знаний тестировщика о внутреннем устройстве программного продукта.
   * **Черный ящик**: тестирование основано исключительно на анализе функциональности без знания внутреннего строения.
   * **Белый ящик**: предполагает полное знание внутренней структуры и алгоритмов работы программы, тесты создаются на основе анализа кода.
   * **Серый ящик**: комбинирует подходы черного и белого ящиков, тестировщик имеет частичное знание о внутреннем устройстве системы, что позволяет более эффективно тестировать продукт, используя преимущества обоих методов.
5. **Тест-дизайн** — это процесс планирования тестов, включающий в себя определение целей тестирования, выбор методов и подходов к тестированию, разработку тестовых случаев и процедур. Основная цель тест-дизайна - обеспечение эффективного и всестороннего тестирования продукта с минимальными затратами времени и ресурсов. Тест-дизайн помогает идентифицировать и изолировать дефекты, гарантировать качество продукта и повысить уверенность в его надежности и безопасности.

**Выводы по работе**

Методы тест-дизайна, такие как анализ эквивалентных классов и тестирование граничных значений, позволяют эффективно планировать тестирование, сокращая количество необходимых тестов при сохранении высокого уровня контроля качества. Использование этих методов способствует повышению эффективности тестирования и оптимизации затрат времени и ресурсов.

Пример тестирования по логину, почте, паролю и валидации пароля:

import unittest  
  
import re  
  
  
def is\_valid\_username(username):  
 return 3 <= len(username) <= 20 and username.isalnum()  
  
  
def is\_valid\_email(email):  
 return "@" in email and "." in email.split("@")[1] and " " not in email  
  
  
def is\_valid\_password(password):  
 return len(password) >= 8 and re.search(r"\d", password) and re.search(r"[a**-**zA**-**Z]", password)  
  
  
def passwords\_match(password, confirmation):  
 return password == confirmation  
  
  
class TestUserForm(unittest.TestCase):  
  
 def test\_is\_valid\_username(self):  
 self.assertTrue(is\_valid\_username("User123"))  
 self.assertFalse(is\_valid\_username("Us"))  
 self.assertFalse(is\_valid\_username("a" \* 21)) # Слишком длинное имя  
 self.assertFalse(is\_valid\_username("user#name")) # Недопустимые символы  
  
 def test\_is\_valid\_email(self):  
 self.assertTrue(is\_valid\_email("user@example.com"))  
 self.assertFalse(is\_valid\_email("userexample.com"))  
 self.assertFalse(is\_valid\_email("user@example"))  
 self.assertFalse(is\_valid\_email("user@ example.com"))  
  
 def test\_is\_valid\_password(self):  
 self.assertTrue(is\_valid\_password("Password1"))  
 self.assertFalse(is\_valid\_password("pass")) # Слишком короткий  
 self.assertFalse(is\_valid\_password("password")) # Без цифр  
 self.assertFalse(is\_valid\_password("12345678")) # Без букв  
  
 def test\_passwords\_match(self):  
 self.assertTrue(passwords\_match("Password1", "Password1"))  
 self.assertFalse(passwords\_match("Password1", "Password2"))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 unittest.main()

python -m unittest .\TestUserForm.py