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

После получения требований нужно провести тест-анализ. Найти серые зоны, визуализировать, разбить на атомарные уровни. Далее написать чек-лист. К тому же требования могут измениться, нужно уточнить все детали перед тест-дизайном.

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

В наборе. У таких класов нет границ, чем у диапазона. Например: список фамилий студентов Яндекс Практикума, на сайте из выпадающего списка можно выбрать один из трех языков - это набор.

**3. Что такое эквивалентность? Что такое класс эквивалентности?**

Эквивалетность - это равноценность обьектов. Класс эквивалентности - значения, которые программа обратотает одинаково и приведет к одному результату.

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

Нет. Нужны позитивные проверки. А проверки на границах нужны, чтобы выявить ошибки, т.к. на границах чаще всего ошибки.

**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.**

Позитивная проверка для того, чтобы убедиться, что приложение работает так как задуманно. Негативные проверки - проверяют как поведет себя приложение, если использовать не так, как задумали, вводить данные не из требований. КЭ и ГЗ для проверки валидации полей и ошибок на границах. Оптимизация проверок, чтобы уменьшить дубликатов.

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

Попарное тестирование. Когда проверок слишком много, попарное тестирование сокращает кол-во проверок в несколько раз и можно проверить только те случаи где пара проверок встречается хотя бы раз.

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

В отличие от чек-листа, тест-кейс - это проверка одного элемента, в тест-кейсах подробно описано , что делать , чтобы достить ОР. Чек-лист - список элементов, которые нужно проверить.

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

Чтобы правильно заполнгить баг-репорт, в них должны быть обязательные поля, такие как: заголовок, ID, шаги воспроизведения, результаты, окружения , приоритет. В баг-репортах коротко и ясно описывают проблему. Это для того, чтобы коллеги могли быстро понять суть проблемы не вникая в детали, экономит время и силы коллег.

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

Заголовок должен отвечать на 3 вопроса ЧТО? ГДЕ? КОГДА?. В заголовке описывают не ожидаемый результат, а фактический. При неправильности составлении заголовка, коллеги не поймут что за ошибка, где ошибка и когда она происходит.

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

Клиент, интернет , сервер. Клиент отвечает за взаимодействие с пользователем, интернет за связь клиента и сервера, сервер - за логические операции , вычисления и хранения данных.

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

Стартовая строка состоит из 3 элементов: метод, путь до ресурса, версия протокола. Метод указывает действие: бэкенд принимаетего в обработку, путь до ресурса- адрес по которому фронтенд отправляет запрос на бэкенд, номер версии протокола. Далее передается заголовок запроса - дополнительная информацияя о клиенте: доменное имя, информация об окружении и допустимый язык в ответе, параметр состояния соединения между клиентом и сервером, парметр сообщения о зашифровонном ответе. Тело сообщения в данном запросе нет.

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

Это данные веб-страницы. Они сохраняются на компьютере клиента. Кэш нужен для быстрой загрузки веб-страницы при повторных переходах на эту веб-страницу. Браузер берет файлы из кэш-хранилища, а не с сервера. Перед тестированием нужно убедиться , что кэш очистен - это глпвное правило при работе с кэшем, иначе тестироващик будет видеть старую страницу со своего компьютера, а не новую с сервера.

**13. 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?**

- HTTP в отличие от HTTPS передает данные в незащищенном виде. HTTP не стоит пользоваться при использовании своих личных данных, например данные банковской карты.
- Стартовая строка, заголовки, тело сообщения. Стартовая строка отвезачет за метот, путь до ресурса и версию протокола; заголовки отвечает за дополнительную информацию от фронтенда к бэкенду; тело сообщения отвечает за данные, которые передает фронтенд.
- GET, POST, PUT, DELETE - распространненые методы. GET запршивает определенные данные у бэкенда по определенному адресу; POST отправляет данные на бэкенд; PUT изменяет данные на бэкенде; DELETE удаляет данные на бэкенде. Например: методом GET пользователь может запросить музыку определенного испольнителя - "песни Виктора Цоя"; PUT может поменять данные профиля соц. сети ВКонтакте - фамилию, имя , возраст, фотографию; POST можно вызвать такси в приложении Яндекс Go, при этом отправляются данные пользователя: номер, геопозиция, имя; DELETE можно удалить свой профиль из соц.сети ВКонтакте.
- Это код результата, которое указывает: успешно ли сервер обработал запрос. Информационные 100-199, успешные 200-299, перенаправления 300-399, клиентские ошибки 400-499, серверные ошибки 500-599.

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

Схема, логин, пароль, символ @, имя хоста, порт, путь, параметры запроса, якорь. Схмета отвечает за передачу данных; логин и пароль отвечает за права доступа пользователя, указывает серверу какой пользователь к нему подключился; символ @ разделяет логин и пароль, имя хоста и порт; хост и порт отвечает за доменное имя и IP-адресс, к которому обращается пользователь; путь отвечает за расположение ресурса; параметры запроса отвечает за дополнительную информацию, которая нужна, чтобы получить данные и отобразить страницу; якорь позволяет сразу попасть в нужную часть веб страницы, к заголовку или обзацу в тексте.

**15. Какие виды мобильных приложений бывают? В чём особенность каждого?**

Веб-приложения, нативные приложения, гибридные приложения. Веб-приложение не нужно устанавливать, чтобы запустить веб-приложение нужен браузер, URL приложения и интернет. Нативное приложение нужно устанавливать, они более производительны, применяют инструменты устройства например камеру, GPS. Интрефейс нативного приложения часто отличается от веб-приложения. Гибридное прилоржение нужно устанавливать, чтобы оно работало в большинстве случаев нужен интернет. Его пишут отдельно под каждую платформу, веб-содержание могут быть едины для всех платформ. Комбинируют свойства нативного и веб-приложения.

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

В эмуляторе в отличие от реального устройства можно протестировать, но не весь функционал приложения. Плюсы эмулятора: покрыть множества праметров мобильного утсройства; быстрее приступить к тестированию, если нет реального устройства; проверить версию приложения, которая еще в разработке; автоматизировать тестирование интерфейса. Минусы эмулятора в том, что не протестируешь все функции приложения, например датчики, производительность.

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

```
{ "Меню": {
    "id": "1",
    "value": Файл,
    "list":
        "items": {
            "new_doc": {
                "value": "Новый",
                "onclick": "create_new_doc"},
            "open_doc": {
                "value": "Открыть...",
                "onclick": "open_doc"},
            "save_doc": {
                "value": "Сохранить",
                "onclick": "save_doc",
            "save_as_doc"
                "Сохранить как...",
                "onclick": "save_as_doc"},
            "print_option": {
                "value": "Параметры печати",
                "onclick":
                    "show_print_option":{
                        "Цвет":"Насыщенный",
                        "Черно-белая печать?":"",
                        "Размер печати":"A4"}}
            }
        }
    }
}
```

Ссылка на иллюстрацию:

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

3- нету ковычек. Нужно вставить ковычки по краям слова "Файл"
14-нету закрывающей фигурной скобки. Нужно вставить закрывающую фигурную скобку в конце строки перед запятой
15-отсутствие двоеточие и открывающей фигурной скобки. Вставить двоеточие в конце строки и открывающую фигурную скобку
16-нет ключа. Перед значением нужно задать ключ далее двоеточие потом значение
28-лишняя фигурная скобка. Удалить лишнюю скобку

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

Это данные состоящие из таблиц и связей между ними. Реляционная бд хранит данные в таблицах, а нереляционая может хранить данные в графе, json документ.

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

INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN. INNER JOIN - возвращает строки на пересечении двух таблиц, LEFT JOIN - возвращает все строки левой таблицы, RIGHT JOIN - возвращает все строки правой таблицы, FULL JOIN - возвращает строки и левой и правой таблиц.

**Исходные данные для заданий ниже**

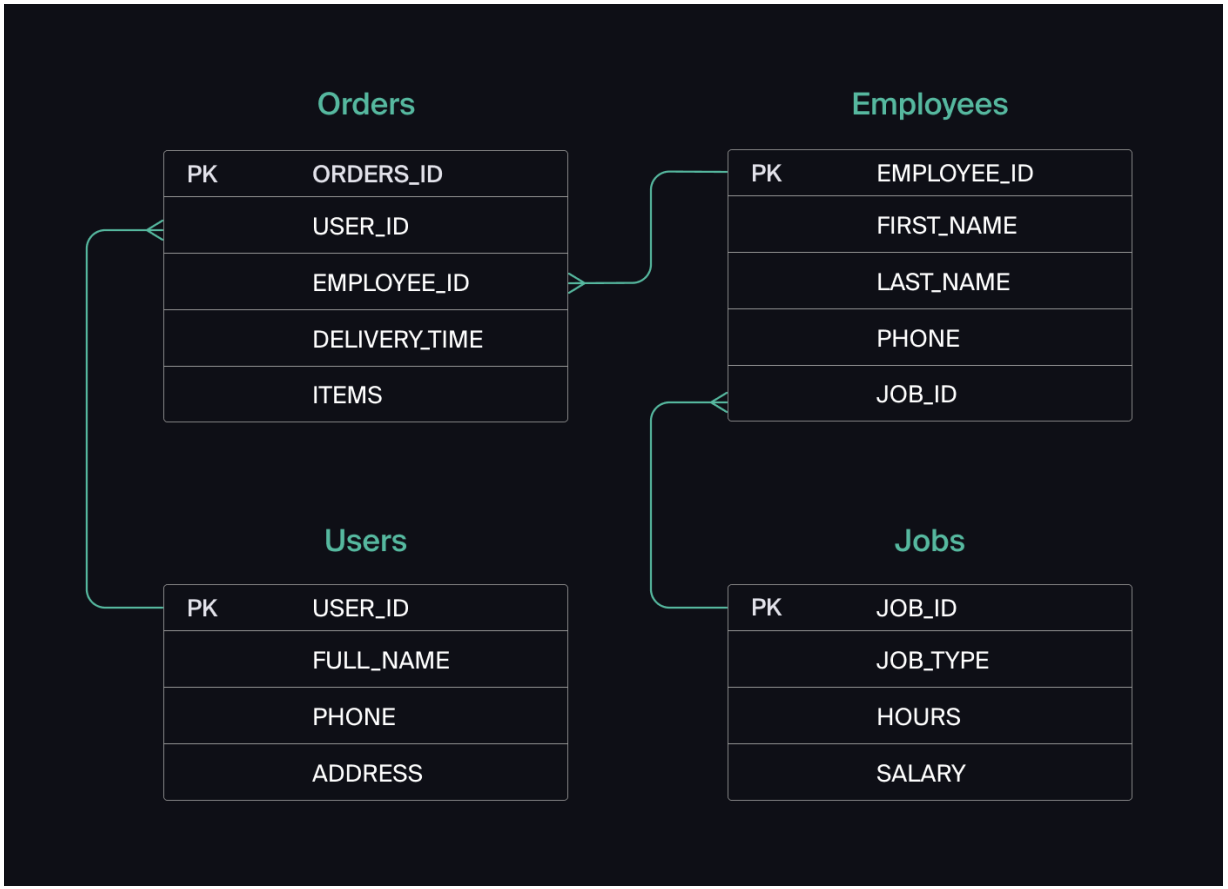
Ты тестируешь сервис, который доставляет еду за 30 минут. Пока это маленький стартап, поэтому ты работаешь всего с четырьмя таблицами:

**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

**20. 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, где есть хотя бы один товар - «гречка» и время доставки свыше 30 минут. В результирующей таблице должны быть ID заказов и ID курьеров.**

**В ответе приложи SQL-запрос.**

```
SELECT ORDERS_ID, EMPLOYEE_ID
FROM  Orders
WHERE ITEMS LIKE '%гречка%' AND DELIVERY_TIME>30;
```

**21. 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 USER_ID, Count(ITEMS)
FROM Orders
GROUP BY USER_ID
ORDER BY Count(ITEMS) DESC
LIMIT 5;
```

**22. 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
FROM Employees
INNER JOIN Jobs ON Employees.JOB_ID=Jobs.JOB_ID
WHERE Jobs.JOB_TYPE LIKE '%менеджер%' AND Jobs.SALARY>70000;
```

**23. 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 - да, т.к. у проекта уже есть много ручных тестов, проект существует давно, а значит проект достаточно большой. Автоматизация позволить сократить время, затраты и повысит продуктивность.
2 - нет. Проект маленький. Нет времени на разработку автотестов, ручных тестов будет мало и можно провести их быстро.
3 - нет. Автотесты придется постоянно изменять - это затратно, долго и не повысит продуктивность.

**Комментарий**