1 Что такое тестирование?

Тестирование програмного обеспечения — это процесс проверки разрабатываемого или готового продукта на уровень его качества.

Тестирование — это непрерывный процесс в ходе которого над продуктом проводится набор проверок (тестов), заранее подготовленных или основанных, например, на опыте, знаниях проверяющего.

2 История развития тестирования как науки. Основные этапы?

3 Технические навыки тестировщика — перечислить

4 Личностные качества тестировщика — перечислить

5 Тестирование QC и QA

6 Верификация и валидация

7 Перечислите основные роли на проекте по разработке ПО

1. Заказчик;
2. Руководитель проекта;
3. Аналитики;
4. Разработчики;
5. Тестирощики;
6. Дизайнеры;
7. Поддержка.

8 Перечислите этапы жизненного цикла разработки ПО

1. Планирование;
2. Анализ;
3. Проектирование;
4. Разработка;
5. Внедрение;
6. Поддержка;
7. Вывод из эксплуатации.

9 Для чего требуется проводить два этапа оценки трудозатрат на анализ, разработку и тестирование?

10 Для чего проводится регрессионное тестирование?

В приложениях очень часто один модуль связан с десятками других модулей. При новой доработке часть уже существующих связей могут быть случайным образом изменены, что приведет к появлению багов. Простыми словами проверяем, что новые доработки не сломали то, что уже хорошо работает.

11 Для чего проводится интеграционное тестирование?

Интеграционное тестирование (integration testing , component integration testing) - направлено на проверку взаимодействия между несколькими частями приложения (каждая из которых, в свою очередь, проверена отдельно на стадии модульного тестирования). Целью данного тестирования является проверка «стыков» между отдельными модулями.

12 Назовите методологию разработки, которая, на ваш взгляд, лучше подходит для большей части проектов?

13 Какие события вам известны из Scrum методологии?

Пять церемоний scrum — это планирование спринта, ежедневный scrum/standup, обзор спринта, ретроспектива спринта и уточнение бэклога.

14 Как качество составленных требований влияет на разработку и тестирование ПО?

Оно позволяет:

1. сократить общие сроки проекта за счет меньшего количества ошибок, повторных работ, тестирования;
2. снизить риски срыва проекта по причине переделки большой части продукта из-за противоречивых и несогласованных между собой требований;
3. снизить уровень хаоса на проекте;
4. повысить качество кода, за счет более четкой и понятной постановки задачи;
5. повысить удовлетворенность конечного заказчика;
6. повысить удовлетворенность и производительность проектной команды.

15 Классификация видов тестирования по уровню доступа к коду и архитектуре приложения

Чёрный ящик(black box testing, closed box testing, specification-based testing) — Метод тестирования, при котором у тестировщика нет доступа к внутренней структуре и коду приложения, недостаточно знаний для их понимания или он сознательно не обращается к ним в процессе тестирования. В рамках тестирования по методу черного ящика основной информацией для создания тест-кейсов выступает документация.

Белый ящик (white box testing , open box testing, clear box testing, glass box testing) — у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного.

Серый ящик(gray box testing) — совокупность методов белого ящика и черного ящика, состоящая в том, что к одной части кода и архитектуре у тестировщика доступ есть, а к другой части - нет. При тестировании в реальных условиях используется чаще всего.

	Преимущества	**Недостатки**
Метод белого ящика	• Показывает скрытые проблемы и упрощает их диагностику. • Допускает достаточно простую автоматизацию тест-кейсов и их выполнение на ранних стадиях развития проекта. • Обладает развитой системой метрик, сбор и анализ которых легко автоматизируется. • Стимулирует разработчиков к написанию качественного кода. • Многие техники этого метода являются хорошо зарекомендовавшими решениями	• Недоступен тестировщикам, у которых нет достаточных знаний в области программирования. • Тестирование сфокусировано на реализованной функциональности, что повышает вероятность пропуска нереализованных требований. • Поведение приложения исследуется в отрыве от реальной среды выполнения и не учитывает её влияние.
Метод черного ящика	• Тестировщик не обязан обладать (глубокими) знаниями в области программирования. • Поведение приложения исследуется в контексте реальной среды выполнения и учитывает её влияние. • Поведение приложения исследуется в контексте реальных пользовательских сценариев. • Тест-кейсы можно создавать уже на стадии появления стабильных требований. • Процесс создания тест-кейсов позволяет выявить дефекты в требованиях. • Допускает создание тест-кейсов, которые можно многократно использовать на разных проектах.	• Возможно повторение части тест-кейсов, уже выполненных разработчиками. • Высока вероятность того, что часть возможных вариантов поведения приложения будет не протестирована • Для разработки высокоэффективных тест-кейсов необходима качественная документация. • Диагностика обнаруженных дефектов более сложна в сравнении с техниками метода белого ящика. • В связи с широким выбором техник и подходов затрудняется планирование и оценка трудозатрат. • В случае автоматизации могут потребоваться сложные дорогостоящие инструментальные средства.
Метод серого ящика	Содержит в себе и преимущества, и недостатки от первых двух методов.	Содержит в себе и преимущества, и недостатки от первых двух методов.

16 Классификация видов тестирования по степи важности тестируемого функционала

Дымовое тестирование (Smoke-test) — проверка самой главной, самой важной, самой ключевой функциональности, неработоспособность которой делает бессмысленной саму идею использования приложения или отдельной его части. Является самым важным видом тестирования в данной классификации.

Тестирование критического пути (Critical path test) - направлено на исследование функциональности, используемой типичными пользователями в типичной повседневной деятельности. Тестируется тот функционал, который используется большинством пользователей чаще всего. Данный вид тестирования проводится после успешного прохождения smoke-теста.

Расширенное тестирование (extended test) - направлено на исследование всей заявленной в требованиях функциональности - даже той, которая низ-

ко проранжирована по степени важности. Также учитывается, какая функциональность является более важной, а какая — менее важной. Но при наличии достаточного количества времени и иных ресурсов тест-кейсы этого уровня могут затронуть могут затрагивать требования самого низкого приоритета. В рамках данного тестирования проверяют нетипичные, маловероятные сценарии использования функционала приложения.

17 Регрессионное тестирование. Определение. Когда применяется?

Классификация по целям и задачам

Регрессионное тестирование

- тестирование, направленное на проверку того факта, что в ранее работоспособной функциональности не появились ошибки, вызванные изменениями в приложении или среде его функционирования. Фредерик Брукс в своей книге «Мифический человеко-месяц» писал: «Фундаментальная проблема при сопровождении программ состоит в том, что исправление одной ошибки с большой вероятностью (20–50 %) влечёт появление новой». Потому регрессионное тестирование является неотъемлемым инструментом обеспечения качества и активно используется практически в любом проекте.

РЕГРЕССИОННОЕ ТЕСТИРОВАНИЕ

Когда проводится:
- Все доработки были протестированы.
- Продукт готов для взаимодействия с пользователями.

Зачем: В приложениях очень часто один модуль связан с десятками других модулей. При новой доработке часть уже существующих связей могут быть случайным образом изменены, что приведет к появлению багов. Простыми словами проверяем, что новые доработки не сломали то, что уже хорошо работает

Кто делает: Команда тестирования или часть команды тестирования.

Краткое описание процесса:
1. Проверяем, что все доработки готовы, багов по ним нет
2. Формируется тест-план, для описания регрессионного тестирования
3. Выполняется регрессионное тестирование
   a. Если в регрессионном тестировании нашли баг, то тестирование останавливается до фикса бага. После чего регресс начинается заново
   b. Багов не обнаружено, регрессионное тестирование завершается
4. Формируется отчет о проведенном тестировании. Приложение готово к работе с пользователями(Или же переходит в бизнес-тестирование/приемочное тестирование/beta-тестирование)

18 Позитивное и негативное тистирование. В чём разница в подходах? Привести пример негативного и позитивного тестировани.

Позитивное тестирование (**positiv**e testing) - подход к тестированию, в котором проверяются сценарии, где пользователь взаимодействует с приложением так, как это задумывалось, без каких либо отклонений, ввода неверных данных, нестандартного поведения.

Пример:
Мы тестируем ДЗ по теории графов. В этом ДЗ выбираем метод работы с консолью, ввод списка смежности через консоль. Тест будет заключаться в воде различных списков смежности, соответствующих условиям ввода, которые запрашивает ДЗ.

Негативное тестирование (**negative** testing) - подход к тестированию, в котором проверяются сценарии, когда с приложением выполняются (некорректные) операции и/или используются данные, потенциально приводящие к ошибкам(Например: деление на ноль).

Мы тестируем ДЗ по теории графов. В этом ДЗ выбираем метод работы с консолью, ввод списка смежности через консоль. Тест будет заключаться в воде различных списков смежности, не соответствующих условиям ввода (например

связь с несуествующей вершиной, для взвешенного графа не будем указывать вес связи), которые запрашивает ДЗ.

19 Повторное тестирование (Re-test). Определение. Когда применяется.

Повторное тестирование (Re-test) - выполнение тест-кейсов, которые ранее обнаружили дефекты, с целью подтверждения устранения дефектов. Фактически этот вид тестирования сводится к действиям на финальной стадии жизненного цикла отчёта о дефекте, направленным на то, чтобы перевести дефект в состояние «проверен» и «закрыт».

20 Классификация видов тестирования по уровню тестирования

Модульное (компонентное) тестирование (unit testing, module testing, component testing) - направлено на проверку отдельных небольших частей приложения, которые можно исследовать изолированно от других подобных частей. При выполнении данного тестирования могут проверяться отдельные функции или методы классов, сами классы, взаимодействие классов, небольшие библиотеки, отдельные части приложения. Часто данный вид тестирования реализуется с использованием дополнительных инструментов и средств автоматизации тестирования, значительно упрощающих и ускоряющих разработку соответствующих тест-кейсов.

Интеграционное тестирование (integration testing , component integration testing) - направлено на проверку взаимодействия между несколькими частями приложения (каждая из которых, в свою очередь, проверена отдельно на стадии модульного тестирования). Целью данного тестирования является проверка «стыков» между отдельными модулями.

Системное тестирование (system testing) - направлено на проверку всего приложения как единого целого, собранного из частей, проверенных на двух предыдущих стадиях. Здесь не только выявляются дефекты «на стыках» компонентов, но и появляется возможность полноценно взаимодействовать с приложением с точки зрения конечного пользователя, применяя множество других видов тестирования.

21 Альфа-тестирование vs Бета-тестирование: определения и отличия.

- **Альфа-тестирование (alpha testing)** выполняется командой разработки с возможным частичным привлечением конечных пользователей. Может являться формой внутреннего приемочного тестирования. Основная суть вида: продукт уже можно показывать внешним пользователям, но он ещё достаточно «сырой», потому основное тестирование выполняется организацией-разработчиком.
- **Бета-тестирование (beta testing)** выполняется вне команды разработки с активным привлечением конечных пользователей/заказчиков. Может являться формой внешнего приемочного тестирования. Основная суть вида: продукт уже можно открыто показывать внешним пользователям, он уже достаточно стабилен, но проблемы всё ещё могут быть и для их выявления нужна обратная связь от реальных пользователей.

22 Тестирование локализации. Определение, применение.

Тестирование локализации - тестирование, направленное на проверку корректности и качества адаптации продукта к использованию на том или ином языке с учётом национальных и культурных особенностей. Главная цель такого тестирования - проверить качество адаптации продукта.

23 Функциональное теститрование vs нефункциональное тестирование: определения и отличия. Примеры.

Классификация по целям и задачам

Функциональное тестирование	Нефункциональное тестирование
Работают ли функции приложения правильно?	Удобный ли интерфейс? Приложение хорошо справляется с высокой нагрузкой? Приложение безопасно? Совместимо ли приложение с мобильными устройствами?

- **Функциональное** тестирование (functional testing) рассматривает заранее указанное поведение и основывается на анализе спецификации компонента или системы в целом, т.е. проверяется корректность работы *функциональности* приложения.

> Нефункциональное тестирование (non-functional testing) — тестирование атрибутов компонента или системы, не относящихся к функциональности.

24 Классификация по уровню архитектуры приложения. Назовите виды данной классификации. Определения данных видов.

Тестирование уровня представления (presentation tier testing) -

сконцентрировано на той части приложения, которая отвечает за взаимодействие с «внешним миром» (как пользователями, так и другими приложениями). Здесь исследуются вопросы удобства использования, скорости отклика интерфейса, совместимости с браузерами, корректности работы интерфейсов.

Тестирование уровня бизнес-логики (business logic tier testing) -

отвечает за проверку основного набора функций приложения и строится на базе ключевых требований к приложению, бизнес-правил и общей проверки функциональности.

Тестирование уровня данных (data tier testing) - сконцентрировано на той части приложения, которая отвечает за хранение и некоторую обработку данных (чаще всего — в базе данных или ином хранилище). Здесь особый интерес представляет тестирование данных, проверка соблюдения бизнес-правил, тестирование производительности.

25 Дайте определение терминам Требование и Тестирование требований

Требование – описание того, какие функции и с соблюдением каких условий должно выполнять приложение в процессе решения полезной для пользователя задачи.

Тестирование требований (тестирование спецификаций) – процесс поиска и обнаружения ошибок, неточностей, двусмысленности, невыполнимости, неполноты и противоречий в документах, описывающих требования к программному продукту.

26 Назовите цель и важность тестирования.

Цель тестирования требований – выявить проблемы и устранить противоречия в них еще до начала разработки и избежать дорогостоящих доработок и изменений на поздних этапах проектов.

Тестирование требований позволяет:

1. сократить общие сроки проекта за счет меньшего количества ошибок, повторных работ, тестирования;
2. снизить риски срыва проекта по причине переделки большой части продукта из-за противоречивых и несогласованных между собой требований;
3. снизить уровень хаоса на проекте;
4. повысить качество кода, за счет более четкой и понятной постановки задачи;
5. повысить удовлетворенность конечного заказчика;
6. повысить удовлетворенность и производительность проектной команды.

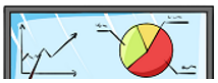
Тестирование требований позволяет:

1. сократить общие сроки проекта за счет меньшего количества ошибок, повторных работ, тестирования;
2. снизить риски срыва проекта по причине переделки большой части продукта из-за противоречивых и несогласованных между собой требований;
3. снизить уровень хаоса на проекте;
4. повысить качество кода, за счет более четкой и понятной постановки задачи;
5. повысить удовлетворенность конечного заказчика;
6. повысить удовлетворенность и производительность проектной команды.

27 Перечислите все уровни требований. Дайте определение каждому уровню требований. Приведите примеры.

Бизнес-требование – высокоуровневая бизнес-цель организации или заказчиков системы.

- ✓ *Нужен инструмент, в реальном времени отображающий наиболее выгодный курс покупки и продажи валюты.*
- ✓ *Сократить время обработки заказа на 50% (цель) – система должна предоставить интерфейс, упрощающий создание заказа (концепция).*

Пользовательские требования – задача, которую определенные классы пользователей должны иметь возможность выполнять в системе, или требуемый атрибут продукта/проекта.

- ✓ *Пользователь должен иметь возможность добавить объект в избранное (функциональность).*
- ✓ *Администратор должен иметь возможность просматривать список всех пользователей, работающих в данный.*

Функциональное требование – описание требуемого поведения системы в определенных условиях.

- ✓ *Система должна автоматически выполнять резервное копирование данных ежедневно в указанный момент времени.*
- ✓ *Приложение не должно выгружать из оперативной памяти фоновые документы в течение 30 минут с момента выполнения с ними последней операции.*

Нефункциональные требования – описание свойств и особенностей, ограничений, которыми должна обладать система.

- ✓ *Система должна быть способна обслуживать необходимое количество пользователей без снижения производительности.*
- ✓ *Система должна иметь возможность увеличивать или уменьшать масштаб по мере необходимости*

28 Перечислите свойства качественных требований

1. Обязательность
2. Актуальность
3. Атомарность
4. Завершенность и полнота
5. Выполнимость
6. Проранжированность по важности, стабильности, срочности
7. Непротиворечивость
8. Недвусмысленность
9. Прослеживаемость
10. Модифицируемость
11. Корректность и проверяемость