**1. Паттерн Page Object (POM)**

Для организации тестов был применен паттерн Page Object. Вся логика взаимодействия с элементами веб-страниц вынесена в отдельные классы. Это делает тесты более читаемыми, поддерживаемыми и уменьшает дублирование кода.

Что показывать:

* Откройте структуру проекта в вашей IDE. Покажите папку pages/.
* Откройте файл pages/base\_page.py. Кратко объясните его назначение (общие методы для всех страниц: find\_element, click, type\_text, open, handle\_cookie\_consent, take\_screenshot).
* Откройте один из классов страниц, например, pages/home\_page.py. Покажите, как объявлены локаторы (LOGIN\_BUTTON\_HEADER, SEARCH\_INPUT) и методы, специфичные для этой страницы (click\_login\_button\_header, search\_for\_item). Укажите, что HomePage наследуется от BasePage.
* Откройте файл с тестами tests/test\_kufar\_pom.py. Покажите, как в тесте (test\_login\_success\_attempt или test\_search\_item) создаются экземпляры классов страниц и вызываются их методы, а не прямые команды Selenium для поиска элементов.

**2. Работа с опциями браузера**

Реализована возможность настройки опций браузера перед запуском тестов. Например, можно запустить тесты в headless-режиме или с другими параметрами.

Что показывать:

Откройте файл tests/conftest.py. Покажите фикстуру driver. Обратите внимание на использование ChromeOptions() или FirefoxOptions() и как в них добавляются опции (например, --headless, --start-maximized).

Покажите функцию pytest\_addoption, которая добавляет кастомные опции командной строки (--browser, --headless).

В терминале выполните команду (пример для Chrome):

**pytest tests/test\_kufar\_pom.py --browser chrome --headless --html=reports/demo\_headless\_report.html**

Объясните, что тесты выполняются без видимого окна браузера.

Затем выполните команду без headless (чтобы показать окно):

**pytest tests/test\_kufar\_pom.py --browser chrome --html=reports/demo\_visible\_report.html**

Во время выполнения будет видно окно браузера.

**3. Работа с куками**

Реализована работа с куками: автоматическое принятие согласия на использование куки при открытии страницы, а также специальный тест для демонстрации операций с куками (получение, добавление, удаление).

В **pages/base\_page.py** покажите метод**handle\_cookie\_consent()**и как он вызывается в методе open().

В**tests/test\_kufar\_pom.py** откройте тест**test\_cookie\_operations.**

В терминале запустите этот тест (флаг -s для вывода print в консоль):

**pytest -k test\_cookie\_operations tests/test\_kufar\_pom.py -s**

Покажите скриншот, который был сделан в этом тесте (если он есть в коде), например screenshots/after\_cookie\_manipulation...png.

4. Запись скриншота

Реализована функция создания скриншотов. Скриншоты делаются как принудительно в определенных точках тестов, так и автоматически при падении любого теста."

В **pages/base\_page.py** покажите метод take\_screenshot().

В tests/conftest.py покажите хук**pytest\_runtest\_makereport**, который отвечает за автоматическое создание скриншота при падении теста.

**pytest tests/test\_kufar\_pom.py::test\_login\_success\_attempt --html=reports/demo\_failed\_report.html --self-contained-html**

Откройте сгенерированный HTML-отчет (reports/demo\_failed\_report.html). Покажите, что в отчете для упавшего теста есть ссылка на скриншот или он встроен.

Не забудьте вернуть тест в рабочее состояние.

5. Параметризация тестов

Что говорить: "Для проверки функциональности с различными наборами входных данных используется параметризация тестов."

В tests**/test\_kufar\_pom.py** откройте тест test\_search\_item. Покажите декоратор**@pytest.mark.parametrize**. Объясните, что тест будет запущен несколько раз с разными значениями search\_term и expected\_header\_part.

В терминале запустите этот параметризованный тест:

pytest -k test\_search\_item tests/test\_kufar\_pom.py -s

6. Управление тестами

Для гибкого управления тестовым набором используются различные механизмы Pytest: метки (маркеры), пропуск тестов, отметка ожидаемо падающих тестов, выборочный запуск и управление порядком выполнения.

Метки (теги): В tests/test\_kufar\_pom.py покажите маркеры над тестами, например, @pytest.mark.login, @pytest.mark.smoke, @pytest.mark.search.  
В терминале запустите тесты по метке:

Покажите, что запустились только тесты с меткой smoke.

Пропуск тестов: Покажите тест test\_skipped\_example с декоратором @pytest.mark.skip(reason="...").  
Запустите все тесты (или только этот):

pytest tests/test\_kufar\_pom.py

В выводе Pytest будет видно, что этот тест пропущен (skipped).

Ожидаемо падающие тесты (XFail): В параметризованном тесте test\_search\_item покажите pytest.param(..., marks=pytest.mark.xfail(...)).  
Запустите этот тест (или все):

pytest -k test\_search\_item tests/test\_kufar\_pom.py

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "https://aistudio.google.com/prompts/_blank).Bash

В выводе для этого параметра будет статус XFAIL или XPASS.

Выборочный запуск: "Тесты можно запускать по имени."  
В терминале запустите один тест по его полному имени:

pytest tests/test\_kufar\_pom.py::test\_login\_success\_attempt

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "https://aistudio.google.com/prompts/_blank).Bash

Порядок запуска: "Для управления порядком выполнения тестов используется плагин pytest-ordering и маркер @pytest.mark.run(order=X)."  
Покажите использование @pytest.mark.run(order=...) над несколькими тестами в test\_kufar\_pom.py.  
Запустите все тесты с флагом -s или -v:

pytest tests/test\_kufar\_pom.py -s

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "https://aistudio.google.com/prompts/_blank).Bash

Обратите внимание на порядок вывода названий тестов в консоли – он должен соответствовать указанным order.

7. Отчет о результатах тестирования

Что говорить: "После выполнения тестов генерируется подробный HTML-отчет с результатами, используя плагин pytest-html."

Что показывать:

В терминале выполните команду для запуска всех тестов и генерации отчета: