Задание для стажеров на позицию QA Auto/SDET

Требования к кандидату: Это задание предназначено для оценки ваших навыков работы с информацией, способности к исследованию, умения работать с документацией, быстрого понимания поставленных задач и базовых навыков программирования.

Важно: Мы не предоставляем рекомендаций по написанию кода. Полагайтесь на свои профессиональные навыки. Также не предоставляются ссылки на предложенные инструменты. В реальной жизни навык работы с информацией имеет ключевое значение.

На выполнение задания отводится 2-4 дня. Задание может быть сдано на проверку при выполнении 3 из 4 всех заданий. После выполнения задания состоится итоговая беседа с руководителем.

Общие требования

- Каждое задание должно быть размещено в отдельной папке.
- Добавьте файл `README.md` для описания кейса, инструкции по запуску или предоставления любой полезной информации.
- Пожалуйста, добавьте файл с зависимостями для запуска, если это необходимо.
- Итоговый набор заданий разместите на своем GitHub. Если это невозможно, используйте облачный диск и предоставьте ссылку.
- Задания, отмеченные звездочкой (/*), выполняются по желанию кандидата.

Задания

Задание 1: Исследовательское тестирование

1. Посетите сайт mts.ru и проведите мини-исследовательское тестирование.
2. Опишите один тест-кейс, состоящий из 6-10 шагов, который должен включать взаимодействие со страницей и как минимум одну проверку (assert).

/* Опишите два тест-кейса: позитивный и негативный сценарии.

Задание 2: Автоматизация тест-кейса

- На основе вашего тест-кейса создайте его автоматизацию.
- Используйте фреймворки Playwright и Pytest для создания UI-теста.
- Опишите комментариями каждый шаг.

/* Для описания шагов используйте фреймворк Allure с его step-обертками.

Критерий оценки: Я должен иметь возможность запустить ваш тест локально. Тест должен открыть браузер, провести тестирование (успешно или неуспешно) и закрыться.

Задание 3: Написание простого сервера

Напишите простой сервер, который может обрабатывать два маршрута:

1. **/inverse** (метод POST): принимает JSON, меняет местами ключи и их значения, и возвращает ответ.

   Пример:

   - Запрос:

     ```
     {"key1": "value1"}
     ```

   - Ответ:

     ```
     {"value1": "key1"}
     ```

   - Статус: 200

2. **/unstable** (метод GET): с рандомной вероятностью возвращает:

   - Код 200 и в ответе `HAPPY`
   - Код 400 и в теле `UNHAPPY`

Решение может быть реализовано с использованием любого фреймворка. Скрипт должен запускаться одним файлом.

/* Реализуйте для вашего сервера Swagger. Интерфейс должен открываться при запуске вашего сервера, и с него должны корректно отправляться запросы на ваш сервер.

/* Напишите юнит-тесты для вашего сервера, используя Pytest.

Задание 4: Нагрузочное тестирование

- Используя сервер из задания 3, напишите скрипт нагрузки.
- Используйте инструменты Locust или K6.
- Запустите на небольшом количестве пользователей для отладки, чтобы не перегрузить вашу машину.
- Опишите ваши наблюдения: количество RPS, время отклика.

/* Напишите скрипт sh/bat, который запустит сервер из задания 3, а затем запустит нагрузочный скрипт.

/* При запуске скрипта проведите мониторинг вашей системы на использование CPU и RAM, добавьте это в описание.