Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве Российской Федерации»**

**(Финансовый университет)**

**Департамент анализа данных и машинного обучения**

**Отчёт**

по дисциплине «Управление качеством программных систем»

**Лабораторная работа №8**

Выполнила:

студентка учебной группы ПИ21-1 Агапенкова Д.А.

Научный руководитель:

*Клочков Евгений Юрьевич*

  Москва 2024

**Цель работы:**

Классификация видов тестирования отражает многообразие походов организации контроля качества программных систем. Понимание классификации позволяет определить возможные варианты процедур тестирования, оценить трудоемкость, спланировать тестирование, выбрать подходящие инструменты. Это знание является необходимым при построении жизненного цикла разработки программных систем. Навыки использования видов тестирования для контроля функционирования программных систем позволяют тестировщику быстрее локализовать дефекты, создавать более полные наборы тестов.

**1. Описание программного проекта**

Тестируется веб-приложение для стилизации изображений «Ден ь». Оно представляет из себя сервис, в который загружаются два изображения: требуемое стилизовать и стиль. Разработан сценарий тестирования. Его можно увидеть на ресурсе: <https://github.com/DashaAga/development-management/blob/main/%D0%9B%D0%B0%D1%8E%D0%BE%D0%BB%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%205.pdf>. Также были разработаны автоматические тесты с помощью Selenium WebDriver для Python.

Код тестируемой программы и тестов находится по ссылке:

https://github.com/den-corp

**2. Использование тестов**

1. Функциональное тестирование
   * Проверка, что все функции веб-приложения работают согласно требованиям.
   * Тестирование различных фильтров и эффектов стилизации изображений.
   * Проверка загрузки и сохранения изображений.
2. Нефункциональное тестирование
   * Производительное тестирование: Оценка скорости обработки изображений.
   * Тестирование удобства использования: Проверка, насколько интерфейс приложения интуитивен и удобен для пользователей.
   * Тестирование совместимости: Проверка работы приложения в разных браузерах и на разных устройствах.
3. Регрессионное тестирование
   * Проверка, что новые изменения в коде не нарушили существующую функциональность.
4. Приемочное тестирование
   * Проверка, что приложение соответствует ожиданиям и требованиям конечных пользователей.

Типы тестирования

1. Черный ящик (Black Box Testing)
   * Функциональное тестирование пользовательских сценариев без знания внутренней структуры кода.
2. Белый ящик (White Box Testing)
   * Проверка внутренней логики и структуры кода, может быть применено для модульных тестов.
3. Серый ящик (Gray Box Testing)
   * Тестирование с ограниченными знаниями о внутренней структуре, может быть использовано для интеграционных тестов.

Методы тестирования

1. Ручное тестирование
   * Проведение тестов вручную по заранее написанному сценарию.
   * Проверка интерфейса, пользовательских сценариев и взаимодействий с системой.
2. Автоматизированное тестирование
   * Использование Selenium WebDriver для автоматизации функциональных тестов.
3. Интеграционное тестирование
   * Проверка взаимодействия между различными компонентами приложения, такими как фронтенд и бэкенд.
4. Тестирование на основе сценариев (Scenario-based Testing)
   * Тестирование основано на заранее написанных сценариях использования системы.

Уровни тестирования

1. Модульное тестирование (Unit Testing)
   * Тестирование отдельных функций или методов, особенно если они содержат сложную логику обработки изображений.
2. Интеграционное тестирование (Integration Testing)
   * Проверка взаимодействия между различными модулями приложения, такими как загрузка изображений, применение фильтров и сохранение результатов.
3. Системное тестирование (System Testing)
   * Полное тестирование всей системы целиком, включая все интегрированные модули и компоненты.
4. Приемочное тестирование (Acceptance Testing)
   * Финальное тестирование для проверки соответствия системы требованиям и ожиданиям пользователей.

В итоге основное внимание будет уделено функциональному тестированию (проверка работы фильтров и эффектов), производительному тестированию (скорость стилизации изображений), и тестированию удобства использования (интерфейс).

Типы тестирования: Использование черного ящика для функциональных тестов с Selenium и ручных тестов для проверки сценариев.

Методы тестирования: Автоматизированное тестирование для повторяющихся сценариев с использованием Selenium WebDriver, и ручное тестирование для сценариев, требующих субъективной оценки, таких как удобство использования.

Уровни тестирования: Интеграционное тестирование для проверки взаимодействия различных частей приложения (фронтенд и бэкенд), и системное тестирование для проверки всей системы целиком.

**3. Примеры дефектов**

Для каждого вида тестирования приведем примеры типичных дефектов, которые могут быть обнаружены.

**Функциональное тестирование**

1. **Некорректная работа функций**
   * Фильтры и эффекты стилизации изображений не работают согласно спецификации.
   * Некорректное применение фильтра к изображению.
   * Ошибки при загрузке изображений (например, неподдерживаемые форматы не обрабатываются должным образом).
   * Проблемы с сохранением обработанных изображений (файл не сохраняется или сохраняется в неверном формате).

**Нефункциональное тестирование**

1. **Производительное тестирование**
   * Долгое время обработки изображений при применении фильтров.
   * Низкая производительность при большом количестве одновременных пользователей.
2. **Тестирование удобства использования**
   * Неинтуитивный пользовательский интерфейс (сложность нахождения нужных функций).
   * Неправильное или неполное отображение элементов интерфейса на разных устройствах.
3. **Тестирование совместимости**
   * Приложение неправильно отображается или не работает в определенных браузерах.
   * Некорректное отображение или работа приложения на мобильных устройствах.

**Регрессионное тестирование**

1. **Нарушение существующей функциональности**
   * Новые изменения в коде ломают ранее работавшие фильтры.
   * Добавление новой функции приводит к сбоям в уже реализованных сценариях загрузки и сохранения изображений.

**Приемочное тестирование**

1. **Несоответствие требованиям пользователя**
   * Приложение не поддерживает все заявленные форматы изображений, что было требованием клиента.
   * Интерфейс приложения не соответствует ожиданиям конечных пользователей по удобству и функциональности.

**Примеры дефектов для каждого типа тестирования**

1. **Черный ящик (Black Box Testing)**
   * Пользователь не может загрузить изображение (функция не работает, как ожидается).
   * Кнопка "Поехали" не активируется после применения фильтра.
2. **Белый ящик (White Box Testing)**
   * Неверная логика в коде фильтра приводит к ошибке обработки изображения.
   * Утечки памяти из-за неправильного управления ресурсами при обработке изображений.
3. **Серый ящик (Gray Box Testing)**
   * Некорректная передача данных между фронтендом и бэкендом, приводящая к ошибкам в применении фильтров.
   * Ошибки в API, которые становятся видны только при интеграции различных компонентов системы.

**Примеры дефектов для каждого метода тестирования**

1. **Ручное тестирование**
   * Обнаружение ошибки в интерфейсе, когда элементы перекрывают друг друга на определенных разрешениях экрана.
   * Проблемы с вводом данных в текстовые поля (например, невозможность ввести определенные символы).
2. **Автоматизированное тестирование**
   * Скрипт автоматического тестирования обнаруживает, что при определенных действиях возникает ошибка на сервере.
   * Тесты выявляют нестабильность работы приложения под нагрузкой (Selenium тесты выполняются медленнее, чем ожидается).
3. **Интеграционное тестирование**
   * При интеграции нового модуля для стилизации изображений возникают ошибки при обмене данными с существующими компонентами.
4. **Тестирование на основе сценариев (Scenario-based Testing)**
   * Сценарий использования, предусматривающий определенную последовательность действий, приводит к крашу приложения.
   * Несоответствие ожидаемого результата при выполнении сложного пользовательского сценария (например, загрузка, стилизация, сохранение, повторная загрузка).

**Примеры дефектов для каждого уровня тестирования**

1. **Модульное тестирование (Unit Testing)**
   * Тесты выявляют, что отдельная функция фильтрации изображений возвращает некорректный результат при определенных входных данных.
   * Неверные расчеты в алгоритме стилизации.
2. **Интеграционное тестирование (Integration Testing)**
   * Ошибки в интеграции модуля загрузки изображений с модулем стилизации, приводящие к сбоям.
   * Проблемы в совместной работе фронтенда и бэкенда при передаче данных изображений.
3. **Системное тестирование (System Testing)**
   * Полное тестирование системы выявляет, что при большом объеме данных (много изображений) система начинает работать нестабильно или медленно.
   * Обнаружение сбоев при одновременном использовании приложения несколькими пользователями.
4. **Приемочное тестирование (Acceptance Testing)**
   * Конечный пользователь обнаруживает, что определенный тип изображения не может быть обработан, хотя это было указано в требованиях.
   * Пользователи жалуются на неудобный процесс выбора и применения фильтров, что не соответствует их ожиданиям.

**4. Выводы и ответы на контрольные вопросы**

План тестирования поможет обеспечить всестороннюю проверку веб-приложения «Ден ь» и гарантировать, что оно работает правильно и удовлетворяет потребности пользователей. Примеры дефектов помогут лучше понять потенциальные проблемы в проекте и организовать эффективное тестирование для их выявления и устранения.

**Контрольные вопросы:**

**1. Определяется ли качество ПО качеством программного кода?**

Качество ПО определяется совокупностью различных характеристик, таких как:

* **Функциональность**: Соответствие ПО функциональным требованиям и спецификациям.
* **Надежность**: Способность ПО работать без сбоев в течение определенного времени.
* **Производительность**: Эффективность и скорость работы ПО.
* **Удобство использования**: Насколько ПО удобно и интуитивно для пользователя.
* **Поддерживаемость**: Легкость исправления ошибок и внесения изменений в ПО.
* **Безопасность**: Устойчивость ПО к внешним угрозам и атакам.

Таким образом, качество кода — это только одна из составляющих общего качества программного обеспечения.

**2. Какие существуют виды тестирования?**

* **Функциональное тестирование**: Проверка соответствия функционала программного обеспечения его спецификациям и требованиям.
* **Нефункциональное тестирование**: Оценка аспектов, не связанных с конкретным функционалом, таких как производительность, безопасность, удобство использования и т.д.
* **Регрессионное тестирование**: Проверка ПО после внесения изменений для подтверждения того, что новый код не нарушил существующую функциональность.
* **Приемочное тестирование**: Оценка ПО конечными пользователями или клиентами для проверки его соответствия их ожиданиям и требованиям.
* **Тестирование совместимости**: Проверка работы ПО в различных средах, на различных устройствах и операционных системах.

**3. Какие существуют типы тестирования?**

* **Черный ящик (Black Box Testing)**: Тестирование без знания внутренней структуры кода, фокусируется на входах и выходах системы.
* **Белый ящик (White Box Testing)**: Тестирование с учетом внутренней структуры и логики кода.
* **Серый ящик (Gray Box Testing)**: Комбинация тестирования черного и белого ящика, где тестировщик имеет ограниченные знания о внутренней структуре системы.

**4. Какие существуют методы тестирования?**

* **Ручное тестирование**: Проведение тестов вручную без использования автоматизированных инструментов.
* **Автоматизированное тестирование**: Использование программных инструментов для автоматизации процесса тестирования.
* **Инкрементное тестирование**: Тестирование модулей или компонентов ПО по мере их разработки.
* **Интеграционное тестирование**: Проверка взаимодействия между интегрированными модулями и компонентами.
* **Тестирование на основе сценариев (Scenario-based Testing)**: Тестирование, основанное на сценариях использования системы.

**5. Какие существуют уровни тестирования?**

* **Модульное тестирование (Unit Testing)**: Тестирование отдельных модулей или компонентов ПО.
* **Интеграционное тестирование (Integration Testing)**: Тестирование взаимодействия между модулями и компонентами.
* **Системное тестирование (System Testing)**: Полное тестирование всей системы в целом, включая все интегрированные модули.
* **Приемочное тестирование (Acceptance Testing)**: Финальное тестирование для проверки соответствия системы требованиям и ожиданиям пользователей.