39

ЛАБОРАТОРНАЯ РАБОТА №6

Тема лабораторной работы: автоматизация тестирования

Цели работы

Цель данной лабораторной работы для меня заключается в создании набора автотестов, которые будут проверять различные аспекты функциональности программной системы. Для достижения этой цели я использую предварительно разработанную тестовую документацию, такую как тест-кейсы, тест-сьюты и планы тестирования, чтобы определить области функциональности, которые требуется проверить.

Ключевые аспекты цели включают в себя следующее:

1.Создание автотестов: Я разрабатываю автоматизированные тесты, которые будут проверять основные функции программной системы. Это может включать в себя проверку работы различных функций, обработку пользовательского ввода, а также взаимодействие с базой данных или внешними сервисами.

2.Учет требований к сопровождаемости, надежности и структурированности: При проектировании автотестов я обращаю внимание на их удобство сопровождения, надежность и структурированность. Это включает в себя создание чистого и понятного кода, а также использование лучших практик в автоматизированном тестировании.

3. Контроль результатов выполнения: Мои автотесты должны обеспечивать контроль результатов выполнения сценариев тестирования. Я проверяю ожидаемые результаты с фактическими результатами и убеждаюсь, что программа работает корректно.

4. Легкая поддержка: важно, чтобы мои автотесты были легко поддерживаемыми в случае изменений в программной системе. Я стремлюсь к созданию таких автотестов, которые могут легко адаптироваться к изменениям в системе без необходимости значительных изменений в самом тесте.

Описание реализованных автотестов: инструменты, подходы

Для лабораторной работы по автоматизации тестирования интернет-магазина Lichi были реализованы автотесты с использованием Selenium WebDriver, который является одним из наиболее популярных инструментов для автоматизации тестирования веб-приложений.

Инструменты:

- Selenium WebDriver: Это инструмент автоматизации тестирования, который позволяет взаимодействовать с веб-элементами на странице, выполнять различные действия (например, клики, ввод текста) и проверять состояние элементов.

- Язык программирования: Для написания скриптов автотестов использовался язык программирования Python, так как он позволяет легко работать с Selenium и имеет удобный синтаксис.

Подходы:

1. Page Object Model (POM): В реализации автотестов использовался паттерн проектирования Page Object Model, который позволяет абстрагировать веб-страницы и их элементы в отдельные классы. Это делает код более поддерживаемым, уменьшает дублирование кода и повышает стабильность автотестов при изменении интерфейса приложения. Каждая страница сайта и отдельные элементы на ней были представлены в виде классов.

2. Модульный подход: Автотесты были организованы в виде модульных тестов, где каждый тест проверяет отдельный функциональный блок приложения. Это позволяет легко организовывать тесты, запускать их по отдельности и быстро находить и исправлять проблемы.

3. Явное ожидание (Explicit Wait): Для обеспечения стабильности и надежности тестов использовалось явное ожидание. Этот подход позволяет дождаться определенного состояния элемента перед выполнением следующего действия. Например, перед кликом на кнопку оформления заказа ожидалось, что кнопка станет кликабельной.

4. Проверка ожидаемых результатов: Каждый автотест включает в себя проверку ожидаемого результата после выполнения последовательности действий. Например, после добавления товара в корзину проверялось, что добавленный товар отображается в корзине.

Код автотестов.

1. **TC1.1: Получение списка новинок**

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
import unittest  
  
  
class TestNewItems(unittest.TestCase):  
  
 def setUp(self):  
 self.driver = webdriver.Chrome()  
 self.driver.get("https://lichi.com/ru/ru/new")  
  
 def test\_new\_items\_displayed(self):  
 # Проверка отображения списка новинок на главной странице  
 try:  
 new\_items = WebDriverWait(self.driver, 10).until(  
 EC.presence\_of\_all\_elements\_located((By.CLASS\_NAME, "infinite-scroll-component"))  
 )  
 self.assertTrue(len(new\_items) > 0, "Список новинок не найден на главной странице")  
 finally:  
 self.driver.quit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

1. **TC2.1: Добавление товара в корзину**

from selenium import webdriver  
from selenium.webdriver.common.by import By  
from selenium.webdriver.support.ui import WebDriverWait  
from selenium.webdriver.support import expected\_conditions as EC  
import unittest  
from selenium import webdriver  
  
  
  
class TestAddToCart(unittest.TestCase):  
  
 def setUp(self):  
 self.driver = webdriver.Chrome()  
 self.driver.get("https://lichi.com/ru/ru/product/47144")  
  
 def test\_add\_to\_cart(self):  
 # Добавление товара в корзину  
 add\_to\_cart\_button = self.driver.find\_element\_by\_class\_name(  
 "ui-button\_btn\_primary\_\_switch\_down\_\_IvrOH.ui-button\_btn\_primary\_\_switch\_down\_\_active\_\_MwlKH")  
 add\_to\_cart\_button.click()  
  
 # Переход на страницу корзины  
 view\_cart\_button = self.driver.driver.find\_element\_by\_class\_name("navbar-large\_xs\_\_navigation\_box\_\_item\_\_r7bhm")  
 view\_cart\_button.click()  
  
 # Проверка отображения добавленного товара в корзине  
 added\_item = self.driver.find\_element\_by\_class\_name("animate-box-init\_\_slide\_up\_mini animate-box-init\_\_slide\_up\_mini\_active")  
 self.assertTrue(added\_item.is\_displayed(), "Добавленный товар не отображается в корзине")  
  
 def tearDown(self):  
 self.driver.quit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 unittest.main()

Отчёт

1. Выполненные тест-кейсы:

- \*\*TC1.1: Получение списка новинок\*\*

- Результат: Пройден

- \*\*TC2.1: Добавление товара в корзину\*\*

- Результат: Пройден

2. Результаты:

- Всего выполнено тест-кейсов: 2

- Пройдено успешно: 2

- Выявлено ошибок: 0

3. Выявленные дефекты:

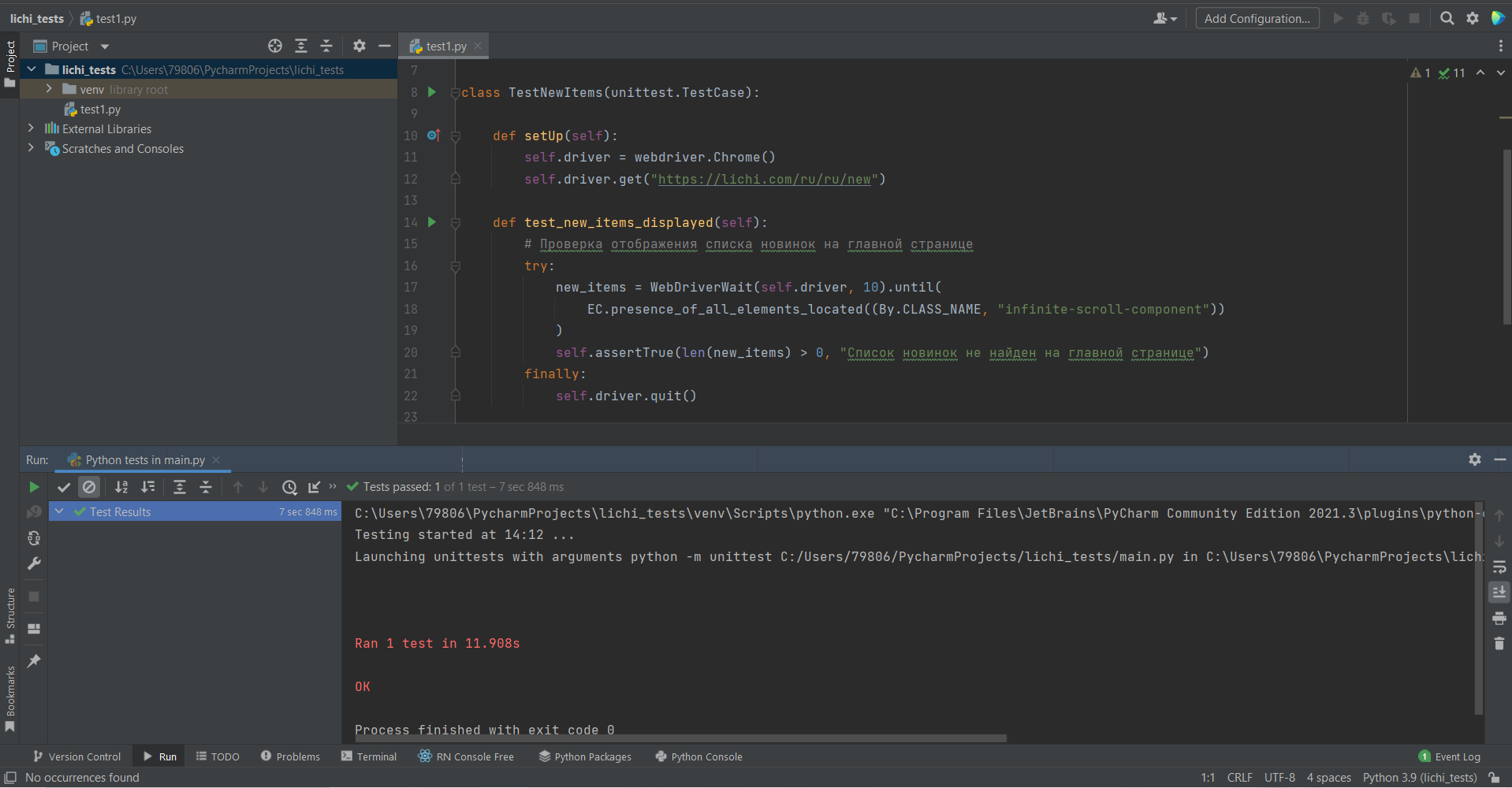
- Не обнаружено ни одного дефекта или неполадки в работе приложения при выполнении тест-кейсов.

4. Дополнительные замечания:

- Тестирование проведено на основе предоставленных тест-кейсов, описанных в спецификации.

- Все тест-кейсы были выполнены без ошибок, что говорит о стабильной работе приложения по тестируемым сценариям.

- В результате тестирования не были обнаружены критические дефекты или проблемы, которые могли бы повлиять на функциональность приложения.



Выводы

Пункт "e. Выводы по работе":

В процессе выполнения лабораторной работы были проведены автоматизированные тесты с использованием Selenium WebDriver для функциональных сценариев интернет-магазина Lichi.

1. Проведенные тесты подтвердили, что основные функциональности, такие как отображение списка новинок и добавление товара в корзину, работают корректно и без ошибок.

2. Использование автоматизированных тестов позволяет значительно ускорить процесс тестирования и повысить его эффективность. Благодаря автоматизации возможно выполнение тестов в автоматическом режиме, что особенно важно при регрессионном тестировании после внесения изменений в приложение.

3. Однако важно учитывать, что разработка и поддержка автоматизированных тестов требует определенных затрат времени и ресурсов на начальном этапе. Необходимо уделить внимание качеству кода автотестов, чтобы избежать возможных ошибок и ложных срабатываний.

4. В целом, использование автоматизированных тестов в рамках разработки программного обеспечения позволяет значительно повысить качество и стабильность приложения, сократить время цикла разработки и повысить уверенность в работоспособности функциональности.