Предмет Тестирование Приложений

Лабораторная работа №3

Тема: Автоматическое тестирование веб-приложения

Тестирование многостраничных веб-приложений, таких как интернет-магазины, является неотъемлемой частью процесса разработки, направленной на обеспечение высокого качества продукта и удовлетворение потребностей пользователей.

Основной целью тестирования является обеспечение высокого качества продукта. Этот процесс позволяет выявлять и устранять ошибки, дефекты и недочеты в функциональности веб-приложения, гарантируя, что оно работает надежно и без сбоев.

Тестирование также направлено на обеспечение надежной работы всех страниц приложения. Это включает в себя проверку корректности отображения основных элементов, правильной работы функционала, такого как поиск и фильтрация товаров, а также обработки заказов и платежей.

Безопасность приложения также находится в фокусе тестирования. Проверка на наличие уязвимостей, предотвращение SQL-инъекций, кросс-сайтового скриптинга и других угроз - все это важные аспекты тестирования безопасности

1. **Установка Selenium:**
    - Установите Selenium WebDriver для языка программирования, который вы собираетесь использовать (например, Java, Python, C#).
    - Загрузите драйвер браузера, с которым вы собираетесь взаимодействовать (например, ChromeDriver, GeckoDriver для Firefox).
2. **Настройка проекта:**
    - Создайте проект для автоматизации тестирования.
    - Добавьте библиотеку Selenium WebDriver в зависимости вашего проекта.
3. **Написание тестов:**
    - Используйте Selenium WebDriver API для написания тестов.
    - Создайте тестовые сценарии, которые воспроизводят типичные действия пользователя на различных страницах приложения (например, поиск товара, добавление товара в корзину, оформление заказа).
4. **Локаторы элементов:**
    - Используйте селекторы (CSS-селекторы, XPath) для нахождения веб-элементов на странице.
    - Обеспечьте уникальность локаторов для каждого элемента, чтобы избежать конфликтов при изменении верстки.
5. **Ожидания:**

- Вставляйте ожидания (wait) в код тестов для обработки асинхронных действий и загрузки страницы.
- Ожидания помогут избежать ложных срабатываний тестов из-за неполной загрузки страницы.

6. **Использование Page Object Pattern:**
    - Разделите логику взаимодействия с веб-элементами и тестовую логику с использованием шаблона Page Object.
    - Создайте классы Page Object для каждой страницы приложения, в которых будут храниться локаторы и методы для взаимодействия с элементами на странице.
7. **Запуск тестов:**
    - Напишите скрипты для запуска тестов в автоматическом режиме.
    - Можно использовать системы сборки, такие как Maven, Gradle, или CI/CD инструменты для автоматического запуска тестов.
8. **Анализ результатов:**
    - После выполнения тестов, проанализируйте результаты.
    - Задокументируйте ошибки, если они возникают, и предоставьте информацию для разработчиков.

Для тестирования многостраничных приложений требуется учитывать множество различных аспектов данного приложения. Ниже можно рассмотреть базисные пункты оформления тестов для приложения интернет магазина

1. **Тестирование функциональности:**
    - **Главная страница:**
        - Проверка отображения основных элементов (логотип, навигационные ссылки, поиск).
        - Тестирование работы главного меню и подменю.
        - Проверка функционала быстрого доступа к разделам (акции, новинки и т.д.).
    - **Страницы категорий и продуктов:**
        - Убедиться, что продукты отображаются корректно.
        - Проверить работу фильтров и сортировки товаров.
        - Тестирование кнопок добавления товара в корзину.
    - **Корзина и оформление заказа:**
        - Проверка добавления/удаления товаров в корзине.
        - Тестирование работы купонов и скидок.
        - Проверка процесса оформления заказа до финального подтверждения.
    - **Личный кабинет пользователя:**

- Проверка регистрации и входа в систему.
- Тестирование функционала изменения личных данных.
- Проверка истории заказов и их статусов.

2. **Тестирование совместимости:**
    - Проверка работы в различных браузерах (Chrome, Firefox, Safari, Edge).
    - Тестирование на различных операционных системах (Windows, macOS, Linux).
3. **Тестирование производительности:**
    - Оценка времени загрузки страниц.
    - Тестирование при медленном интернет-соединении.
    - Проверка отзывчивости интерфейса при большом количестве одновременных пользователей.
4. **Тестирование безопасности:**
    - Проверка наличия SSL-сертификата.
    - Тестирование на уязвимости (SQL-инъекции, XSS-атаки и т.д.).
    - Проверка защищенности платежных операций.

Ход работы

1. Подготовьте новый проект для создания группы тестов для приложения https://demoqa.com/books
2. Разработайте список тестов для проверки функциональности
3. Разработайте список тестов для проверки базы данных (используйте API предоставленный на сайте)
4. Проведите тестирование по составленным тестам (7 тестов на функциональность 3 теста для базы данных 2 теста на безопасность)
5. Соберите и интерпретируйте результаты работы программы тестировщика (заполните таблицы тестирования).
6. Проанализируйте критичность результатов тестирования и подготовьте вывод
7. Заполните отчет по лабораторной работе.

По завершению работы, составьте отчет, в котором должно быть – Ваша фамилия, имя, группа, тема работы, краткое описание задания и информация о тестируемом приложении. Готовые отчеты сохраняем в формате PDF название в формате Имя_Фамилия_номер и название группы_Лабораторная_№.