**1. Что такое тестирование программного обеспечения? Каковы его основные цели?**

Ответ:

Тестирование ПО - это проверка соответствия реальных и ожидаемых результатов поведения программы на конечном наборе тестов.

Цели:

Выявление дефектов до их обнаружения пользователями.

Предоставление информации о текущем состоянии продукта.

Проверка соответствия ПО заявленным требованиям.

**2. Принципы тестирования: "Тестирование демонстрирует наличие дефектов" и "Исчерпывающее тестирование невозможно".**

Ответ:

"Тестирование демонстрирует наличие дефектов": Тестирование помогает найти ошибки, но не гарантирует их полное отсутствие.

"Исчерпывающее тестирование невозможно": Проверить все возможные комбинации входных данных и сценариев физически нереально из-за ограниченности ресурсов.

**3. Что такое баг в тестировании? Опиши жизненный цикл бага.**

Ответ:

Баг - отклонение фактического результата от ожидаемого.

Жизненный цикл бага:

New → 2. Assigned → 3. Fixed → 4. Closed (или Re-opened при неисправлении).

Возможные состояния: Rejected, Deferred.

**4. Разница между UX и UI дизайном.**

Ответ:

UX (User Experience): Опыт пользователя, удобство достижения целей.

UI (User Interface): Визуальное оформление интерфейса (цвета, кнопки, шрифты).

**5. Этапы жизненного цикла разработки ПО (SDLC).**

Ответ:

Анализ требований.

Планирование.

Проектирование.

Разработка.

Тестирование.

Развертывание.

**6. Модели разработки ПО: водопадная vs V-образная.**

Ответ:

Водопадная: Линейная последовательность этапов. Нельзя вернуться назад. Подходит для проектов с четкими требованиями.

V-образная: Тестирование планируется параллельно разработке. Акцент на верификацию и валидацию. Подходит для критичных систем (медицина, безопасность).

**7. Agile и его ценности.**

Ответ:

Agile - гибкая методология с итерациями и быстрым реагированием на изменения.

Ценности:

Люди важнее процессов.

Рабочий продукт важнее документации.

Сотрудничество с заказчиком.

Готовность к изменениям.

**8. Разница между Scrum и Kanban.**

Ответ:

Scrum: Работа делится на спринты (1-3 недели), есть роли (Scrum-мастер, Product Owner).

Kanban: Гибкое управление задачами через визуальную доску ("To Do", "In Progress", "Done"). Нет фиксированных итераций.

**9. User Story.**

Ответ:

Формат:

"Как [роль], я хочу [функцию], чтобы [ценность]".

Пример: "Как пользователь, я хочу фильтровать товары по цене, чтобы быстро найти подходящий вариант".

Критерии INVEST: Independent, Negotiable, Valuable и др.

**10. Этапы STLC.**

Ответ:

Планирование и анализ требований.

Разработка тест-кейсов.

Выполнение тестов и фиксация дефектов.

Анализ результатов и отчетность.

**11. Виды тестирования.**

Примеры:

Функциональное: Проверка работы функций (например, авторизация).

Нагрузочное: Оценка производительности при высокой нагрузке.

Юзабилити: Удобство интерфейса.

**12. Статическое vs динамическое тестирование.**

Ответ:

Статическое: Без запуска кода (анализ документов, кода).

Динамическое: С запуском кода (проверка поведения программы).

**13. Валидация и верификация.**

Ответ:

Верификация: "Правильно ли мы делаем продукт?" (соответствие спецификации).

Валидация: "Правильный ли продукт мы делаем?" (соответствие нуждам пользователя).

**14. Требования к ПО и их свойства.**

Свойства: Полнота, однозначность, непротиворечивость, тестируемость.

**15. Методы тестирования: черный, белый, серый ящик.**

Ответ:

Черный ящик: Без доступа к коду (тестирование через UI).

Белый ящик: С доступом к коду (модульное тестирование).

Серый ящик: Частичный доступ (например, тестирование API).

**16. Позитивное и негативное тестирование.**

Ответ:

Позитивное: Проверка штатных сценариев (корректный ввод пароля).

Негативное: Проверка обработки ошибок (неверный формат email).

**17. Фронтенд vs бэкенд тестирование.**

Ответ:

Фронтенд: UI, взаимодействие с пользователем.

Бэкенд: Логика, базы данных, API.

**18. Функциональное и нефункциональное тестирование.**

Ответ:

Функциональное: "Что делает система?" (например, оплата товара).

Нефункциональное: "Как работает система?" (производительность, безопасность).

**19. Дымовое vs регрессионное тестирование.**

Ответ:

Дымовое: Проверка ключевых функций после сборки.

Регрессионное: Проверка, что изменения не сломали существующий функционал.

**20. Санитарное тестирование.**

Ответ:

Проверка работоспособности конкретной функции после изменений. Глубже, чем дымовое, но ужеще, чем регрессионное.

**21. Какие виды тестовой документации ты знаешь?**

Ответ:

• Тест-план. Описывает весь объем работ по тестированию.

• Тестовая стратегия. Определяет то, как тестируем ПО.

Это набор идей, которые направляют процесс тестирования.

• Чек-лист. Список, содержащий ряд необходимых проверок.

• Тест-кейс. Описывает наши тесты. Говорит, как их выполнить, при каких условиях и что должно получиться после выполнения шагов, которые заложены в тест-кейсе.

• Тест-сьют. Комплект тест-кейсов для исследуемого ком-

понента или системы.

• Баг-репорт. Содержит полное описание дефекта.

• Отчёт о результатах тестирования. Здесь обобщаются

все результаты работ по тестированию.

• Use Case. Сценарий взаимодействия пользователя с про-

граммным продуктом для достижения конкретной цели.

**22. Что такое чек-лист и как он используется в тестировании?**

Ответ:

Чек-лист – это список, содержащий ряд необходимых проверок во время тестирования программного продукта

ЗАДАЧИ, КОТОРЫЕ РЕШАЕТ ЧЕК-ЛИСТ:

• Систематизирует процесс. Чек-лист разбивает сложную работу на части и помогает не упустить из внимания важные детали

• Облегчают делегирование. С инструкцией сотрудникам проще разобраться в новой задаче без потери качества

• Снижают необходимость в контроле. Руководитель может от-

следить ход работ и корректировать процесс на любом этапе.

Четкий алгоритм облегчает проверку задач.

**23. Опиши структуру тест-кейса. Чем он отличается от чек-листа?**

Тест-кейс – это документ, который описывает наши тесты. Говорит, как их выполнить, при каких условиях и что должно получиться после выполнения тех шагов, которые

заложены в тест-кейсе, то есть каков ожидаемый результат.

• ID – уникальный номер. Обычно проставляется автоматиче-

ски в системах хранения тест-кейсов.

• Краткое описание тест-кейса (Name). Название тест-кейса

должно быть коротким и понятным. Оба эти слова важны.

• Ссылка на требования. Ссылка на требование или ТЗ, на осно-

ве которого был составлен тест-кейс.

• Автор (Author). Тестировщик, написавший тест-кейс.

• Приоритет (Priority) – насколько важен этот тест-кейс, в какую

очередь его стоит выполнять.

• Название/модуль/версия продукта (Component/Version) –

описание ПО, на котором можно выполнить тест-кейс.

• Предварительные условия (Precondition) – шаги, которые не-

обходимо выполнить перед началом тестирования.

• Шаги (Steps) – точная последовательность действий для выпол-

нения проверки. Шаги должны быть четкими и понятными.

• Ожидаемый результат (Expected result) – что мы получаем по-

сле выполнения шагов.

• Приложения (Attachments) – дополнительная информация,

которая поможет выполнить тест-кейс, например, скриншоты,

текстовые файлы и прочие файлы.

ТЕСТ-КЕЙС

Тест-кейс – это документ, который описывает наши тесты. Говорит, как их выполнить, при каких условиях и что должно получиться после выполнения тех шагов, которые

заложены в тест-кейсе, то есть каков ожидаемый результат.

Чек-лист – это список того, что проверяем.

Тест-кейс – документ, описывающий, как проверяем.

Список тест-кейсов является своего рода чек-листом

(если смотреть просто на названия тест-кейсов).

**24. Что такое тест-план и какие вопросы он должен охватывать?**

ТЕСТ-ПЛАН

Тест-план (Test Plan) – документ, описывающий весь объём работ по тестированию: описания объекта тестирования, стратегии, критериев начала и окончания тестирования, необходимое оборудование и знания, оценки рисков с вариантами их разрешения.

Тест-план призван ответить

на следующие вопросы:

• Что НАДО тестировать?

• Что БУДЕМ тестировать? (Тест-Аналитик).

• КАК будем тестировать? (Тест-Дизайнер).

• На каких уровнях будем проводить тестирование?

• Какие виды тестирования применим?

• Каким образом будем тестировать – руками или автотестами?

• КОГДА будем тестировать? Оценка трудозатрат и сроков.

• Какие РИСКИ возможны? Какие затраты времени, средств и труда они могут повлечь. Степень их влияния на исход проекта, прописать мероприятия по нейтрализации по-

следствий срабатывания рисков.

**25. Какие атрибуты должны быть в баг-репорте?**

Атрибуты баг-репорта:

1. Заголовок ошибки отвечает на три вопроса:

• Что произошло?

• Где появилась ошибка?

• Когда или при каких условиях найден дефект?

2. Описание ошибки

3. Номер версии (Version)

4. Автор баг репорта (обычно это Тестировщик)

5. Серьёзность (Severity)

6. Приоритет (Priority)

7. Начальные условия. В случае, если есть специфичные действия или шаги воспроизведения достаточно объёмные, то указываются начальные условия.

8. Шаги воспроизведения. Шаги, при которых повторяется найденная ошибка

9. Ожидаемый результат

10. Фактический результат

11. Вложения

**26. В чем разница между severity и priority в баг-репорте?**

СЕРЬЁЗНОСТЬ (Severity) – атрибут, характеризующий влияние дефекта на работоспособность ПО. Проставляется специалистом по тестированию.

ПРИОРИТЕТ (Priority) – это атрибут, указывающий на оче-

рёдность выполнения задачи или устранения дефекта. Про-

ставляется руководителем или менеджером проекта.

Приоритет отличается от Серьезности тем, что указывает, когда необходимо исправить ошибку.

ПРИМЕР бага с низкой серьёзностью, но высоким приоритетом:

Опечатка на главной странице бизнес-портала.

Системе не вредит, а репутации – да.

**27. Какие техники тест-дизайна ты знаешь? Опиши классы эквивалентности и граничные значения.**

Тест-дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест-кейсы) в соответствии с определёнными ранее критериями качества и целями тестирования.

К Л А С С Ы Э К В И В А Л Е Н Т Н О С Т И

И ГРАНИЧНЫЕ ЗНАЧЕНИЯ

КЛАССЫ ЭКВИВАЛЕНТНОСТИ – это техника, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений.

Любой тест, выполненный из одного и того же класса эквивалентности, приведет к точно такому же результату, как и выполнение всех остальных тестов из этого же класса.

ГРАНИЧНОЕ ЗНАЧЕНИЕ – это значение, которое находится на границе классов эквивалентности.

Техника анализа граничных значений – техника, проверяющая поведение системы либо отдельного модуля на граничных значениях входных данных.

Большинство ошибок возникает именно на границах между классами эквивалентности. Т.е. тестировщику, в первую очередь, важно проверить переходы на стыке

границ каждого класса.

**28. Что такое исследовательское и ad-hoc тестирование?**

Исследовательское тестирование (Exploratory Testing) – частично формализованный подход, в рамках которого тестировщик выполняет работу с продуктом по выбранному сценарию, который, в свою очередь, дорабатывается в процессе выполнения с целью более полного исследования приложения.

AD-HOC ТЕСТИРОВАНИЕ

Свободное (интуитивное) тестирование (Ad-hoc Testing) – полностью неформализованный подход, в котором не предполагается использования ни тест-кейсов, ни чек-листов, ни сценариев. Тестировщик опирается на свою интуицию и опыт для спонтанного выполнения с продуктом действий, которые, как он считает, могут обнаружить ошибку.

**29. Какие особенности тестирования мобильных приложений?**

Особенности тестирования мобильных приложений

• Тестирование пользовательского взаимодействия – удобства работы с приложнием: свайпы, тапы, скролы и т.п.

• Тестирование совместимости – установка на разные ОС, платформы, на разных моделях, проверка на разных разрешениях и т.п.

• Тестирование подключения – проверка на разных типах

подключения (wi-fi, мобильная сеть), переключение типов и

оффлайн работа.

• Тестирование производительности – утечка памяти, стабильность работы при большом количестве пользователей и т.п.

• Тестирование локализации – проверка размещения локализованного (переведённого) текста на экране, формата дат и т.д.

**30. Что такое API тестирование? Какие протоколы используются для веб-сервисов?**

Тестирование API – это тип тестирования, который сосредоточен на уровне бизнес-логики архитектуры ПО.

ПРОТОКОЛЫ РЕАЛИЗАЦИИ ВЕБ-СЕРВИСОВ:

XML-RPC (XML Remote Procedure Call) – протокол удаленного вызова процедур с использованием XML. Прародитель SOAP.

SOAP (Simple Object Access Protocol) – протокол обмена структурированными сообщениями в распределённой вычислительной среде. Протокол используется для обмена произвольными сообщениями в формате XML.

JSON-RPC (JSON Remote Procedure Call) – это текстовый формат обмена данными, данные передаются в формате JSON. (JavaScript Object Notation). Структура JSON состоит из набора пар ключ – значения.

REST (Representational State Transfer, передача самоописываемого состояния) – архитектурный стиль взаимодействия компьютерных систем в сети, основанный на методах протокола HTTP. Другими словами, REST – это набор правил о том, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать.