Письменные ответы на вопросы

Вопрос 1

Для решения данной задачи я бы использовала технику парного тестирования (pairwise testing) . Техника парного тестирования основана на идее создания тест-кейсов, которые покрывают все возможные комбинации значений двух и более параметров, что позволяет достичь хорошего покрытия тестирования при минимальном количестве тест-кейсов. Применив ресурс https://inductive.no/pairwiser-tool/, можно разработать 109 тест-кейсов. Количество возможных тестов составит 360, что означает, что есть ещё 251 возможная комбинация, которая не используется в тестах. Однако, разработанные тесты обеспечат 100% покрытие всех возможных комбинаций из трех значений, покрыв 446 комбинаций, что является полным покрытием всех этих комбинаций. Ни одна комбинация не останется непокрытой.

Вопрос 2

Тест-кейсы принято оформлять по определённому стандарту, каждый тест-кейс должен состоять из нескольких чётких элементов — атрибутов, так же тест-кейсы обычно пишутся от третьего лица, в форме инструкций для тестировщика.
Поэтому я бы добавила идентификатор тест-кейса “**Тест-кейс №:** ID-001” и изменила шаги воспроизведения написав: **Перейти по ссылке** *https://practicum.yandex.ru/* **на главную страницу Яндекс Практикума.**

Вопрос 3

Шаги воспроизведения:
1. Открыть форму создания заказа.
2. Открыть инструмент разработчика Devtools нажав на кнопку F12
3. Перейти во вкладку Network (Сеть)
4. Заполнить форму, простовить необходимые галочки в чек-боксах, создать заказ.
5. Посмотреть в Devtools отправленные HTTP-запросы и выбрать нужный нам запрос (обычно это POST-запрос)
6. Просмотрите данные, отправленные в этом запросе, и убедитесь, что передаваемое значение чек-бокса соответствует ожидаемому (true).
7. Если передаваемое значение верное, просмотрите логи и код бэкенда, чтобы убедиться, что обновление базы данных выполняется корректно.
8. Если проблема не найдена на стороне бэкенда, проверьте, что данные успешно записываются в базу данных.

Вопрос 4

Отправленный GET-запрос сам по себе не должен вызывать случайное удаление всей информации в базе данных. GET-запросы, в соответствии со спецификацией HTTP, являются безопасными и не должны изменять состояние сервера или базы данных. Они предназначены только для получения данных. Однако, если приложение не правильно обрабатывает и не проверяет пользовательский ввод перед его использованием в GET-запросах, возникает риск SQL-инъекций. В этом случае, злоумышленник может использовать уязвимость для внесения вредоносного SQL-кода в запросы к базе данных, в следствии чего могут наступить потенциальные негативные последствия для базы данных.

Вопрос 5

Таблица сотрудников **employee** с полями:

- id — идентификатор сотрудника, # можно сделать первичным ключом, поле уникально и помогает идентифицировать каждого сотрудника в таблице.
- fio — ФИО сотрудника,
- position_id — идентификатор должности. # можно сделать внешним ключом ссылающимся на поле id в таблице position. Это позволит установить связь между таблицами, где каждому сотруднику будет соответствовать только существующая должность в таблице position.

Таблица должностей **position** с полями:

- id — идентификатор должности, # можно сделать первичным ключом, поле уникально и помогает идентифицировать каждую должность в таблице.
- name — название должности,
- salary — зарплата на данной должности.

Вопрос 6

```
SELECT employee.fio, position.name, position.salary
FROM employee
JOIN position ON employee.position_id = position.id;
```

Вопрос 7

В данном коде отсутствует проверка результата сравнения a == 2. Для проверки необходимо использовать оператор сравнения assert, который проверяет, является ли выражение истинным. Правильный код будет выглядеть так:

```
def test_integer_division():
    a = 5 // 2
    assert a == 2
```

test_integer_division()

Вопрос 8

Классы эквивалентности и граничные значения существуют отдельно, когда классы эквивалентности не покрывают все возможные случаи входных данных. Классы эквивалентности определяют различные группы входных данных, а граничные значения учитывают крайние случаи.
Пример: для функции, определяющей тип треугольника по его сторонам, классы эквивалентности могут быть "равносторонний", "равнобедренный" и "разносторонний". Граничные значения включают случаи с нулевыми сторонами и крайними значениями сторон (минимальными и максимальными).

Вопрос 9

Я считаю, что нельзя исключить проверку в середине диапазона в пользу проверок на границах. Проверка в середине диапазона важна для обнаружения потенциальных проблем или ошибок, которые могут возникнуть внутри диапазона значений. Она может выявить проблемы, которые не проявляются на границах.

Вопрос 10

Для воспроизведения бага и получения логов в мобильной версии Яндекс Аренды, необходимо сделать следующее:

1. Убедиться, что на мобильном устройстве включены настройки разработчика и отладка по USB. Это позволит подключить устройство к компьютеру и получить доступ к логам.
2. Подключить мобильное устройство к компьютеру с помощью USB-кабеля.
3. Открыть командную строку (на Windows) или терминал (на macOS или Linux) на компьютере.
4. Ввести команду, чтобы проверить, видит ли компьютер мобильное устройство: adb devices
5. Запустить приложение на устройстве.
6. Выполнить команду adb logcat в командной строке, чтобы начать запись логов приложения.
7. Перейти в раздел "Показать на карте" в приложении и вызывать ошибку.
8. После возникновения ошибки вернуться к командной строке и нажмите Ctrl + C, чтобы остановить запись логов.

Логи приложения будут выведены в командной строке или терминале. Можно сделать скриншот логов выведенных в терминале или сохранить логи в файл, командой adb

logcat -d > logs.txt. Полученные данные необходимо отправить разработчику для дальнейшего анализа.

Отчёт о тестировании

Функциональное тестирование веб-приложения

Приложение проверено на стенде https://4805d8bc-6170-4c28-952c-966390ae7843.serverhub.praktikum-services.ru/.

Все известные требования были покрыты чек-листом: https://docs.google.com/spreadsheets/d/1_Pqq_qny2dHvJDXwBo9Fw6aXF-4pXN4B/edit?usp=sharing&ouid=105663013887289533919&rtpof=true&sd=true.

Результаты выполнения тестов можно посмотреть здесь: https://docs.google.com/spreadsheets/d/1_Pqq_qny2dHvJDXwBo9Fw6aXF-4pXN4B/edit?usp=sharing&ouid=105663013887289533919&rtpof=true&sd=true.

Из **324** успешно прошло **286**, не прошло — **29** и пропущено **9**.

Список багов, найденных при тестировании, разбит по приоритетам:

1. Блокирующие: 1
BUGS-29
2. Критичные: 13
BUGS-6
BUGS-9
BUGS-11
BUGS-12
BUGS-14
BUGS-16
BUGS-18
BUGS-19
BUGS-20
BUGS-22
BUGS-23
BUGS-24
BUGS-28
3. Средний приоритет: 8
BUGS-2
BUGS-3
BUGS-5
BUGS-8
BUGS-17
BUGS-25

BUGS-26
BUGS-27
4. Низкий приоритет: 6
BUGS-1
BUGS-4
BUGS-10
BUGS-13
BUGS-15
BUGS-21
5. Незначительный приоритет: 1
BUGS-7

Заключение:

У приложения есть несколько критически важных проблем, включая блокирующий баг, который мешает пользователям оформить заказ. Однако, помимо этого бага, приложение также содержит несколько других критических проблем, таких как возможность оформления заказа без заполнения всех обязательных полей или отсутствие отображения ошибок под незаполненными или некорректно заполненными полями. Это может ввести пользователей в заблуждение и испортить их общее впечатление от использования приложения.

В требованиях отсутствуют указания относительно всплывающего окна "Хотите оформить заказ", которое должно появляться при нажатии на кнопку "Заказать" на экране "Про аренду". Также требования не содержат информации о кнопке "Go!" и ее предполагаемом функционале. Более того, поле "Фамилия" на экране "Для кого самокат" не может принимать значения, содержащие тире и пробелы, в то время как поле "Имя" не имеет такого ограничения. Наличие и функции кнопка “Заказать” в шапке форм “ Для кого самокат” и “Про аренду”

Проверенная мной функциональность не готова к релизу. В приложении имеется блокирующий баг и начительное количество критических багов, что может испортить пользователю впечатления при работе с приложением, испортить репутацию компании и отпугнуть потенциальных клиентов.

Ретест багов в мобильном приложении

Был проверен фикс багов. Из них не исправлено 1, исправлено — 3.

Список багов можно посмотреть здесь: retest_samokat-2

Регрессионное тестирование мобильного приложения по готовым тест-кейсам

Результаты выполнения регрессионных тестов можно посмотреть здесь: https://docs.google.com/spreadsheets/d/1b53bcv8xqedzRpxmSNMtMinpgECzxWy2/edit?usp=sharing&ouid=105663013887289533919&rtpof=true&sd=true.

Из 10 успешно прошло 1, не прошло — 9.

Список багов, найденных при тестировании, разбит по приоритетам:

1. Блокирующие: 0

2. Критичные: 6

BUGS-MOBILE-1
BUGS-MOBILE-2
BUGS-MOBILE-3
BUGS-MOBILE-4
BUGS-MOBILE-7
BUGS-MOBILE-8

4. Обычный приоритет: 1
BUGS-MOBILE-9

5. Низкий приоритет: 2
BUGS-MOBILE-5
BUGS-MOBILE-6

Заключение:

Мобильное приложение имеет значительное количество критических багов, которые серьезно влияют на его функциональность. Проблемы включают поломки при принятии заказа, если он отменяется или принимается другим курьером, неполное отображение адреса и станций метро, а также отсутствие отображения комментариев пользователей. Кроме того, обнаружено дублирование карточек заказа, что затрудняет работу в приложении.

На основании вышеуказанных проблем, можно сделать вывод, что такой продукт нельзя выпускать в релиз. Все найденные баги были классифицированы как критические, и их исправление является неотложной задачей для обеспечения стабильности и надежности работы приложения.
Рекомендуется внимательно и тщательно устранить все критические проблемы перед выпуском приложения в релиз. Только после исправления всех критических багов, приложение будет готово для успешного запуска и использования широкой аудиторией.