Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
им. Н.П. ОГАРЁВА»
(ФГБОУ ВО «МГУ им. Н.П. Огарёва»)

Факультет математики и информационных технологий

Кафедра фундаментальной информатики

ЛАБОРАТОРНАЯ РАБОТА №3

по дисциплине: Методы тестирования программных продуктов

Автор отчета по лабораторной работе: _____
подпись, дата

Обозначение работы:

Направление подготовки 02.04.02 Фундаментальная информатика и информационные технологии

Руководитель работы канд. техн. наук _____
подпись, дата

Саранск 2025

1. Описание предмета тестирования

https://www.tesla.com/ - сайт предназначен для представления и продажи электромобилей, а также предлагает другие продукты и услуги, включая солнечные панели и решения для хранения энергии. Основные функции сайта включают просмотр моделей автомобилей, их конфигурирование, заказ тест-драйва, покупку аксессуаров и получение информации об услугах и поддержке.

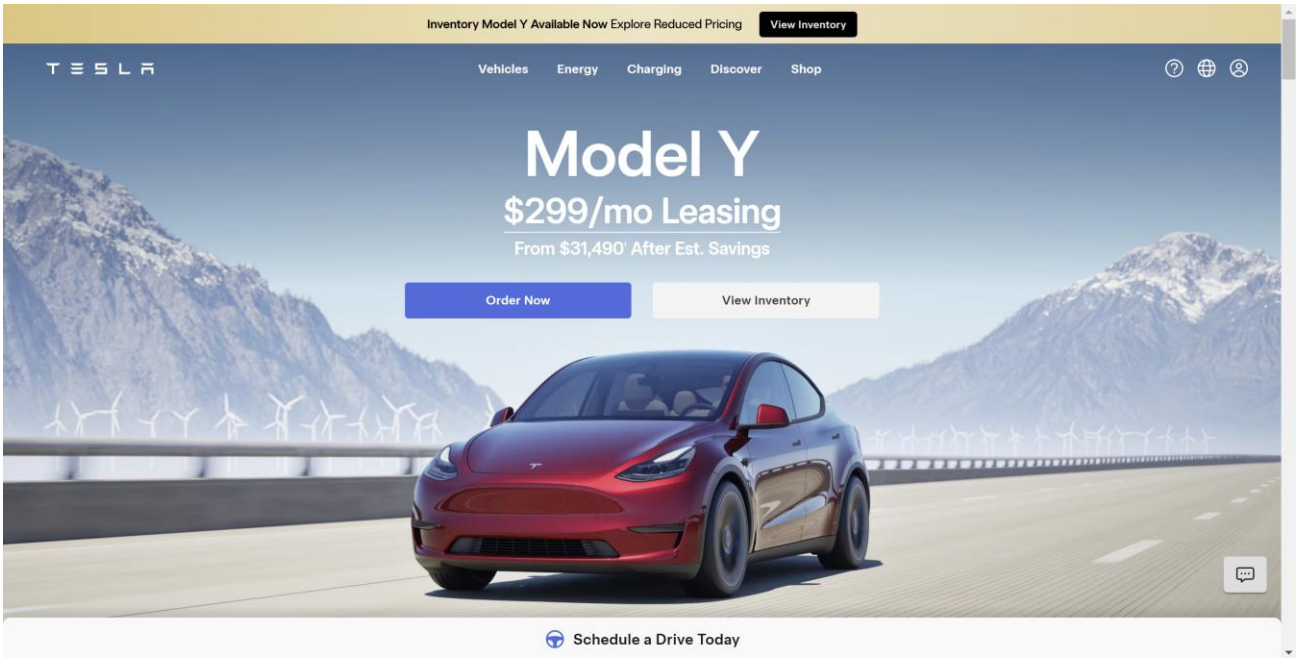


Рисунок 1 — Главная страница сайта

2. Описание окружения тестирования

Компьютер: Asus Zenbook 14X OLED 2023

- Процессор: Intel Core i9-13900H 2.60 GHz
- Оперативная память: 16 Гб, LPDDR5
- Жесткий диск: SSD 512 ГБ
- Видеокарта: Intel Iris Xe

Операционная система: Windows 11 Pro, версия 24H2 (сборка 26100.3194)

Браузер: Google Chrome, версия 134.0.6998.89

- Установленные расширения: AdBlock (включен), Allow CORS (отключен для тестирования)

Антивирусное ПО: Windows Defender (активен, настройки по умолчанию)

Интернет-соединение: Wi-Fi, скорость 100 Мбит/с (стабильное соединение)

Разрешение экрана: 2880x1800 пикселей

Инструмент для автоматизированного тестирования:

Selenium IDE — это инструмент с открытым исходным кодом для автоматизации тестирования веб-приложений. Он представляет собой расширение для браузеров (например, Chrome или Firefox), которое позволяет записывать, редактировать и воспроизводить тесты, которые в дальнейшем можно экспортировать например в файлы python.

3. use cases (пользовательские сценарии)

Сценарий 1: Позитивный — Регистрация аккаунта Tesla

Цель: Пользователь хочет создать учетную запись на сайте Tesla для доступа к персонализированным функциям (например, управление заказом или настройка автомобиля).

Последовательность действий:

1. В правом верхнем углу нажать на иконку профиля
2. В открывшемся окне выбрать "Create Account" (Создать аккаунт).
3. Ввести данные для регистрации:
   - Регион – Other Europe.
   - Язык – English
   - First Name – Kostya
   - Last Name - Pyatkin
   - Email (например, "bavapew967@btcours.com").
   - Пароль (например, " Test1234!").
4. Нажать кнопку "Next"
5. Проверить email на наличие письма с подтверждением

6. Ввести код подтверждения из письма
7. Убедиться, что открылся личный кабинет

Ожидаемый результат: успешная регистрация пользователя

Листинг кода:

Так как при регистрации аккаунта требуется подтверждение по электронной почте, то тест останавливается на моменте ввода кода подтверждения. Так как Selenium не имеет такой возможности. Так же регистрация защищена капчей, что не дает автоматически завершить регистрацию.

```
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

class TestRegistration():
  def setup_method(self, method):
    self.driver = webdriver.Chrome()
    self.vars = {}

  def teardown_method(self, method):
    self.driver.quit()

  def test_registration(self):
    self.driver.get("https://www.tesla.com/")
    time.sleep(5)
    self.driver.set_window_size(734, 852)
```

```
    try:
        close_button = WebDriverWait(self.driver, 5).until(
            expected_conditions.presence_of_element_located(
                (By.CSS_SELECTOR, 'button[aria-label="Close Panel"].tds-icon-
btn')
            )
        )
        close_button.click()
    except:
        pass
    time.sleep(5)
    self.driver.find_element(By.CSS_SELECTOR, ".tds--highlighted").click()
    time.sleep(1)
    self.driver.find_element(By.XPATH, "//a[@id=\'dx-nav-item--
account\']/span").click()
    time.sleep(5)
    self.driver.find_element(By.ID, "form-button-create").click()
    time.sleep(3)
    self.driver.find_element(By.ID, "region-dropdown").click()
    time.sleep(2)
    self.driver.find_element(By.ID, "region-dropdown-US").click()
    time.sleep(3)
    self.driver.find_element(By.ID, "form-input-
first_name").send_keys("Kostya")
    self.driver.find_element(By.ID, "form-input-
last_name").send_keys("Pyatkin")
    self.driver.switch_to.frame(0)
    self.driver.find_element(By.ID, "checkbox").click()
    time.sleep(30)
```

Сценарий 2: Позитивный — Использование Charging Calculator

Цель: Пользователь хочет рассчитать стоимость зарядки автомобиля Tesla Model S с помощью калькулятора зарядки.

Последовательность действий:

1. В верхней части главной страницы найти раздел "Charging"
2. Нажать на ссылку "Charging Calculator"
3. Выбрать модель автомобиля (например, "Model S")
4. Указать параметры зарядки:
    - Временной промежуток (например, "Monthly")
    - Расстояние, проезжаемое в сутки (например, 150 миль)
5. Просмотреть результат (стоимость зарядки и экономия на топливе)

Ожидаемый результат: успешное отображение стоимости зарядки

Листинг кода:

```
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities

class TestChargingcalculator():
  def setup_method(self, method):
    self.driver = webdriver.Chrome()
    self.vars = {}

  def teardown_method(self, method):
    self.driver.quit()

  def test_chargingcalculator(self):
    self.driver.get("https://www.tesla.com/")
    self.driver.set_window_size(734, 852)
```

```
    time.sleep(3)
    try:
        close_button = WebDriverWait(self.driver, 5).until(
            expected_conditions.presence_of_element_located(
                (By.CSS_SELECTOR, 'button[aria-label="Close Panel"].tds-icon-
btn')
            )
        )
        close_button.click()
    except:
        pass

    time.sleep(3)

    self.driver.find_element(By.CSS_SELECTOR, ".tds--highlighted").click()
    time.sleep(3)

    self.driver.find_element(By.CSS_SELECTOR, "#dx-nav-item--
charging").click()
    time.sleep(3)

    self.driver.find_element(By.LINK_TEXT, "Charging Calculator").click()
    time.sleep(3)

    self.driver.find_element(By.CSS_SELECTOR, ".tds-form-input-
text").send_keys("1001")
    self.driver.find_element(By.ID, "monthly").click()
    time.sleep(10)
```

Сценарий 3: Негативный — Попытка заказа с некорректными данными оплаты

Цель: Пользователь пытается оформить заказ на Model S с использованием неверных данных карты.

Последовательность действий:

1. В верхнем меню выбрать "Vehicles" и кликнуть на "Model S"
2. Нажать "Order Now"
3. Настроить автомобиль (выбрать любые параметры: цвет, колеса, интерьер)
4. Нажать " Order with Card" для перехода к оплате
5. Ввести некорректные данные карты (например, номер карты: "1234-5678-9012-3456", срок действия: "01/25", CVV: "12")
6. Нажать "Place Order" для подтверждения оплаты
7. Дождаться реакции системы на ошибку

Ожидаемый результат: отображение сообщения об ошибке при покупке автомобиля

Листинг кода:

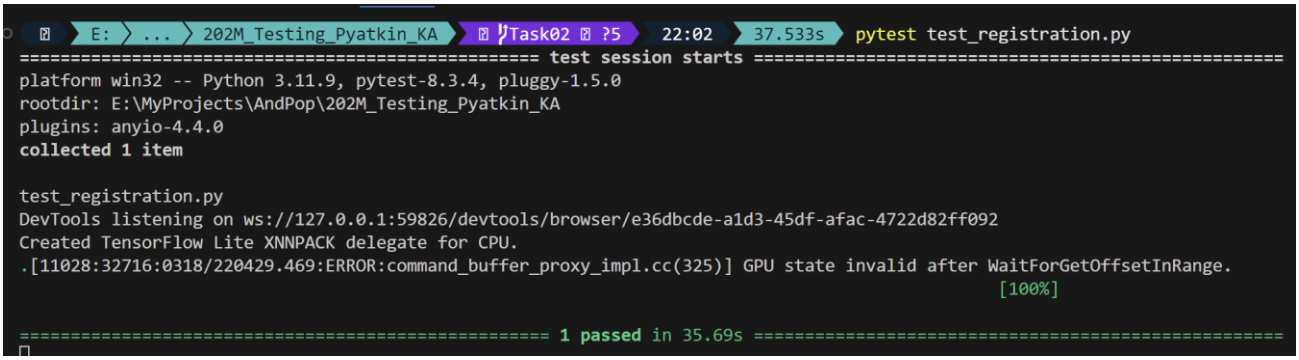
```
# Generated by Selenium IDE
import pytest
import time
import json
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.support import expected_conditions
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.desired_capabilities import DesiredCapabilities
from selenium.common.exceptions import NoSuchElementException

class TestBuecar():
  def setup_method(self, method):
    self.driver = webdriver.Chrome()
    self.vars = {}

  def teardown_method(self, method):
    self.driver.quit()

  def test_buecar(self):
```

```
    self.driver.get("https://www.tesla.com/")
    self.driver.set_window_size(734, 852)

    time.sleep(3)
    try:
        close_button = WebDriverWait(self.driver, 5).until(
            expected_conditions.presence_of_element_located(
                (By.CSS_SELECTOR,  'button[aria-label="Close  Panel"].tds-icon-
btn')
            )
        )
        close_button.click()
    except:
        pass

    time.sleep(3)

    self.driver.find_element(By.LINK_TEXT, "Order Now").click()
    time.sleep(5)
    self.driver.execute_script("window.scrollTo(0,1877)")
    self.driver.execute_script("window.scrollTo(0,2716.5)")
    self.driver.execute_script("window.scrollTo(0,2807.5)")
    time.sleep(2)

    self.driver.find_element(By.CSS_SELECTOR, ".tds-btn--secondary").click()
    self.driver.execute_script("window.scrollTo(0,3717.5)")

    time.sleep(5)
    WebDriverWait(self.driver, 10).until(
        expected_conditions.frame_to_be_available_and_switch_to_it(
            (By.CLASS_NAME, "payment-website")
        )
    )
    self.driver.find_element(By.NAME,
"/creditCardHolderName").send_keys("Kostya")
```

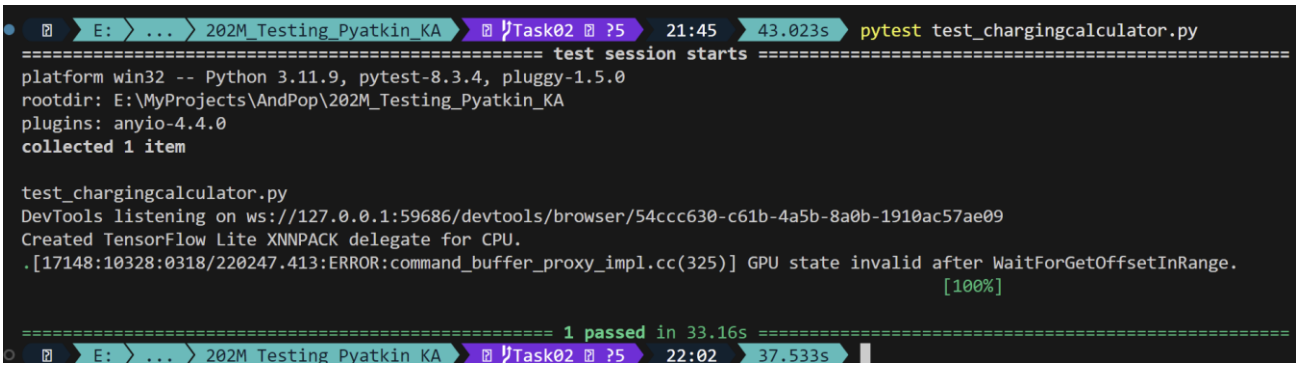
```
    self.driver.find_element(By.NAME,     "/creditCardNumber").send_keys("4117
7300 1045 2872")
    dropdown = self.driver.find_element(By.NAME, "/creditCardExpiryMonth")
    dropdown.find_element(By.XPATH, "//option[. = '01']").click()
    dropdown = self.driver.find_element(By.NAME, "/creditCardExpiryYear")

    dropdown.find_element(By.XPATH, "//option[. = '2025']").click()
    self.driver.find_element(By.NAME, "/creditCardCvv").send_keys("123")
    self.driver.find_element(By.NAME,     "/billingZipCode").send_keys("12342-
3232")
    self.driver.switch_to.default_content()
    self.driver.find_element(By.NAME, "postalCode").send_keys("19726")
    time.sleep(5)
    self.driver.find_element(By.ID, "FIRST_NAME").send_keys("Kostya")
    self.driver.find_element(By.ID, "LAST_NAME").send_keys("Pyatkin")
    self.driver.find_element(By.ID, "EMAIL").send_keys("aaa@aa.aa")
    self.driver.find_element(By.ID, "EMAIL_CONFIRM").send_keys("aaa@aa.aa")
    self.driver.find_element(By.ID,    "PHONE_NUMBER").send_keys("(201)    555-
0012")

    self.driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
    time.sleep(3)
    self.driver.find_element(By.CSS_SELECTOR, ".tds-btn--large").click()

    time.sleep(5)

    try:
        error_element    =    self.driver.find_element(By.XPATH,    "//*[@data-
id='error-alert']")
    except NoSuchElementException:
        raise   Exception("Элемент   с   data-id='error-alert'   не   найден   на
странице")
```

4. Результат автоматического тестирования

Регистрация аккаунта Tesla



Использование Charging Calculator



Попытка заказа с некорректными данными оплаты