**Список вопросов по теме №5**

**По документу Шпаргалка\_начинающего\_тестировщика**

1. **Что такое тестирование программного обеспечения? Каковы его основные цели?** *(См. стр. 5)*

Тестирование (Software Testing) – проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом.

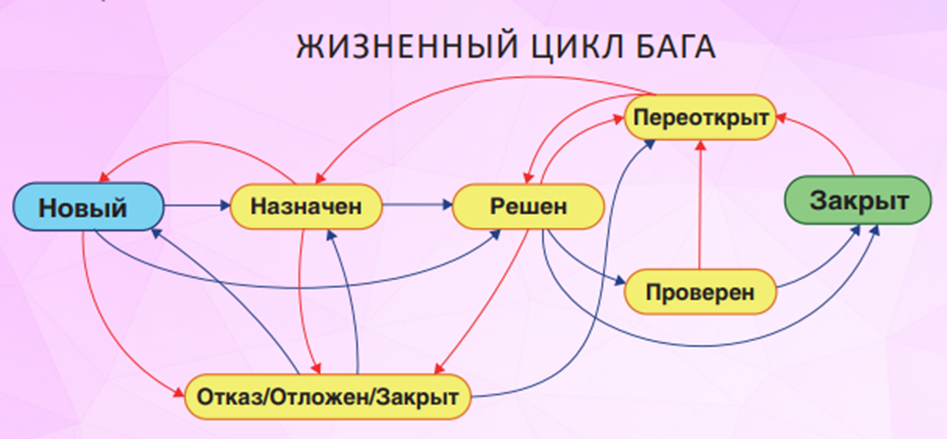
Цели тестирования: • Выявление дефектов до того, как их обнаружат пользователи. • Предоставление актуальной информации о состоянии продукта на данный момент. • Проверка на соответствие ПО всем заявленным требованиям.

1. **Объясни принципы тестирования, такие как "Тестирование демонстрирует наличие дефектов" и "Исчерпывающее тестирование невозможно".** *(См. стр. 5)*

Принципы тестирования: • Тестирование демонстрирует наличие дефектов. Тестирование только снижает вероятность наличия дефектов, которые находятся в ПО, но не гарантирует их отсутствия. • Исчерпывающее тестирование невозможно. Полное тестирование с использованием всех входных комбинаций данных, результатов и предусловий физически невыполнимо. • Раннее тестирование. Следует начинать тестирование на ранних стадиях жизненного цикла разработки ПО, чтобы найти дефекты как можно раньше. • Скопление дефектов. Большая часть дефектов находится в ограниченном количестве модулей. • Эффект (парадокс) пестицида. Если повторять те же тестовые сценарии снова и снова, в какой-то момент этот набор тестов перестанет выявлять новые дефекты. Суть его в том, что если вы долго проводите одни и те же проверки, скорее всего новых багов вы не найдете. Именно поэтому периодически нужно «встряхивать» тестовую базу, ревьюить её новыми сотрудниками, проводить исследовательское тестирование. • Тестирование зависит от контекста. Например, программное обеспечение, в котором критически важна безопасность, тестируется иначе, чем новостной портал. • Заблуждение об отсутствии ошибок. Отсутствие найденных дефектов при тестировании не всегда означает готовность продукта к релизу. Система должна быть удобна пользователю в использовании и удовлетворять его ожиданиям и потребностям.

1. **Что такое баг в тестировании? Опиши жизненный цикл бага.** *(См. стр. 7)*

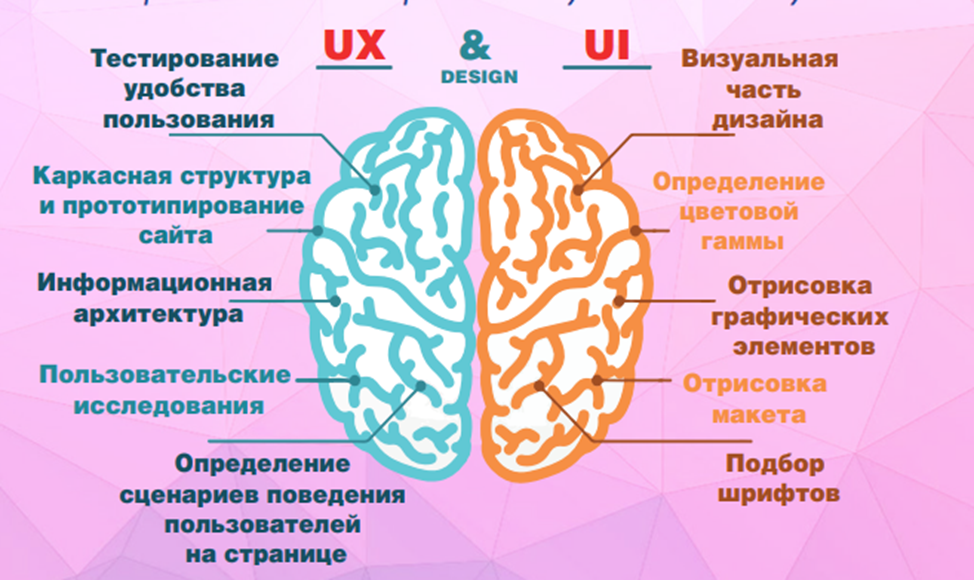
БАГ (bug, дефект) – отклонение фактического результата от ожидаемого. Найденные баги оформляются в баг-репорты (стр. 36). Ожидаемый результат (Expected result) – описание того, как именно должна работать система в соответствии с документацией. Фактический результат (Actual result) – это тот результат, который получает тестировщик во время тестирования. То есть то, как система работает на самом деле.



Новый (New) – Впервые найденный баг, занесённый в систему. • Отказ (Rejected) – Баг отклонён. Причины: некачественное описание или такой дефект уже существует, невозможность воспроизвести баг. • Закрыт (Closed). – Баг перестал быть актуальным. • Назначен (Assigned) – Дефект просмотрен и открыт, то есть признан для исправления. • Решен (Fixed) – Дефект исправили, и он в этом состоянии требует перепроверки тестировщиком. • Переоткрыт (Re-opened) – Если дефект не исправлен или исправлен не полностью.

1. **В чем разница между UX и UI дизайном?** *(См. стр. 8)*

UX дизайнер планирует то, как вы будете взаимодействовать с интерфейсом, и какие шаги вам нужно предпринять, чтобы сделать что-то. А UI дизайнер придумывает, как каждый из этих шагов будет выглядеть.



1. **Опиши этапы жизненного цикла разработки ПО (SDLC).** *(См. стр. 9)*

ЖИЗНЕННЫЙ ЦИКЛ РАЗРАБОТКИ: • Анализ требований отвечает на вопрос «Какие проблемы требуют решений?». • Планирование отвечает на вопрос «Что мы хотим сделать?». • Проектирование и дизайн отвечает на вопрос «Как мы добьемся наших целей?». • Разработка ПО регулирует процесс создания продукта. • Тестирование регулирует обеспечение качественной работы продукта. • Развертывание регулирует использование финального продукта.

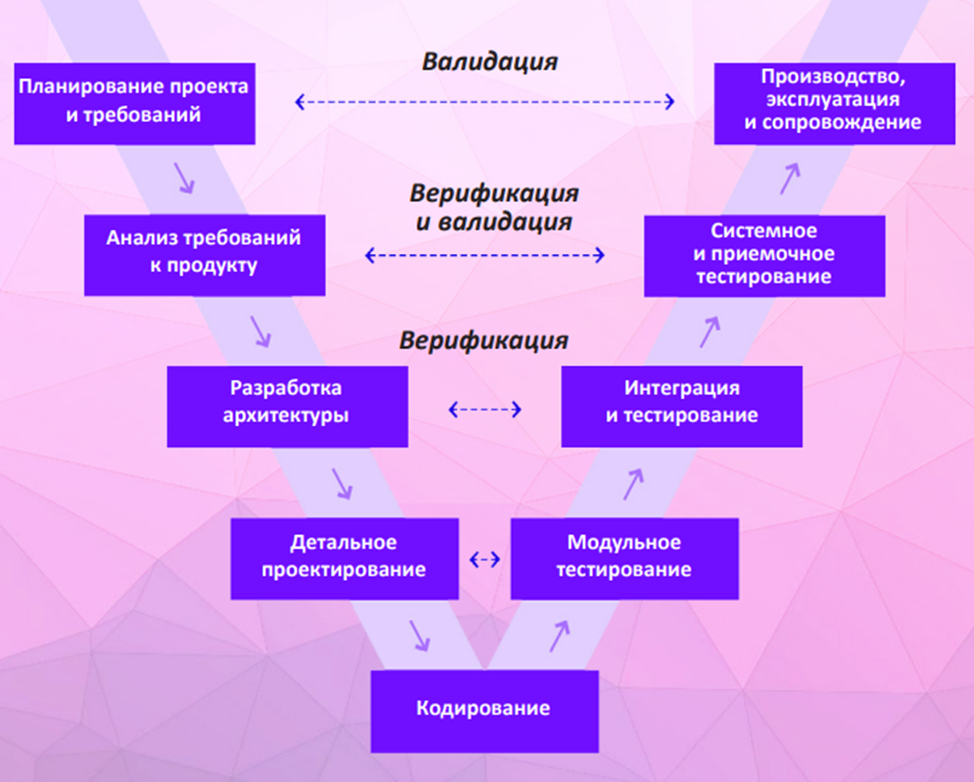


1. **Какие модели разработки ПО ты знаешь? Сравни водопадную и V-образную модели.** \*(См. стр. 10-11)\*

ВОДОПАДНАЯ МОДЕЛЬ Waterfall Model – подразумевает последовательное прохождение стадий, каждая из которых должна завершиться полностью до начала следующей. Особенности: • Разработка проходит быстро. • Стоимость и сроки заранее определены. • Хороший результат только в проектах с четко и заранее определенными требованиями. • Нет возможности сделать шаг назад.



V-Model – Унаследовала структуру «шаг за шагом» от водопадной модели. Применима к системам, которым особенно важно бесперебойное функционирование. Например, прикладные программы в клиниках для наблюдения за пациентами, интегрированное ПО для механизмов управления аварийными подушками безопасности в транспортных средствах и т.д. V-образную модель ещё называют разработкой через тестирование. Особенности: • Направлена на тщательную проверку и тестирование продукта с ранних стадий проектирования. • Стадия тестирования проводится одновременно с соответствующей стадией разработки. • Количество ошибок в архитектуре ПО сводится к минимуму. • Если при разработке архитектуры была допущена ошибка, то вернуться и исправить её будет стоить дорого, как и в «водопаде».



1. **Что такое Agile? Каковы его основные ценности?** *(См. стр. 14)*

Методология Agile – это набор практик, целью которых является оперативная реакция на изменения в ходе рабочего процесса. Такие подходы помогают командам быстро реагировать на обратную связь от клиентов и заказчиков, тем самым постоянно улучшая производимый продукт.

Ценности Agile • Люди важнее процессов и стандартных инструментов; • Создание продукта важнее, чем подбор и согласование документации; • Услышать позицию клиента и сотрудничать с ним превыше контракта; • Вносить коррективы важнее, чем изначально разработанная стратегия.

1. **В чем разница между Scrum и Kanban?** *(См. стр. 15)*

Scrum vs Kanban • Основная разница между Scrum и Канбан – в длине итераций. В Scrum итерации – в среднем 2 недели, в Kanban задачи программисту можно подкидывать хоть каждый день. • Если в Scrum цель команды – закончить спринт, то в Kanban – задачу.

1. **Что такое User Story? Как она должна быть оформлена?** *(См. стр. 16)*

User Story  –  это приём записи требований, который помогает понять нужду клиента и после обсуждения выбрать, описать и утвердить то решение, которое удовлетворит эту нужду.

История должна быть написана по такому образцу: Как , я , (Как пациент, я хочу созваниваться с врачами по видеосвязи, чтобы иметь возможность обсудить вопросы здоровья). User Story можно оценить по критериям «INVEST»: Independent – независимая от других историй, то есть истории могут быть реализованы в любом порядке. Negotiable – обсуждаемая, отражает суть, а не детали; не содержит конкретных шагов реализации. Valuable – ценная для клиентов, бизнеса и стейкхолдеров. Estimable – оцениваемая по сложности и трудозатратам. Small – компактная, может быть сделана за одну итерацию. Testable – тестируемая, имеет критерии приемки.

1. **Опиши этапы жизненного цикла тестирования (STLC).** *(См. стр. 17)*

**

• Общее планирование и анализ требований. Необходимо получить ответы на следующие вопросы: что нам предстоит тестировать; как много будет работы; какие есть сложности; всё ли необходимое у нас есть и т.п. • Уточнение критериев приёмки. Позволяет сформулировать метрики и признаки необходимости начала (приостановки/возобновления/завершения) тестирования. • Уточнение стратегии тестирования. Рассматриваются и уточняются те части стратегии тестирования, которые актуальны для текущей итерации. • Разработка тест-кейсов. Посвящена разработке, пересмотру и уточнению тест-кейсов. • Выполнение тест-кейсов и фиксация найденных дефектов. Дефекты исправляются сразу по факту их обнаружения в процессе выполнения тест-кейсов. • Анализ результатов тестирования и отчётность. Создается отчет о результатах тестирования. QA-команда обсуждает и анализирует баги, делает выводы из возникших проблем, чтобы избежать подобных проблем в будущем.

1. **Какие виды тестирования существуют? Приведи примеры.** \*(См. стр. 18-27)\*

ВИДЫ ТЕСТИРОВАНИЯ упрощенная классификация ПО ЗАПУСКУ КОДА НА ИСПОЛНЕНИЕ: • Статическое тестирование – без запуска. • Динамическое тестирование – с запуском. ПО ДОСТУПУ К КОДУ И АРХИТЕКТУРЕ ПРИЛОЖЕНИЯ: • Метод белого ящика – доступ к коду есть. • Метод чёрного ящика – доступа к коду нет. • Метод серого ящика – к части кода доступ есть, к части – нет. ПО СТЕПЕНИ АВТОМАТИЗАЦИИ: • Ручное тестирование. • Автоматизированное тестирование. ПО УРОВНЮ ДЕТАЛИЗАЦИИ ПРИЛОЖЕНИЯ: • Модульное тестирование. • Интеграционное тестирование – проверка взаимодействия между модулями. • Системное тестирование – приложение проверяется как единое целое. ПО СТЕПЕНИ ВАЖНОСТИ ТЕСТИРУЕМЫХ ФУНКЦИЙ: • Дымовое тестирование – проверка самой важной, самой ключевой функциональности приложения. • Тестирование критического пути – проверка функциональности, используемой типичными пользователями в типичной повседневной деятельности. • Расширенное тестирование – проверка всей (остальной) функциональности, заявленной в требованиях. ПО ПРИНЦИПАМ РАБОТЫ С ПРИЛОЖЕНИЕМ: • Позитивное тестирование – это тестирование с применением сценариев, которые соответствуют нормальному (штатному, ожидаемому) поведению системы. С его помощью мы можем определить, что система делает то, для чего и была создана. • Негативное тестирование – тестирование направлено на проверку устойчивости системы к различным воздействиям, валидации неверных данных, обработку исключительных ситуаций.

1. **В чем разница между статическим и динамическим тестированием?** *(См. стр. 19)*

СТАТИЧЕСКОЕ ТЕСТИРОВАНИЕ – при статическом тестировании код не выполняется (программа не запускается). Проверяется: • Документы (требования, тест-кейсы, описания архитектуры приложения, схемы баз данных и т.д.). • Графические прототипы (дизайн). • Код приложения (выполняется программистами в рамках аудита кода (code review), являющегося специфической вариацией взаимного просмотра в применении к исходному коду). • Параметры (настройки) среды исполнения приложения. Статическое тестирование начинается на ранних этапах жизненного цикла ПО и является, соответственно, частью процесса верификации.

ДИНАМИЧЕСКОЕ ТЕСТИРОВАНИЕ – тестирование с запуском кода на исполнение, как всего приложения целиком (системное тестирование), так нескольких взаимосвязанных частей (интеграционное тестирование), отдельных частей (модульное тестирование) и даже отдельные участки кода. Динамическое тестирование включает в себя тестирование ПО в режиме реального времени путем предоставления входных данных и изучения результата поведения ПО. Проверка осуществляется с помощью заранее подготовленного набора тестов. Является частью процесса валидации программного обеспечения.

1. **Что такое валидация и верификация в тестировании?** *(См. стр. 19)*

ВЕРИФИКАЦИЯ – процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки сформированным в начале этапа условиям.

Валидация – нет. Заказчик хотел коня. ВАЛИДАЦИЯ – проверка соответствия ПО требованиям пользователя. Программа может на 100 % соответсвовать спецификации, но при этом выполнять совершенно не то, что хотел пользователь/заказчик.

ПРИМЕР. Релиз – ОСЁЛ. Заказчик по ТЗ хотел: непарнокопытное животное с седлом, 4 ногами и умеющее скакать. Верификация пройдена.

1. **Какие бывают требования к ПО? Какими свойствами они должны обладать?** *(См. стр. 20)*

Требования к ПО – это спецификация того, что должно быть реализовано. В них описано поведение системы, свойства системы или ее атрибуты. Требования к ПО состоят из трех уровней: •Бизнес-требования.

Цели, которые организация намерена достичь с ее помощью разрабатываемого продукта. •Пользовательские требования.

Цели и задачи, которые пользователи должны иметь возможность выполнять с помощью ПО. •Функциональные требования определяют, каким должно быть поведение продукта в тех или иных условиях.

Свойства хорошего документа

• Требования должны быть полными, правильно и в полной мере описывать функцию, которую необходимо реализовать. •Однозначность. Одинаковое восприятие документа всей командой. •Непротиворечивость. Не должно быть противоречивых требований, конфликтующих между собой. •Необходимость. Требования должны отражать функциональности, действительно необходимые для пользователя, для удовлетворения пользователей. •Выполнимость. Насколько требования возможно реализовать. • Тестируемость. Возможность проверить все прописанные требования после их реализации. Улучшить качество продукта. •Модифицируемость. Это свойство характеризует простоту внесения изменений в отдельные требования и в набор требований. •Атомарность. Требование описывает только одну ситуацию. •Прослеживаемость. •Актуальность

1. **Объясни методы тестирования: черный, белый и серый ящик.** *(См. стр. 21)*

МЕТОД БЕЛОГО ЯЩИКА у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного. Некоторые авторы склонны связывать этот метод со статическим тестированием, но ничто не мешает тестировщику запустить код на выполнение и при этом периодически обращаться к самому коду. А модульное тестирование и вовсе предполагает запуск кода на исполнение и при этом работу именно с кодом, а не с «приложением целиком».

МЕТОД ЧЁРНОГО ЯЩИКА у тестировщика либо нет доступа к внутренней структуре и коду приложения, либо недостаточно знаний для их понимания, либо он сознательно не обращается к ним в процессе тестирования. Большинство видов тестирования работают по методу чёрного ящика, идею которого в альтернативном определении можно сформулировать так: тестировщик оказывает на приложение воздействия (и проверяет реакцию) тем же способом, каким при реальной эксплуатации приложения на него воздействовали бы пользователи или другие приложения. В рамках тестирования по методу чёрного ящика основной информацией для создания тест-кейсов является документация. МЕТОД СЕРОГО ЯЩИКА комбинация методов белого ящика и чёрного ящика, состоящая в том, что к части кода и архитектуры у тестировщика доступ есть, а к части — нет

1. **Что такое позитивное и негативное тестирование?** *(См. стр. 26)*

Позитивное тестирование – это тестирование с применением сценариев, которые соответствуют нормальному (штатному, ожидаемому) поведению системы. С его помощью мы можем определить, что система делает то, для чего и была создана. Негативное тестирование – это тестирование в рамках которого применяются сценарии, которые соответствуют внештатному поведению тестируемой системы. Это могут быть исключительные ситуации или неверные данные.

1. **В чем разница между фронтенд и бэкенд тестированием?** *(См. стр. 26)*

Frontend-тестирование • Всегда выполняется в графическом интерфейсе. • Тестер должен быть осведомлен о требованиях бизнеса, а также об использовании инструментов сред автоматизации. • GUI используется для выполнения тестирования. • Не требуется хранения информации в базе данных. • Важно проверить общую функциональность приложения.

Backend-тестирование • Включает тестирование баз данных и бизнес-логики. • Тестер должен иметь большой опыт работы с базами данных и концепциями языка структурированных запросов (SQL). • Важно для проверки на наличие взаимоблокировок, повреждения данных, потери данных и т.д. • Широко используется тестирования базы данных: SQL-тестирование и API-тестирование.

1. **Что такое функциональное и нефункциональное тестирование?** *(См. стр. 27)*

Функциональное тестирование (functional testing) – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Нефункциональное тестирование (non-functional testing) – анализ атрибутов качества компонента или системы, не относящихся к функциональности, то есть проверка «как работает система».

1. **Чем отличается дымовое тестирование от регрессионного?** *(См. стр. 28)*

• РЕГРЕССИОННОЕ ТЕСТИРОВАНИЕ (Regression Testing). Не задело ли изменение в коде другой функуионал продукта.

• ДЫМОВОЕ ТЕСТИРОВАНИЕ (Smoke Testing). Проверка критически важных функций и стабильности системы в целом перед более тщательное тестированием.

1. **Что такое санитарное тестирование?** *(См. стр. 28)*

Санитарное тестированиеориентировано на глубинное исследование определенной функции, а дымовое – на тестирование большого количества функционала за самые короткие сроки.

1. **Какие виды тестовой документации ты знаешь?** *(См. стр. 31)*

ВИДЫ ТЕСТОВОЙ ДОКУМЕНТАЦИИ • 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 компонента 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 программным продуктом для достижения конкретной цели

1. **Что такое чек-лист и как он используется в тестировании?** *(См. стр. 31)*

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 отследить ход работ и корректировать процесс на любом этапе. Четкий алгоритм облегчает проверку задач.

1. **Опиши структуру тест-кейса. Чем он отличается от чек-листа?** *(См. стр. 32)*

• ID – уникальный номер. Обычно проставляется автоматически 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 основе 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 необходимо выполнить перед началом тестирования. • Шаги (Steps) – точная последовательность действий для выполнения проверки. Шаги должны быть четкими и понятными. • Ожидаемый результат (Expected result) – что мы получаем после выполнения шагов. • Приложения (Attachments) – дополнительная информация, которая поможет выполнить тест-кейс, например, скриншоты, текстовые файлы и прочие файлы.



1. **Что такое тест-план и какие вопросы он должен охватывать?** *(См. стр. 34)*

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 последствий срабатывания рисков.

1. **Какие атрибуты должны быть в баг-репорте?** *(См. стр. 36)*

БАГ РЕПОРТ (Bug Report – «отчёт об ошибке») – это технический документ, поэтому он создается по определенным правилам. Формат баг-репорта меняется в зависимости от компании, но костяк и суть всегда сохраняются. (Что такое баг, стр. 7) 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

1. **В чем разница между severity и priority в баг-репорте?** *(См. стр. 36)*

СЕРЬЁЗНОСТЬ (Severity) – атрибут, характеризующий влияние дефекта на работоспособность ПО. Проставляется специалистом по тестированию. ПРИОРИТЕТ (Priority) – это атрибут, указывающий на очерёдность выполнения задачи или устранения дефекта. Проставляется руководителем или менеджером проекта.

1. **Какие техники тест-дизайна ты знаешь? Опиши классы эквивалентности и граничные значения.** *(См. стр. 37)*

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.

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

ПРИМЕР. На картинке ниже представлено четыре возрастные группы: с рождения до 15 лет, от 15 до 25 лет, от 25 до 60 лет и от 60 до 99 лет. Сверху отображено эквивалентное разбиение (проверки: 8 лет, 19, 42 и 79 – можно выбрать любые другие возраста в пределах группы). Снизу – граничные значения

1. **Что такое исследовательское и ad-hoc тестирование?** *(См. стр. 40)*

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.

1. **Какие особенности тестирования мобильных приложений?** \*(См. стр. 41-43)\*

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.

1. **Что такое API тестирование? Какие протоколы используются для веб-сервисов?** *(См. стр. 52)*

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

Веб-сервисы – это способ связи (обмен данными) между двумя электронными устройствами (приложениями) по сети. 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.