1. Почему практики тест-дизайна нельзя применять сразу после получения требований?

Перед началом тест-дизайна должно быть два этапа: тест-план и тест-анализ.
На этапе тест-плана нужно правильно распределить ресурсы команды.
На этапе тест-анализа нужно:
1. Изучить документацию и требования;
2. Уточнить все неявные требования и выявить "серые зоны";
3. Декомпозировать и визуализировать требования;
4. Разделить проверки по приоритету.
Если пропустить эти этапы и приступить сразу к составлению тестовой документации можно:
1. Пропустить много недоработок в требованиях, "серые зоны";
2. Не успеть протестировать важную функциональность;
3. Упустить ошибки.

2. В какой ситуации классы эквивалентности и граничные значения могут существовать по отдельности? Аргументируй свой ответ и приведи примеры.

Классы эквивалентности и граничные значения не могут существовать отдельно. Так как ГЗ являются дополнением КЭ. Мы применяем ГЗ тогда, когда класс эквивалентности - это диапазон. Например: длина строки от 2 до 15 сивмолов. И не применяем ГЗ для класса у которого нет границ - это набор значений. Например, праздники (1 января, 23 февраля, 8 марта), обязательность заполнения поля (да, нет), формат текстового поля (английские буквы, русские буквы, спец.символы, только цифры).

3. Что такое эквивалентность? Что такое класс эквивалентности?

Значения, которые система обрабатывает одинаково - эквивалентны. Такие значения объединяют в класс эквивалентности - диапазон или набор.

4. Можно ли исключить проверку в середине диапазона в пользу проверок на границах, входящих в диапазон? Аргументируй свой ответ.

Нельзя. Чтобы подобрать тестовое значение желательно взять одно тестовое значение внутри класса. Значения на границах диапазона - это другая группа проверок. Но и нельзя исключать проверки на границах диапазона, так как чаще всего баги встречаются именно там.

5. Представь, что тебе нужно протестировать форму: у каждого поля есть валидатор. Результат работы формы зависит от комбинации данных в полях. Какие практики тест-дизайна следует применить и почему? Аргументируй свой ответ.

Если результат работы формы зависит от комбинации данных в полях, то стоит применить техники тест-дизайна: таблица принятия решений или Pairwise. Таблицу принятия решений стоит использовать тогда, когда параметров для проверки немного. Таблица принятия решений поможет структурировать данные и перебрать все возможные комбинации значений параметров, тем самым исключив все ошибки. Если параметров для проверки много, стоит применять технику Pairwise. Данная техника сократит проверки, но поможет проверить систему и выявить большинство ошибок.

6. Какими способами можно оптимизировать количество проверок при работе с таблицами принятия решений? Аргументируй свой ответ.

Оптимизировать количество проверок при работе с таблицами принятия решений можно, если один параметр зависит от другого. Например, пользователь получает скидку, если он зарегестрирован. То есть можно исключить проверку получает ли скидку незарегестрированный пользователь, тем самым на проверку уйдет меньше времени. Параметр скидка зависит от параметра регистрации, если регистрации нет, то параметра "скидка" не будет.

7. Ты знаешь, что попарное тестирование исключает часть проверок. Аргументируй, почему его можно применять.

Если пара в комбинации с другими параметрами вызовет ошибку, то такая же ошибка будет происходить в любой комбинации с такой парой. Например: ошибка вомпроизводится в Яндекс Браузере при разрешении экрана 1280x720 с ОС Win7. Скорее всего ошибка воспроизведется и с другими ОС. Таким образом Pairwise помогает исключить лишние проверки, обеспечить хорошее тестовое покрытие и выявить максимальное количество ошибок с минимальным набором тестовых значений. Кроме того составить набор тестовых значений можно с помощью специальных инструментов, что оптимизирует работы и позволит ничего не упустить.

8. Опиши, чем чек-лист отличается от тест-кейсов. Приведи примеры, где применяют и то, и другое.

Чек-листы и тест-кейсы - это тестовая документация. Чек-листы описывают только то, что нужно проверить. Чаще всего их используют для проектов, которые существуют уже давно и тестировщик полностью знаком с продуктом. Нет смысла описывать подробно как достичь определенного результата. Кроме того можно использовать чек-листы для составления проверок вёрстки или когда времени на тестирование мало. Тест-кейсы содержат подробные шаги проверки функциональности с описанием ожидаемого результата, дополнительных условий и окружения. Тест-кейсы лучше применять, когда проект новый и не всегда очевидно как работает система, для важной функциональности. Нет смысла писать тест-кейсы для проверки UI или функциональности, где очевидны шаги выполнения и их мало.

9. Как правильно составить баг-репорт? Какие элементы баг-репорта — обязательные? Почему?

Баг-репорт должен содержать:
1. Уникальный ID (чтобы свободно ориентироваться в баг-трекинговой системе и найти нужный баг по уникальному номеру),
2. Заголовок, шаги воспроизведения (чтобы знать при каких действиях появляется ошибка),
3. Результаты фактический и ожидаемый (чтобы сравнить как работает система и как она должна работать на самом деле),
4. Окружение(ошибка может происходить в определенном браузере или на определенном устройстве)
5. Приоритет (насколько важно быстро исправить ошибку).
Ещё могут быть необязательные поля:
1. Описание (если информации в заголовке недостаточно),
2. Предусловие (если нужно привести систему в определенный вид перед выполнением шагов),
3. Постусловие (если надо вернуть систему в исходное состояние после тестирования),
4. Дополнительные материалы (скриншот, скринкаст, логи).
Название лучше составлять по схеме "Что? Где? Когда?", из названия сразу должно быть понятно что произошло, где произошло и при каком условии. Программист должен понять из названия что произошло, не заглядывая в баг-репорт.

10. По каким правилам составляют заголовок баг-репорта? Что будет, если составить заголовок неправильно?

Название лучше составлять по схеме "Что? Где? Когда?", из названия сразу должно быть понятно что произошло, где произошло и при каком условии. Программист должен понять из названия что произошло, не заглядывая в баг-репорт. Если неправильно составить заголовок, то будет непонятно где произошла ошибка, какая ошибка произошла и почему. Программисту нужно будет тратить время на воспроизведение ошибки, чтобы понять что происходит. Но чаще всего если название составлено правильно, достаточно взглянуть только на него. Кроме того название не должно быть слишком объемным. Если оно будет слишком большим можно запутаться и дольше вникать что вообще не так. Если в названии недостаточно информации всегда можно вынести её в описание баг-репорта.

11. Из чего состоит клиент-серверная архитектура приложения? Кратко опиши функциональность каждого элемента.

Клиент-серверная архитектура состоит из 3 элементов:
1. Клиент - приложение с которым работает пользователь.
2. Сервер - система, к которой обращается клиент, чтобы получить ответ(данные).
3. Сеть - система из нескольких устройств, которая помогает клиенту и серверу обмениваться данными.
Веб-приложения состоят из фронтенда и бэкенда.
1. Фронтенд (frontend) — это видимая, клиентская часть приложения
2. Бэкенд (backend) — это логика и данные приложения, которые хранятся на сервере.
Например: мне нужно купить билет. Фронтенд (клиент) - я отправляю данные и запрос на покупку билета, бэкенд (сервер) - сервер обрабатывает данные и отправляет ответ на запрос - билет куплен.

12. Опиши этапы обработки запроса после того, как в адресную строку браузера вводят URL: https://yandex.ru.

После того как в адресную строку браузера вводят URL, браузер отправляет запрос на DNS - сервер, который его обрабатывает и возвращает IP - адресс доменного имени сайта. По IP-адрессу браузер получает от сервера всю необходимую информацию и отображает нужную страницу.

13. Опиши роль HTML, CSS и JS в создании веб-страницы. За что они отвечают?

HTML - язык разметки гипертекста, который создает структуру веб-страницы. Разметка указывает браузеру в каком формате отображать данные. (Например, на сайте есть заголовок “Заказать продукты”)
CSS - отвечает за внешний вид веб-страницы, помогает html- элементу придать индивидуальный стиль. (Например, заголовок “Заказать продукты” красного цвета, кегель 20)
JS - язык программирования, который отвечает за логику работы веб-страницы, помогают сделать веб-страницу более интерактивной (например, кнопка “Заказать”)

14. Что такое кэш? Зачем он нужен? Какое правило нужно соблюдать при работе с кэшем в тестировании?

Кэш (Casche Storage) - файлы, которые сохраняются на компьютере после посещения веб-страниц: изображения, аудио, видео, html, css, js файлы. Это нужно для того, чтобы при повторном входе на сайт, страница загружалась быстрее: браузер возмет файлы с компьютера, а не сервера. Веб-приожения часто обновляются и из-за файлов, которые хранятся в кэше могут возникать ошибки: могут разъехаться вёрстка, не работать кнопки. Во избежание ошибок нужно чистить кэш пред началом тестирования новой версии сайта, особенно если это мешает тестировать какую-то функциональность.

15. Опиши уровни модели TCP/IP. За что отвечает каждый уровень? С каким уровнем тестировщик работает чаще всего?

В модели TCP/IP есть 4 уровня:
1. Прикладной - он забирает у приложения данные и отправляет на следующий уровень обработки сетевой модели. Один из самых распространённых протоколов прикладного уровня — HTTP(S).
2. Транспортный - контролирует и обеспечивает передачу данных по сети. На этом уровне работают протоколы UDP и TCP.
3. Сетевой - на нём данные формируются в пакеты; чтобы точно рассчитать маршрут доставки, добавляются адреса клиента и сервера. На этом уровне работает протокол IP (Internet Protocol), который формирует IP-адрес.
4. Уровень сетевых интерфейсов - определяет, как упаковать данные, чтобы передать по сети. На нём также назначается оборудование, по которому данные передадутся в виде электрических или оптических сигналов.
Тестировщик чаще всего работает с прикладным и транспортным уровнями.

16. Чем отличаются протоколы TCP и UDP? В каких ситуациях применяют каждый из них? Приведи примеры.

Протокол TCP:
1. обеспечивает надежную передачу данных
2. данные разбиваются на фрагменты
3. перед отправкой данных перед клиентом и сервером устанавливается соединение
4. сообщает клиенту о статусе передачи данных
Применяется в случаях, когда важна надежность и упорядоченность: e-mail, WWW и др.
Протокол UPD:
1. указывает только порты сервера и клиента, не соединяет сервер с клиентом заранее
2. данные передаются целиком
3. не обеспечивает надежную передачу данных
4. обеспечивает скорость передачи данных
5. не гарантирует доставку данных, данные могут передаться хаотично
Применяется, когда важна скорость передачи данных и некритична потеря некоторых файлов: сервисы онлайн-просмотра видео, аудио и видеозвонки, передача потоковых аудио/видео.

17. Ты знаешь, что есть протоколы HTTP и HTTPS.
- Чем отличаются HTTP и HTTPS? В каких случаях не стоит пользоваться HTTP?
- Из каких компонентов состоит HTTP запрос: за что отвечает каждый?
- Какие HTTP-методы ты знаешь? За что они отвечают? Приведи примеры применения разных методов.
- Что такое код HTTP-ответа? Какие коды бывают?

Протокол HTTP передаёт данные в незащищенном виде. На сайтах, которые используют такой протокол можно перехватить и украсть личную информацию. Чтобы это избежать используют протокол с расширением защиты - HTTPS. Он шифрует соединение по криптографическим протоколам. Так сервер и клиент могут безопасно обмениваться данными.
HTTP не следует использовать для: электронных платёжных систем, для сервисов, которые собирают и хранят личную информацию пользователя, социальные сети и личные кабинеты. HTTP -запрос состоит из трех блоков:
1. Стартовая строка (метод , пути до ресурса в формате URL и версия протокола);
2. Заголовки (дополнительная информация от клиента серверу);
3. Тело сообщения (данные, которые передает клиент).
Самые распространённые методы:
1. GET - запрашивает данные *(Наприер: запросить список продуктов);*
2. POST - создаёт или получает данные *(Например: создание корзины продуктов);*
3. DELETE - удаляет данные *(Например: удаление созданной корзины);*
4. PUT - обновляет данные (*Например: изменение имени для набора продуктов).*
Код HTTP- ответа - это трехзначное число, код которого указывает успешно ли сервер обработал запрос. Коды ответов бывают:
1xx (информационные сообщения)
2xx(сообщения об успехе),
3xx (перенаправление),
4xx(клиентские ошибки),
5xx(ошибки сервера);

18. Опиши, из каких компонентов может состоять URL. За что отвечает каждый из них?

URL состоит из:
1. Схема - протокол, по которому передаются данные. Например: HTTP/HTTPS
2. Логин:пароль - указывает серверу, какой пользователь к нему обратился;
3. Имя хоста:порт - доменное имя или IP -адрес сервера, к которому обращается пользователь и номер, который сообщает куда отправить запрос;
4. Пути - местоположение ресурса
5. Параметр запроса - дополнительные параметры вида ключ=значение
6. Якорь - дополнительный указатель, который помогает сразу попасть в нужную часть веб-страницы.

19. Какие виды мобильных приложений бывают? В чём особенность каждого?

Мобильные приложения делят на три вида:
Нативные приложения - нужно скачать установочный файл и установить на мобильное устройство. Приложение пишут на Java и Kotlin для Android, Swift и Objective-C для iOS. Они более производительны, задействуют процессор и оперативную память устройства, могут применять инструменты мобильного устройства: камера, GPS и д.р.
Веб приложения - используют клиент-серверную архитектуру и адаптированы под мобильные устройства. Их не нужно устанавливать, можно запустить в браузере с помощью URL и интернет-соединения.
Гибридные приложения - Совмещает в себе свойства нативного и веб- приложений. Его пишут отдельно под каждую платформу. Веб-содержание наполняют: HTML, CSS и JavaScript. Они могут быть едины для всех платформ. Их нужно устанавливать, но чтобы оно работало нужен доступ к интернету.

20. Чем эмулятор отличается от реального устройства? Кратко опиши недостатки и преимущества при тестировании.

Эмуляторы как правило доступны и и бесплатны для всех, что помогает сократить траты компаний. *На эмуляторе можно протестировать:* как приложение ведет себя при разной ориентации экрана, на разный версиях ОС, протестировать работу приложения с GPS, работу приложения при разной скорости сети, работу и вёрстку приложения с разными разрешениями экрана. С эмуляторра проще снимать скриншоты и скринкасты. *Минусы*: нельзя проверить на эмуляторе как работает приложение с железом, производительность. Нельзя проверить взаимодействие с Touch ID/Face ID, со встроенной клавиатурой, работу приложения в режиме энергосбережения, взаимодействие с фоновыми приложениями, жесты, мобильные браузеры (от Samsung, Xiaomi и д.р.), внешние прерывания (смс, звонки). Всё это лучше проверять на реальных устройствах.

21. Проверь, есть ли ошибки в JSON
Если да, напиши номер строки с ошибкой, опиши ошибку и предложи исправление.

```
01  { "Меню": {
02      "id": "1",
03      "value": Файл,
04      "list":
05          "items": {
06              "new_doc": {
07                  "value": "Новый",
08                  "onclick": "create_new_doc"},
09              "open_doc": {
10                  "value": "Открыть...",
11                  "onclick": "open_doc"},
12              "save_doc": {
13                  "value": "Сохранить",
14                  "onclick": "save_doc",
15              "save_as_doc"
16                  "Сохранить как...",
17                  "onclick": "save_as_doc"},
18              "print_option": {
19                  "value": "Параметры печати",
20                  "onclick":
21                      "show_print_option":{
22                          "Цвет":"Насыщенный",
23                          "Черно-белая печать?":"",
24                          "Размер печати":"A4"}}
25              }
26          }
27      }
28  }
```

Ссылка на иллюстрацию:

https://code.s3.yandex.net/qa/schemes/diploma-29.png

Строка 3: значение Файл должно быть заключено в кавычки. Правильный вариант: "value": "Файл",
Строка 5:ключ “items” лишний
Строка 15: после ключа “save_as_doc” пропущено двоеточие. Правильный вариант: “save_as_doc”: “Сохранить как...”
Строка 20: после “onclick” не хватает значения. Правильный вариант: “onclick”: “print_option”,
Строка 21: Неправильно оформлен массив. Не хватает значение “value” и квадратных скобок []. Правильный пример оформления: "show_print_option": {
“value”: [{"Цвет": "Насыщенный",
"Черно-белая печать": "",
"Размер печати": "A4"}]
}

22. Что такое реляционная база данных? Чем она отличается от нереляционной?

Реляционные базы данных состоят из таблиц и связей между ними. Нереляционные базы - это базы данных в которой не используется табличная схема столбцов и строк. Например, данные хранятся в формате "ключ":"значение" или графов.

23. Напиши, какие виды JOIN бывают. В чем особенность каждого?

Оператор JOIN - используют для соединения таблиц.
NNER JOIN - возвращает строки строго на пересечении двух таблиц, формирует выборку только из тех данных, у которых выполнено условие присоединения.
LEFT OUTER JOIN - возвращаются все строки левой таблицы. Если есть недостающие данные, вместо строк правой таблицы подставляются NULL-значения.
RIGHT OUTER JOIN - возвращаются все строки правой таблицы. Если есть недостающие данные, вместо строк левой таблицы подставляются NULL-значения.
FULL OUTER JOIN - возвращаются строки и левой, и правой таблиц. Если есть строки, у которых не выполняется условие соединения, они дополнятся NULL-значениями.

Исходные данные для заданий ниже

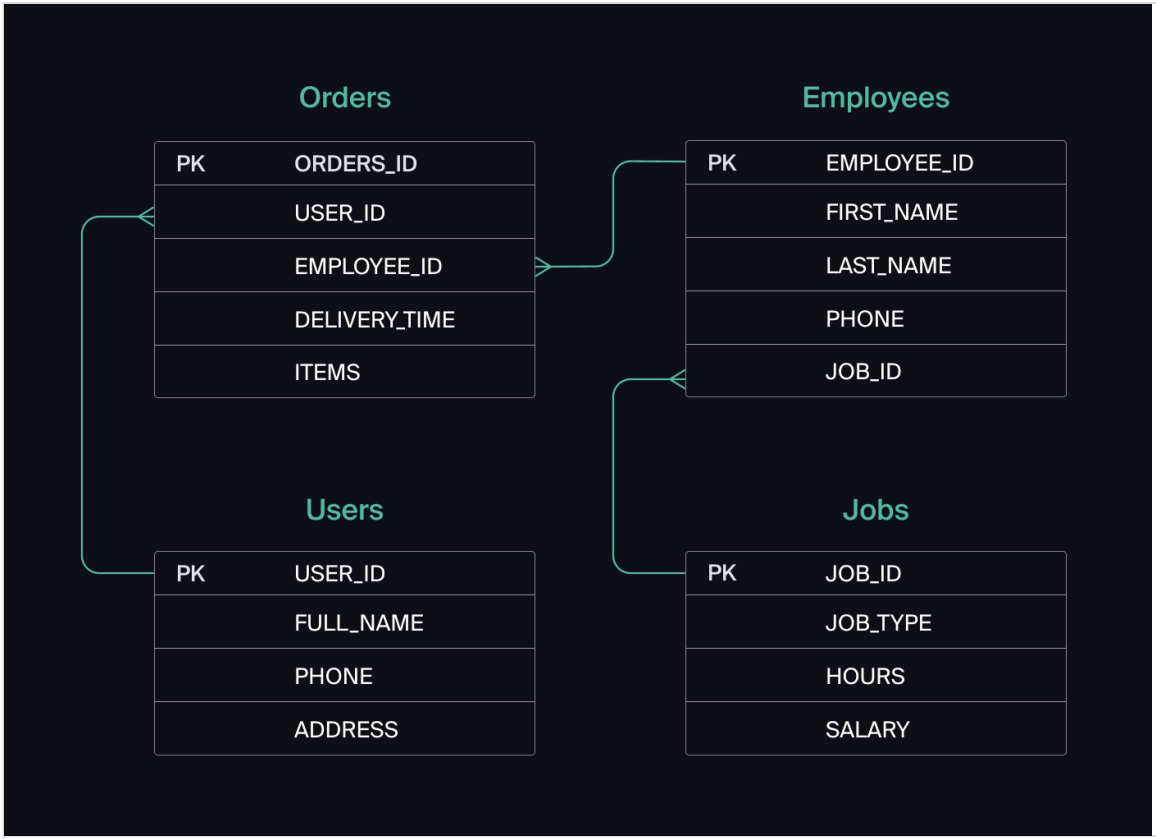
Ты тестируешь сервис, который доставляет еду за 30 минут. Пока это маленький стартап, поэтому ты работаешь всего с четырьмя таблицами:

Orders — все доставленные заказы;
ORDERS_ID — ID заказов, int;
USER_ID — ID пользователей, int;
EMPLOYEE_ID — ID сотрудников, int;
DELIVERY_TIME — время доставки в минутах, int;
ITEMS — список товаров, char;

Users — пользователи;
USER_ID — ID пользователей, int;
FULL_NAME — полное ФИО пользователя, char;
PHONE — номер телефона пользователя, char;
ADDRESS — адрес пользователя, char;

Employees — работники;
EMPLOYEE_ID — ID сотрудника, int;
FIRST_NAME —имя сотрудника, char;
LAST_NAME — фамилия сотрудника, char;
PHONE — телефон сотрудника, char;
JOB_ID — ID специализации, int;

Jobs — типы работ в сервисе
JOB_ID — ID специализации, int;
JOB_TYPE — тип специализации, char;
HOURS — число рабочих часов в неделю, int;
SALARY — зарплата сотрудника с данной специализацией в рублях, int;



Ссылка на иллюстрацию:

https://code.s3.yandex.net/qa/schemes/diploma-33.png

24. В службу поддержки пришло много жалоб: заказы, в которых есть гречка, доставляют почти час, хотя сервис обещает успеть в 30 минут.

Проверь, действительно ли курьеры опаздывают. Выбери все заказы с товаром «гречка» и временем доставки свыше 30 минут. В результирующей таблице должны быть ID заказов и ID курьеров.

В ответе приложи SQL-запрос.

```
SELECT
    Order.ORDERS_ID AS ORDERS_ID,
    Employees.EMPLOYEE_ID AS EMPLOYEE_ID
FROM
  Orders
INNER JOIN Employees ON Employees.EMPLOYEE_ID =  Orders.EMPLOYEE_ID
WHERE
Order.ITEMS LIKE '%гречка%' AND Order.DELIVERY_TIME > 30
GROUP BY
ORDERS_ID,
EMPLOYEE_ID;
```

25. Менеджер предложил добавить новую функциональность в продукт: мониторинг, который показывает самых активных клиентов за всё время работы компании.

Проверь, что список пользователей корректно выводится на экран. На этом этапе разработки достаточно проверить только ID клиентов.

Выбери пять самых активных клиентов по количеству заказов.

В результирующую таблицу выведи ID каждого пользователя и число заказов.

Отсортируй данные по убыванию числа заказов, выбери пять самых активных клиентов.

В ответе приложи SQL-запрос.

```
SELECT
    USERS_ID,
    COUNT(ORDERS_ID) AS ORDERS_ID
FROM
  Orders
GROUP BY
USERS_ID
ORDER BY
  ORDERS_ID DESC
  LIMIT
  5;
```

26. Из бухгалтерии пришёл баг-репорт: зарплаты сотрудников рассчитываются некорректно. Оказалось, что почти все ошибки в расчётах — в расчётных листах менеджеров.

Выведи список ID всех сотрудников, у которых в специализации содержится «менеджер», с зарплатой больше 70 000 рублей.

В ответе приложи SQL-запрос.

```
SELECT
    Employees.EMPLOYEE_ID AS EMPLOYEE_ID,
 FROM
  Employees
INNER JOIN Jobs ON Jobs.JOB_ID =  Employees.JOB_ID
WHERE
Jobs.JOB_TYPE LIKE ‘%менеджер%’ AND Jobs.SALARY > 70000;
```

27. Изучи три ситуации и ответь на вопрос: стоит ли писать автотесты в этом случае? Аргументируй свой ответ.

1) Проект существует давно, у него написано много ручных тестов.

2) Проект временный: продлится всего несколько месяцев.

3) Проект нестабилен: в функциональность часто вносят изменения.

1) Если проект существует давно и у него написано много ручных тестов, то следует писать автотесты чтобы оптимизировать работу и проводить тестирование быстрее. Это значительно сэкономит ресурсы команды.
2) Нет смысла писать автотесты, если проект временный, на их написание уйдет много времени, но используют их всего раз или два. Лучше в данном случае ограничиться ручным тестированием.
3) Если проект нестабилен нет смысла писать автотесты. Требования часто меняются и любые изменения значительно повлияют на то какие проверки нужно проводить. Придётся постоянно переписывать код, на что уйдет много времени. В данном случае лучше ограничиться ручным тестированием.