**Вопросы от HR на скрининге**

**1. Почему необходимо разделять тестирование на отдельные этапы?**

Разделение тестирования на этапы позволяет:

* Локализовать ошибки на ранних стадиях (юнит-тестирование, интеграционное, системное и т. д.).
* Управлять рисками, контролировать качество.
* Обеспечивать эффективную регрессию.
* Обеспечивать независимость тестирования от разработки.

**2. Назови главное преимущество гибких методологий по отношению к другим методологиям**

Главное преимущество — **гибкость и адаптивность**. Agile позволяет быстрее реагировать на изменения, выпускать ценные функции раньше и тесно взаимодействовать с пользователем.

**3. Что такое Definition of Ready и Definition of Done?**

* **DoR** (Definition of Ready) — критерии, которым должна соответствовать задача, чтобы её взяли в работу (достаточная детализация, нет блокеров, согласованы требования).
* **DoD** (Definition of Done) — критерии, которым должна соответствовать задача, чтобы считаться завершенной (код написан, протестирован, задеплоен и соответствует требованиям).

**4. Что такое User Acceptance Testing?**

UAT (приемочное тестирование) — это тестирование, которое проводят **реальные пользователи** или бизнес-аналитики перед релизом, чтобы убедиться, что система соответствует бизнес-требованиям.

**5. Хорошие атрибуты отчета о тестировании?**

* Четкость и понятность.
* Обоснованные выводы.
* Метрики (количество тестов, покрытие, количество багов, их критичность).
* Рекомендации по качеству релиза.
* Описание рисков и их влияние.

**6. Опиши, как составлял тест-кейсы для нового функционала. Какие техники тест-дизайна применял?**

Пример:  
При тестировании сложной формы с несколькими зависимыми полями использовал:

* **Классы эквивалентности и анализ граничных значений** (проверка валидных/невалидных данных).
* **Попарное тестирование (Pairwise Testing)** — для сокращения количества тестов при множестве параметров.
* **Таблицы принятия решений** — для проверки всех возможных комбинаций условий.
* **State Transition Testing** — если система зависит от предыдущих состояний (например, процесс одобрения заявки).

**7. В чем разница между тест-кейсом и чек-листом? Когда какой документ необходимо использовать?**

* **Тест-кейс** — детализированная пошаговая инструкция с входными данными и ожидаемым результатом. Применяется для критичных и сложных сценариев.
* **Чек-лист** — список проверок без детализированных шагов. Используется, когда нужно быстро пробежаться по основному функционалу.

Пример:

* Для тестирования формы входа (проверка логина, пароля, капчи) **можно использовать чек-лист**.
* Для тестирования сложного алгоритма расчета скидки с кучей условий **лучше использовать тест-кейсы**.

**8. Чем отличается нагрузочное тестирование от стресс-тестирования?**

* **Нагрузочное тестирование** — проверяет, как система ведет себя при **ожидаемой** нагрузке.
* **Стресс-тестирование** — проверяет, как система ведет себя при **чрезмерной** нагрузке и в условиях нехватки ресурсов.

**9. Кейс-задача: какую задачу выбрать?**

Выбор зависит от контекста, но безопаснее выбрать **функционально-интеграционное тестирование системы "Б"**, так как оно связано с новым функционалом и несет больше рисков.

Обоснование:

* В **системе "А"** уже отлаженные процессы, и регрессия, скорее всего, выявит меньше проблем.
* В **системе "Б"** функционал новый, неизвестный, а значит, вероятность критических дефектов выше.

**10. Какие шаги предпринимаешь перед тестированием новой функциональности?**

1. Анализ требований (что должно работать, бизнес-логика).
2. Определение тестового покрытия (какие кейсы критичны). это не про готовые тест-кейсы, а про список критичных аспектов, которые обязательно нужно проверить.
3. Определение входных данных и тестовых окружений.
4. Выбор техник тест-дизайна.
5. Написание тест-кейсов.

**11. Как расставляешь приоритеты при тестировании различных функциональностей?**

Приоритизация по:

* **Критичности для бизнеса** (что приносит деньги и используется чаще).
* **Рискам** (где выше вероятность сломаться).
* **Сложности исправления** (чем сложнее фикс, тем раньше тестируем).

**12. Как будешь действовать, если на проде после релиза не оказалось функциональности, которую тестировал?**

1. Проверю релизные ноты и список изменений.
2. Свяжусь с разработкой и продакт-менеджером для выяснения причины.
3. Проверю, был ли код влит в продакшен. (можно проверить в системе контроля версий или по логам деплоя)
4. Если функциональность была пропущена — инициирую выпуск срочного хотфикса.

**13. Ты получил жалобу от клиента на нестабильную работу приложения, но воспроизвести проблему не можешь. Что делать?**

1. Уточнить у клиента детали (когда, при каких условиях, какие шаги).
2. Запросить логи, снять метрики нагрузки.
3. Проверить окружение пользователя (браузер, версия ОС).
4. Попробовать воспроизвести на аналогичном стенде.
5. Если проблема случайная — включить логирование, попросить пользователя записать видео или протестировать в реальном времени.

**Задачи на собесе**

1. Техническое задание from-tend

URL: <https://dindonpingpong.github.io/flip-coin/>

**Описание:** Перед вами веб-интерфейс для игры "Подбрасывание монеты", который позволит пользователям выбирать между "орлом" и "решкой", устанавливать ставку и запускать игру с помощью кнопки "сыграть". Интерфейс должен включать анимацию во время розыгрыша и отображать результат после завершения игры.

**Требования:**

1. **Выбор стороны:**
   * Пользователь должен иметь возможность выбрать "орёл" или "решка" с помощью радиокнопок.
   * По умолчанию ни одна из сторон не должна быть выбрана.
2. **Ставка:**
   * Пользователь должен ввести сумму ставки в текстовое поле.
   * Ставка должна быть валидной (например, положительное число).
   * Должно быть ограничение на максимальную ставку (например, 1000 единиц).
3. **Кнопка "сыграть":**
   * При нажатии на кнопку "сыграть" должна запускаться анимация подбрасывания монеты.
   * Кнопка должна быть активна только при корректном выборе стороны и валидной ставке.
4. **Анимация:**
   * Во время розыгрыша должна отображаться анимация, показывающая процесс подбрасывания монеты.
   * Анимация должна длиться 2-3 секунды.
5. **Результат:**
   * После завершения анимации должно отображаться сообщение с результатом (выигрыш или проигрыш) и итоговая сумма.

**Задание:**  
Опишите тестовые сценарии, которые необходимо включить в тестовую модель при проведении будущего тестирования.

1. Задача  
   Есть система аунтефикации на продакшене. Бизнесу этот вариант не понравился и он сказал переделать. Системного анализ не проводили, бизнес требований тоже нет. Разработчик реализовал через GET запрос  
   <https://example.com/api/login?login=login&password=password>   
   Как бы ты протестировал такое ?