Домашнее задание к вебинару №23 «Unit-тестирование»

Вводные данные:

Перед Вами поставлена задача реализовать бэк для страницы оплаты просмотра фильма через некий платёжный сервис **А**. Рассмотрим упрощённую схему, подразумевающую, что сервис сразу отвечает нам о возможности списания средств, без дополнительной авторизации через 3D-Secure.

Фронт для этой страницы уже реализован, он передаёт JSON-объект, содержащий текстовые поля *card_number* (номер карты, 16 цифр, считаем, что валидный номер всегда содержит 16 цифр, валидность номера по алгоритму Луна проверять не нужно), *card_holder* (владелец карты, имя и фамилия латиницей, может также содержать дефис), *card_expiration* (месяц/год окончания действия карты в формате мм/гг), *cvv* (код с обратной стороны карты, 3 цифры), *order_number*(номер заказа, до 16 произвольных символов), *sum* (сумма оплаты, разделитель дробной и целой части **запятая**, поэтому и строка, а не число).

Бэк состоит из одного метода контроллера и выполняет следующие действия:

- Валидирует данные, если в данных есть ошибка, то возвращает сообщение об ошибке с кодом 400;
- Если данные верные, то передаёт их в API-запросе на сервис **А**. Сервис **А** пытается списать деньги, если ему это не удаётся, то он возвращает HTTP-код 403, если удаётся, то HTTP-код 200;
- В случае ошибки передаём её обратно на фронт.
- В случае успешного списания денег необходимо записать в БД информацию об успешной оплате. Предполагаем, что у нас есть соответствующий метод репозитория setOrderIsPaid(string $orderNumber, float $sum): bool, реализованный ранее. Метод проверяет соответствие номера заказа и его суммы и возвращает true, если списание успешно. В случае ошибок выбрасываются различные исключения.

Задача:

Предлагается описать кейсы тестирования данной задачи для трёх уровней: модульного, интеграционного и системного. Считаем, что взаимодействуют 4 «модуля»: бэк, фронт, репозиторий и сервис **А**.

Кейсы тестирования предлагается описывать примерно так:

1. Модульные тесты:

    a. Если *card_holder* содержит более одного пробела, то тестируемый метод возвращает 400 с сообщением об ошибке;
    b. …
2. Интеграционные тесты:
    a. Проверяем связку «фронт-бэк»:
        i. Если *card_holder* содержит более одного пробела, то после получения ответа от бэка на фронте выделяется поле «Номер карты» красной рамкой;
        ii. …
    b. Проверяем связку «бэк-репозиторий»:
        i. …
3. Системные тесты:
    a. Если *cvv* неверен, то после получения ответа от бэка на фронте выводится сообщение об ошибке «Не удалось списать данные с карты, проверьте реквизиты».

Поведение фронта в случае ошибок можете выбрать произвольное, но консистентное, т.е. при ошибках одинакового уровня поведение должно быть одинаковым. Пример неконсистентного поведения:

- при ошибке в *card_number* выделяется поле «Номер карты» красной рамкой
- при ошибке в формате *card_expiration* закрывается браузер с ошибкой Out of memory.

Задача будет зачтена, если будут выполнены следующие критерии:

1. Модульные кейсы покрывают все возможные примеры невалидных данных;
2. Интеграционные и системные кейсы правильно разделены по уровням, т.е. нет системных кейсов внутри раздела интеграционных и наоборот;
3. Есть хотя бы по два интеграционных кейса для каждой связки и хотя бы два системных;
4. Предложенные кейсы гарантируют, что в итоге пользователь сможет воспользоваться реализованной страницей, и это учтено на каждом уровне (это некая подсказка, более подробно могу сформулировать только после первой попытки сдачи работы);

Дополнительные замечания:

1. Необязательно переносить на уровень выше **все** тесты, но желательно перенести хотя бы те, которые проверяют сильно отличающееся поведение;

2. Следуем правилу «не нужно проверять чужой код»: если то, что наш модуль передаёт другому, должно приводить к схожим результатам, то достаточно одного теста;
3. Поскольку мы тестируем оплату, то стоимость тестирования в некоторых ситуациях будет выражаться в реальных денежных затратах, а не в виртуальных. Стоит обратить на это внимание и, возможно, предложить способы удешевления тестирования.