Письменные ответы на вопросы

*помести сюда ответы на 10 вопросов из теоретического блока

Ответ на вопрос 1.

Для составления тест-кейсов я использовала бы такую технику тест-дизайна, как “Попарное тестирование”.

Это связано с тем, что в условии указано большое количество комбинаций, которые необходимо проверить. Если тестировать все возможные комбинации, то это займет большое количество времени и ресурсов, в чем компании, как правило, ограничены. В учебнике Ли Копланда указана гипотеза, почему попарное тестирование хорошо работает - она заключается в том, что большинство дефектов являются либо одиночными (тестируемая функция просто не работает и любой тест на эту функцию найдет дефект), либо двойными (это пара из функции/модуля, с которыми функция/модуль проваливаются, хотя все остальные пары работают успешно). Поэтому при помощи попарного тестирования можно определить минимальный набор данных, который позволяет проверить все одиночные и попарные дефекты.

Ответ на вопрос 2.

1. Нужно изменить заголовок. По заголовку, указанному в задании, непонятно, в чем смысл тест-кейса.
2. Я бы изменила стилистику написания в соответствии с орфографическими нормами - название системы я бы указала в кавычках “Яндекс Практикум”, и в именительном падеже.
3. Шаги воспроизведения должны носить обезличенный характер и глагол должен стоять в инфинитиве - “Открыть главную страницу”.
4. Исходя из заголовка, тест-кейс направлен на отображение главной страницы - соответственно, акцент сделан на какие-либо элементы, которые должны на ней присутствовать. В таком случае, нужно изменить ожидаемый результат - указать, что именно отображается на главной странице, а не то, что она открылась.

Ответ на вопрос 3.

Чтобы локализовать ошибку в Devtools нужно выполнить следующее:

1. Открыть форму создания заказа (предварительно удалив все ранее отправленные запросы и полученные ответы во вкладке “Network”)
2. Проставить чек-бокс “Заплатить сразу” и посмотреть, какой запрос ушел на сервер - должно отправляться значение “true” (если валидация происходит на стороне клиента)

2.1. Проставить чек-бокс “Заплатить сразу” и отправить форму и посмотреть, какой запрос ушел на сервер - должно отправляться значение “true” (если валидация происходит на стороне сервера)

3. Посмотреть ответ бэкенда на отправленный запрос - какой статус-код приходит и что отображается во вкладке “Response”. Возможно, ошибка происходит в процессе взаимодействия фронтенда и бэкенда - либо фронтенд неправильно передает данные на бэкенд, либо бэкенд неправильно передает данные в базу данных
4. Открыть вкладку “Console”, чтобы проверить, какие ошибки могли возникнуть и где - на стороне фронтенда или на стороне бекэнда, для локализации ошибки
5. Просмотреть логи ошибок, отображающихся во вкладке “Console”
6. Воспользоваться СУБД, отправив запрос в базу данных, для просмотра данных, отображающихся в таблице

Ответ на вопрос 4.

В спецификации HTTP метод GET означает получение любой информации. При отправке GET-запроса сервер выдает пользователю запрашиваемую информацию. Согласно акроним CRUD, GET-запрос позволяет пользователю получать данные только для чтения. Кроме того, метод GET не имеет тела для передачи данных о том, что удалить в базе данных.

Тем не менее, некорректная работа сервера может повлиять на неправильную обработку запроса, в связи с чем данные в базе могут удалиться - но это крайне маловероятно. Кроме того, в Postman предусмотрена возможность кастомизировать собственные методы - создавать свои. Возможно, есть вероятность того, что можно создать свой метод с названием “GET”, который будет в себе нести функцию метода “DELETE”, однако, это тоже маловероятно.

При корректном использовании метода и нормального функционирования сервера метод GET не может удалить всю информацию из базы данных.

Ответ на вопрос 5.

Первичный ключ - однозначный идентификатор, который является обязательным в любой таблице реляционной базы данных, так как он уникальный и определяет конкретные строки в таблице. Вторичный (внешний) ключ служит для того, чтобы была возможность связать две или больше таблиц между собой, он является необязательным.

Таким образом, первичным ключом будет являться поле “id” в таблице “employee”, внешним ключом будет являться поле “position_id” в таблице в таблице “employee”.

Соответственно, employee.position_id = position.id.

При этом, в таблице “position” первичным ключом будет служить поле “id”.

Ответ на вопрос 6.

SELECT e.fio, p.name, p.salary2

FROM employee AS e

INNER JOIN position AS p ON e.position_id = p.id

Ответ на вопрос 7.

Код должен проверять, что оператор // действительно выполняет деление на целое число.

Но при этом в коде отсутствует выражение, которое проверяло бы, выполняется ли деление - assert, которое работает как логическое выражение и возвращает True или False.

Для деления в Python есть оператор //, при использовании которого дробная часть отсекается - целочислительное деление. Если 5 разделить на 2 получится 2.5, а без остатка - соответственно, 2.

Таким образом, третья строка кода будет выглядеть так: assert a == 2 (вместо a == 2), то есть мы проверяем, что деление при помощи оператора // на целое число выполняется, и выполняется корректно.

Ответ на вопрос 8.

Класс эквивалентности - это входные данные, которые обрабатываются приложением одинаково, либо обработка которых приводит к одному и тому же результату. Граничные значения - это данные, входящие внутрь классов эквивалентности, на которые приложение может специфически реагировать.

Например, поле ввода принимает символы в количестве от 2 до 5 - мы выделяем классы эквивалентности “-∞, 1”, “2-5”, “6 - +∞”, на границах классов есть значения 0, 1, 2, 3, 4, 5, 6, 7 - в данному случае КЭ и ГЗ связаны между собой.

Но тем не менее классы эквивалентности и граничные значения могут существовать по отдельности. Так - для этого же поля можно выделить отдельные классы, такие как: русские буквы, английские буквы, спецсимволы, иероглифы и так далее - это отдельные классы эквивалентности, которые существуют отдельно от граничных значений, которые также можно разделить на валидные и невалидные значения.

Ответ на вопрос 9.

Как я писала выше - проверки на границах проводятся потому, что именно на границах могут сосредоточиться баги и вызывать специфическую реакцию системы, и чтобы это исключить проводится тестирование как внутри диапазонов, так и на их границах.

При тестировании граничных значений так или иначе затрагивается проверка значений, входящих внутрь диапазона (класса), поэтому вполне возможно исключить отдельную проверку в середине диапазона. Однако, для того, чтобы более качественно провести тестирование, лучше не исключать проверку в середине диапазона. В случае, если

диапазон небольшой, я думаю, можно воздержаться от проверки в середине диапазона, однако если он широкий - тогда лучше проверять в том числе реакцию приложения на значение, которое находится в середине диапазона.

Ответ на вопрос 10.

1. Воспроизвести баг:

   а) Открыть приложение “Яндекс Аренда” на мобильном устройстве;

   б) Перейти в раздел “Показать на карте”;

   в) Записать время и дату, когда воспроизведен баг

   Ожидаемый результат: Появляется сообщение “Непредвиденная ошибка”, и приложение закрывается.

2.1. Для устройства на платформе Android (предусловие - установлен “Android Studio”):

а) Перевести устройство в режим разработчика/отладки (в настройках мобильного устройства);

б) Подключить мобильное устройство при помощи USB к компьютеру;

в) Открыть Android Studio;

г) Перейти во вкладку Logcat;

д) Выбрать мобильное устройство, с которого снимаем логи;

е) Выбрать приложение, с которого снимаем логи - “Яндекс Аренда”;

Ожидаемый результат: в окне отображаются все логи, снятые с приложения “Яндекс Аренда” (так как при воспроизведении бага мы записали время и дату, можно по этим данным найти интересующие нас логи);

ж) Скопировать интересующие логи и отправить разработчику.

2.2. Для устройства на платформе iOS (предусловие - установлен “xCode”):

а) Подключить устройство при помощи USB к компьютеру;

б) Открыть xCode → Windows → Devices and Simulators;

в) Нажать на кнопку “Open Console”.

В данном случае логи поступают в реальном времени, поэтому воспроизвести баг (пункт 1) необходимо повторно с подключенным к компьютеру устройством, чтобы логи отобразились в консоли.

г) Воспроизвести баг на устройстве;

д) Скопировать интересующие логи и отправить разработчику.

Отчёт о тестировании

*заполни отчёт по образцу ниже

Функциональное тестирование веб-приложения

Приложение проверено на стенде:

https://29ff9df0-7e61-4171-8225-adbaeb03e8e1.serverhub.praktikum-services.ru

https://4a550070-5ab0-44ac-89f4-d0bdd977ce9d.serverhub.praktikum-services.ru

(задание, связанное с тестированием веб-приложения, заняло два дня, в связи с чем тестирование проводилось на двух стендах)

Все известные требования были покрыты чек-листом:

https://docs.google.com/spreadsheets/d/1ne90x27o2F9SFpV75FQvPB6kYkW4gCkgC0ADx2mrpKE/edit?usp=sharing

Результаты выполнения тестов можно посмотреть здесь:

https://docs.google.com/spreadsheets/d/1ne90x27o2F9SFpV75FQvPB6kYkW4gCkgC0ADx2mrpKE/edit?usp=sharing

Из **176** тестов успешно прошло **140**, не прошло — **36.**

Список багов, найденных при тестировании, разбит по приоритетам:

1. Блокирующие:

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-26

2. Критичные:

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-2

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-4

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-5

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-12

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-13

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-14

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-15

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-16

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-23

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-24

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-25

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-27

3. Средний приоритет:

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-3

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-7

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-8

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-19

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-20

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-21

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-22

4. Низкий приоритет:

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-1

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-6

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-9

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-10

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-11

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-17

https://berkut-ekaterina.youtrack.cloud/issue/SZ_Scooter_WEB-18

Заключение:

*ответь на несколько вопросов ниже в свободной форме

1. **Какой баг показался самым критичным?**

У меня нашелся один блокирующий баг, который воспроизводится только в браузере: Google Chrome (Версия 118.0.5993.89 (Официальная сборка), (64 бит)). Он блокирует возможность пользователя оформить заказ, так как в форме подтверждения заказа некликабельна кнопка “Да”. Браузер Google Chrome является самым популярным браузером, в связи с чем большинство пользователей не сможет оформить заказ, поэтому этот баг - самый критичный.

2. **На твой взгляд, какая самая «хитрая» серая зона есть в требованиях?**

Самая хитрая серая зона в требованиях - не описана функциональность окна подтверждения заказа. Ни в требованиях, ни в макетах это окно не описано - неожиданно было, когда оно появилось. Вот и думай - “баг или фича”.

Данная серая зона не описана в чек-листе, так как её нужно уточнять: возможно, данная функциональность должна была быть отменена в настоящей версии приложения, но разработчики её не убрали, в связи с чем - это будет багом. Возможно, эта функциональность была реализована до написания требований, после чего заказчик согласовал требования без этой функциональности, так как она не нужна - тогда это тоже баг. А возможно, описания этой функциональности нет в требованиях, потому что их не обновили - тогда это не баг, а фича. Таким образом, ввиду того, что функциональность окна подтверждения заказа требует уточнения, описание его работы в чек-листе не учтено.

Еще к серой зоне я бы отнесла тот факт, что можно выбрать два чек-бокса в поле “Цвет” одновременно, но заказать только один самокат. Если это не двухцветный самокат, тогда нужно реализовать требования так, чтобы пользователю на выбор был доступен один цвет.

3. **Проверенная тобой функциональность готова к релизу? Почему?**

Я считаю, что проверенная мной функциональность не готова к релизу, так как имеется большое количество критических багов и блокер. Блокирующий баг лишает большинства пользователей возможности сделать заказ. Критические баги, касающиеся валидации полей, могут нанести ущерб бизнесу: можно оформить заказ без указания адреса, оформить заказ несуществующей датой или прошедшей - такие ошибки могут повлечь за собой убытки для заказчика.

Ретест багов в мобильном приложении

Был проверен фикс багов. Из них не исправлен **1 баг,** исправлено —**3 бага.**

Список багов можно посмотреть здесь:

https://berkut-ekaterina.youtrack.cloud/issues?q=%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82:%20%7B%D0%91%D0%B0%D0%B3%D0%B8%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%A1%D0%B0%D0%BC%D0%BE%D0%BA%D0%B0%D1%82%7D

https://berkut-ekaterina.youtrack.cloud/issue/retest_samokat-2

https://berkut-ekaterina.youtrack.cloud/issue/retest_samokat-3

https://berkut-ekaterina.youtrack.cloud/issue/retest_samokat-4

https://berkut-ekaterina.youtrack.cloud/issue/retest_samokat-5

Регрессионное тестирование мобильного приложения по готовым тест-кейсам

Результаты выполнения регрессионных тестов можно посмотреть здесь:

https://docs.google.com/spreadsheets/d/1rHMYEk4Qi85XjzSsmtR6DUD-GJLKrV-btkvYDduTJ6c/edit?usp=sharing

Из **10 тестов** успешно прошло **1**, не прошло — **9**.

Список багов, найденных при тестировании, разбит по приоритетам:

1. Блокирующие:

Не обнаружено

2. Критичные:

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-1

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-2

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-3

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-4

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-7

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-8

3. Средний приоритет:

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-6

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-9

4. Низкий приоритет:

https://berkut-ekaterina.youtrack.cloud/issue/REG_M-5

Заключение:

*ответь на несколько вопросов ниже в свободной форме

1. **Какой баг показался самым критичным?**

Самыми критичными багами, на мой взгляд, оказались такие: курьер может принять отмененный заказ и курьер может принять заказ, который уже принял другой курьер. Такие баги могут лишить курьера возможности заработать денежные средства - а это

основная цель регистрации курьеров в данном приложении, бизнес может потерять денежные средства и репутацию, а продукт не будет пользоваться спросом.

2. **Такой продукт можно выпускать в релиз? Почему?**

Такой продукт не стоит выпускать в релиз, потому что имеется достаточно много критических багов. Если выпустить такой продукт в релиз, то он может не достичь основной пользовательской цели - заработать денежные средства, так как курьер может принять отмененный или уже взятый заказ. Кроме того, в карточках заказов некорректно отображаются данные о станции метро, адресе и комментарии к заказу - это также может повлечь отток пользователей. Ввиду того, что имеются такие баги, приложение может не пользоваться спросом, в связи с чем бизнес потеряет свои денежные средства. Таким образом, до устранения критических багов, выпускать продукт в релиз нецелесообразно.

Выводы о проделанной работе

*ответь на несколько вопросов ниже в свободной форме

Как для тебя прошла первая практическая часть проекта?

Первая часть проекта очень интересная. Мне понравилось работать как с веб-версией приложения, так и с мобильной версией. Очень занятно работать с интеграцией АПИ в мобильном приложении - на проекте по “мобилкам” у нас такого не было.

С какими сложностями пришлось столкнуться?

Ввиду того, что во время выполнения проекта по “мобилкам” у нас не было интеграции с АПИ, пришлось дополнительно изучить, как правильно выполнять запросы и использовать ответы в мобильном приложении - но мне, наоборот, понравилось! Еще была сложность с прохождением тест-кейса № 4 во время регресса - но тоже ничего, справилась! В общем, нет ничего невозможного :)

Что получилось хорошо, а что не очень? Какие мысли остались?

Остались мысли, что, наверное, я не всё учла во время тестирования веб-приложения, хотя старалась максимально описать и логику, и интерфейс. Но, один из принципов тестирования гласит, что “исчерпывающее тестирование невозможно”, поэтому даже если стараться составить 10 000 тест-кейсов, учесть все-все проверки (хотя это совсем нецелесообразно!), обязательно где-то выскочит маленький хитрый баг! В общем, я думаю, что всё получилось хорошо, мне очень понравилось работать над первой частью проекта.