Тестовое задание для Junior QA Engineer

Формат сдачи: один файл `assignment.md` (Markdown) + (по желанию) Postman-коллекция.
Ориентир по времени: до 3–4 часов.
Цель: проверить умение понять требования, спланировать тестирование, написать понятные тест-кейсы, проверить базовое API, оформить баг-репорты и зафиксировать вопросы/допущения.

---

1) Объекты тестирования

A. Веб-UI (демо интернет-магазин)

- Сайт: https://www.saucedemo.com/
- Доступ:
  - **username:** `standard_user`
  - **password:** `secret_sauce`
- Что считается «скоупом»: логин/логаут, список товаров (в т.ч. сортировка), карточки, корзина, оформление заказа (happy-path + базовые валидации форм).
- **Ограничения:** тестируем на **desktop Chrome**. Мобильная/кросс-браузерность **вне зоны**.

B. Учебное API (e-commerce)

- Сервис: https://dummyjson.com/ (публичное демо API).
- Рекомендуемые участки: аутентификация (`/auth/login` , `/auth/me`), товары (`/products` , поиск, категории), корзины (`/carts` , базовые операции).
- **Важно:** это учебный сервис — мутации имитируются; это нужно учесть в ожидаемых результатах.

---

2) Что нужно сделать и сдать

Соберите **один Markdown-файл** `assignment.md` со следующими разделами (по порядку):

2.1. Короткий тест-план (до 1 страницы)

Опишите:

- Цели и **границы** тестирования (что в зоне/вне зоны).
- Окружение (браузер/версия, ОС).
- Подходы: позитив/негатив, граничные значения, приоритизация.
- Что пойдёт в **smoke** при приёмке.
- Критерии завершения тестирования (это условия, при которых тестировщик считает работу завершённой и может остановить тестирование.).

2.2. Тест-кейсы по UI (минимум 12, максимум 20)

Таблица в Markdown. Покройте как минимум:

- Логин/логаут (успех/ошибка).
- Добавление/удаление товара, консистентность счётчика корзины.
- Сортировка списка по цене/алфавиту (любой один вариант обязателен).
- Оформление заказа: один **happy-path** + 2–3 **негативных** проверки валидаций.
- Навигация/обновление страницы (сохранение состояния).

Шаблон строки (используйте его, но не заполняйте за рамки вашего решения):

```
| ID | Название | Предусловия | Шаги | Ожидаемый результат | Тест-данные | Приоритет |
```

2.3. Тест-кейсы по API (минимум 8, максимум 12)

Таблица в Markdown. Обязательно включите:

- Аутентификация: успех и минимум 2 негативных сценария (например, пустые поля/неверные данные/отсутствие токена).
- Товары: список с ограничением `limit/skip` и поиск `q`.
- Категории: получение списка и выбор одной категории.
- Корзины: базовая операция (получение корзины пользователя или имитация добавления).
- Проверяйте **коды ответов**, **ключевые поля/типы**, адекватность **сообщения об ошибках**.

Шаблон строки:

```
| ID | Endpoint | Метод | Тело/Параметры | Ожидаемый код | Ожидаемое тело (ключевые поля) | Примечания |
```

2.4. Чек-лист приёмки (8–10 пунктов)

Короткий список того, что обязательно проверите как **smoke** перед релизом (UI + API).

2.5. Баг-репорты (минимум 3)

Оформите реальные найденные проблемы **или** обоснованные UX/валид. замечания.
Шаблон:

- Заголовок, Окружение, Шаги, Фактический результат, Ожидаемый результат, Severity/Priority, Примечания (скриншот по желанию).

2.6. Вопросы и допущения (5–8 пунктов)

Что осталось неясным, какие решения приняли сами (например, про локаль, формат дат, требования к доступности).

(Опционально) 2.7. Коллекция Postman

Если удобно — приложите JSON-коллекцию (с тест-скриптами/assert’ами) и кратко опишите переменные.

---

3) Ограничения и правила

- Не нужно автоматизировать тесты — **только ручное тестирование** и артефакты.
- Не нужно исчерпывающее покрытие всех комбинаций — **работайте по рискам**.
- Достаточно одного браузера (Chrome desktop).
- Если что-то в демо-API «симулируется» — это не баг, но **пропишите ожидания корректно**.

---

4) Критерии оценки

Критерий	Что смотрим
Понимание границ	Чётко отделены «в зоне» и «вне зоны»
Полнота покрытия	UI ≥12 кейсов с балансом позитив/негатив; API ≥8 кейсов с кодами и ключевыми полями
Качество формулировок	Шаги проверяемы, ожидаемый результат конкретен, понятная приоритизация

Критерий	Что смотрим
Интеграционное мышление	Последовательность действий (UI ↔ состояние), реалистичные ожидания к API
Баг-репорты	Конкретные, воспроизводимые, с корректными ожиданиями
Коммуникация	Вопросы/допущения, умение «договориться о требованиях»
Акуратность	Структура файла, лаконичность, грамотность

---

5) Подсказки

- Для API заранее определите **минимальный набор проверок схемы** (например, наличие `total`, `products`, `id`, `title` и т.п.).
- Негативные сценарии обычно самые ценные: пустые обязательные поля, неправильные типы, отсутствие авторизации.

---

6) Как сдавать

- Отправьте `assignment.md` (и, при наличии, Postman-коллекцию `.json`).
- В начале файла укажите ФИО и время, затраченное на выполнение (оценочно).