**Управление качеством программных систем**

**Мерзлова Анастасия, ПИ21-2**

**Лабораторная 4**

*Методы тест-дизайна*

## Цель работы:

Целью данной лабораторной работы является проведение анализа формы добавления животного на платформе приюта для животных, выделение эквивалентных классов и расчет количества необходимых тестов для проверки формы.

**Список используемых тест-кейсов:**

1.1 Добавление животного (Позитивный)

1.2 Редактирование животного (Позитивный)

1.3 Удаление животного (Позитивный)

2.1 Регистрация пользователя (Позитивный)

2.2 Авторизация пользователя (Позитивный)

2.3 Редактирование пользователя (Негативный)

2.4 Удаление пользователя (Позитивный)

2.5 Безопасность пользовательских данных (Негативный)

3.1 Поиск животного по имени (Позитивный)

3.2 Фильтрация (Позитивный)

4.1 Ввод некорректных данных (Позитивный)

4.2 Переход на страницу с животными (Позитивный)

4.3 Переход на главную страницу (Позитивный)

## Выделение эквивалентных классов для одной из форм приложения:

Для данной задачи выберем форму "Добавление животного".

Эквивалентные классы:

- Класс 1: Пользователь вводит корректные данные.

- Класс 2: Пользователь вводит некорректные данные (например, оставляет обязательные поля пустыми или вводит неправильный формат данных).

**Эквивалентный класс 1: Пользователь вводит корректные данные**

1. Тестирование на позитивный сценарий с корректными данными:

- Описание (Тип): Этот тест проверяет, как приложение обрабатывает корректные данные, введенные пользователем при добавлении животного.

- Предусловия: Пользователь на странице "Пристроить питомца".

- Шаги:

1. Пользователь заполняет форму с данными нового животного, включая имя, вид, возраст и другие необходимые поля.

2. Пользователь нажимает кнопку "Добавить животное".

- Ожидаемый результат: Система успешно добавляет нового животного в базу данных и отображает сообщение об успешном добавлении.

**Эквивалентный класс 2: Пользователь вводит некорректные данные**

1. Тестирование на негативный сценарий с некорректными данными:

- Описание (Тип): Этот тест проверяет, как приложение обрабатывает некорректные данные, введенные пользователем при добавлении животного.

- Предусловия: Пользователь на странице "Пристроить питомца".

- Шаги:

1. Пользователь оставляет обязательные поля пустыми или вводит данные неправильного формата (например, нечисловые значения в полях, где ожидаются числа).

2. Пользователь нажимает кнопку "Добавить животное".

- Ожидаемый результат: Система должна корректно обработать некорректные данные и отобразить соответствующие сообщения об ошибках на странице, указывая на обязательность заполнения определенных полей или формат ввода данных.

Выбор этих тестов обоснован тем, что они позволяют проверить базовую функциональность формы добавления животного на платформе. Первый тест покрывает основной позитивный сценарий, когда пользователь вводит корректные данные, а второй тест - проверяет, как приложение обрабатывает ошибочные данные и обеспечивает корректную обработку ошибок ввода данных.

## Расчет количества тестов для проверки формы приложения с учетом требования минимизации количества проводимых тестов:

Предположим, у нас есть 5 основных тестовых сценариев для формы "Добавление животного":

1. Позитивный сценарий для корректных данных.

2. Негативный сценарий для некорректных данных.

3. Позитивный сценарий для успешного добавления животного.

4. Негативный сценарий для неудачного добавления животного.

5. Сценарий для проверки обработки исключений.

Поскольку каждый сценарий представляет разные условия и ожидаемые результаты, мы можем ограничиться одним тестом для каждого эквивалентного класса. Таким образом, у нас будет 2 теста для проверки формы "Добавление животного".

## Выводы по работе:

В ходе выполнения лабораторной работы были выделены эквивалентные классы для формы добавления животного на платформе приюта для животных, составлены тест-кейсы для каждого класса, а также проведен расчет количества тестов, необходимых для проверки формы. Это позволит эффективно протестировать функциональность приложения и выявить возможные проблемы ввода данных.

## Контрольные вопросы:

1. Методика выделения эквивалентных классов:

Методика выделения эквивалентных классов используется для группировки входных данных или состояний программы на классы, внутри которых данные или состояния считаются эквивалентными. Цель этой методики - уменьшить количество тестов, необходимых для проверки программы, путем выбора представителей каждого класса для тестирования. Процесс выделения эквивалентных классов включает в себя анализ диапазона входных данных и определение значений, которые будут представлять каждый класс.

2. Цель тестирования граничных значений:

Цель тестирования граничных значений заключается в проверке поведения программы в краевых случаях, близких к границам допустимых значений входных данных. Это важно, потому что часто ошибки в программе проявляются именно при обработке граничных значений. Тестирование граничных значений позволяет выявить потенциальные проблемы и обеспечить более полное покрытие кода тестами.

3. Методика черного ящика:

Методика черного ящика, или методика функционального тестирования, основана на анализе функциональности программы без знания ее внутренней структуры. Тестировщики используют только внешние характеристики программы, такие как входные и выходные данные, для разработки и выполнения тестов. Они не обращают внимания на внутреннюю реализацию программы.

4. Разница между методиками черного, белого и серого ящиков:

- Черный ящик (Black Box): Тестировщик работает только с внешними характеристиками программы и не имеет доступа к ее внутренней структуре.

- Белый ящик (White Box): Тестировщик имеет доступ к внутренней структуре программы и использует это знание для разработки и выполнения тестов.

- Серый ящик (Grey Box): Сочетание черного и белого ящиков, где тестировщик имеет некоторое, но не полное знание о внутренней структуре программы.

5. Тест-дизайн:

Тест-дизайн представляет собой процесс разработки тестовых случаев на основе требований и спецификаций программы. Цель тест-дизайна - создание эффективных и полных тестов, которые позволяют проверить все аспекты программы. Тест-дизайн включает в себя выбор тестовых данных, определение последовательности действий для проведения теста и создание ожидаемых результатов для проверки.