Для подготовки:

Самопрезентация: про себя + краткий обзор резюме

Подготовить вопросы для HR + для команды

Теория:

Модели разработки ПО:
1. Водопадная
2. V-модель
3. Итерационная

Тестирование - это процесс анализа программного продукта, комплекс определенных мероприятий с целью выявления и последующего устранения дефектов, проверки его соответствия необходимым требованиям, тем самым повышая его качество.

Цель тестирования: повысить вероятность того, что приложение будет работать при любых обстоятельствах и условиях за счёт очищения программы от ошибок.

Принципы тестирования:
1. Тестирование показывает наличие, а не отсутствие дефектов
2. Заблуждение об отсутствии ошибок.
2. Исчерпывающее тестирование невозможно
3. Раннее тестирование
4. Скопление дефектов
5. Парадокс пестицидов
6. Тестирование зависит от контекста

Testing<QC<QA:
- Тестирование - проверка созданного продукта на соответствие требованиям.
- Контроль качества - тестирование + проверка продукта на соответствие заранее согласованному уровню качества на разных этапах разработки ПО.
- Обеспечение качества - контроль качества + превентивная система отслеживания качества.

Верификация/валидация:
- Верификация - статическая оценка системы с целью определения удовлетворяют ли результаты текущего этапа разработки ПО условиям, сформированным в начале этого этапа. Делаем ли мы продукт правильно? Процесс удостоверения того, соответствует ли продукт требованиям бизнеса
- Валидация - динамический процесс определения соответствия разрабатываемого ПО потребностям пользователя, желаниям заказчика, требованиям к системе. Делаем ли мы правильный продукт? Удостоверения на требования пользователя

Процесс тестирования включает в себя:

1. Тест менеджмент
2. Тест дизайн
3. Выполнение тестирования
4. Анализ результатов

Анализ требований:
1. Уровень бизнес-требований
2. Уровень пользовательских требований
3. Уровень продуктных требований (функциональные и нефункциональные)

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

Техники тест-дизайна:
- эквивалентное разбиение
- анализ граничных значений
- причина~следствие
- предугадывание ошибки
- исчерпывающее тестирование
- попарное тестирование
- таблица принятия решений

Тестовая документация:
1. Стратегия тестирования - общий недетализированный статический документ.
2. Тест план - детальное описание процесса тестирования.
3. Чек-лист - список проверок для тестирования продукта.
4. Тест-сьют - набор тест-кейсов.
5. Тест-кейс - это базовый тестировочный артефакт, который описывает совокупность шагов и условий, необходимых для проверки какой-либо функции или ее части.

Зачем нужны тест-кейсы? (анализ документации + ТТД = тест-кейсы)
- *чтобы проводить регрессионное тестирование*
- *они обеспечивают качественное тестовое покрытие* - то одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.
- *их может пройти любой член команды, даже джун*
- *их можно рассматривать как часть проектной документации (например, их может просматривать аналитик)*
- *основа для автоматизации*

6. Баг-репорт - технический документ, который содержит в себе полное описание бага, включающее информацию как о самом баге, так и об условиях его возникновения.

- *Баг — это несоответствие фактического результата выполнения программы ожидаемому результату.*

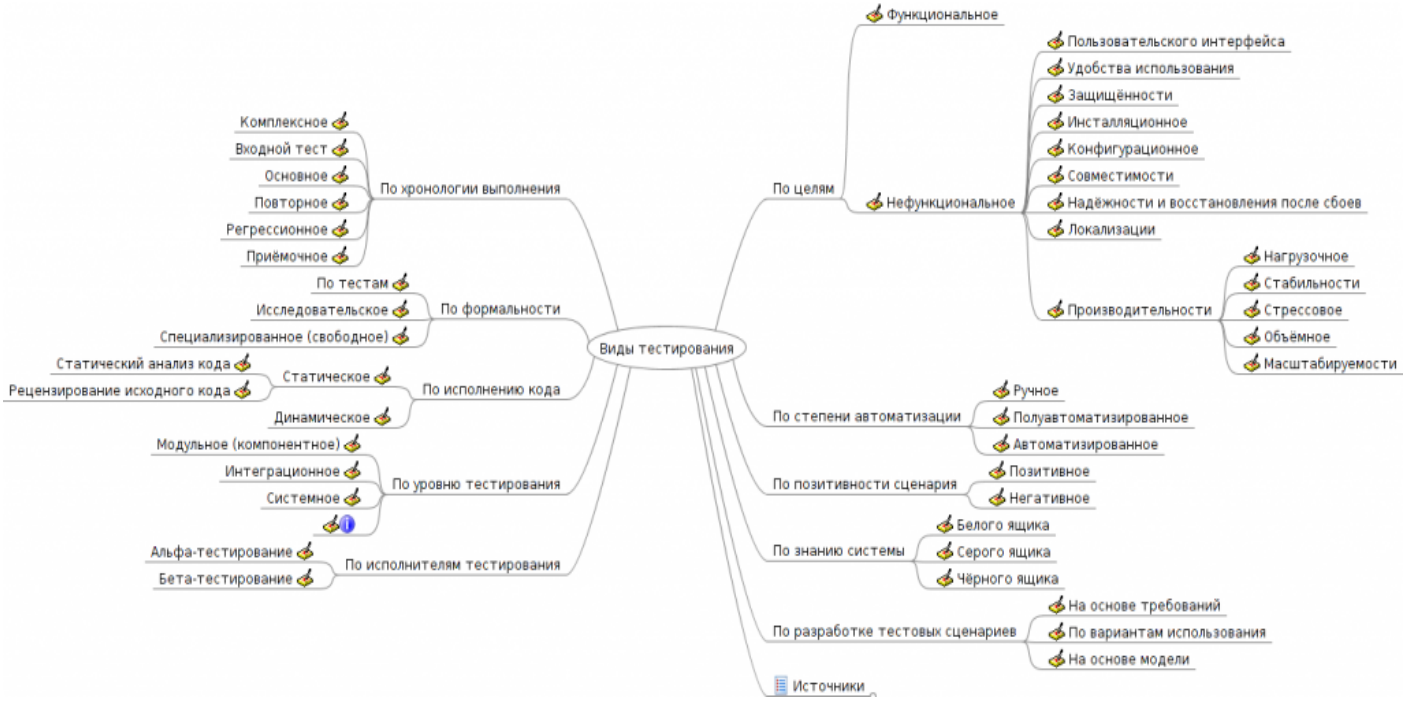
- Серьезность (Severity) — это атрибут, характеризующий влияние дефекта на работоспособность приложения.
- Приоритет (Priority) — это атрибут, указывающий на очередность выполнения задачи или устранения дефекта.
- Жизненный цикл бага - это последовательность этапов через которые проходит баг-репорт после своего создания.

7. Матрица соответствия требований - это двумерная таблица пересечений требований и имеющихся тестов, используемая для валидация тестового покрытия продукта.

8. Тест отчёт - документ, обобщающий результаты работ по тестированию.

9. Тестовый сценарий - это неавтоматизированный или автоматизированный сценарий, содержащий инструкции по реализации тестового набора.

Виды тестирования:



Критерии выхода из тестирования:
1. Время вышло
2. Эвристика пиньяты — до первой критической проблемы
3. Эвристика мертвой лошади — совокупность багов.
4. Задание выполнено — все определенные заранее проверки проведены, блокеров нет
5. Отмена задания — заказчик попросил прекратить.
6. Я зашёл в тупик — например, не хватает квалификации.
7. Уклонение от тестирования старой версии продукта.

API - набор функций, через которые одна компьютерная программа может взаимодействовать с другой.
Soap и Rest - это правила, по которым api функционирует. Soap - протокол обмена структурированными сообщениями в формате XML в распределённой вычислительной среде. Rest - архитектурный стиль взаимодействия компонентов в распределенной вычислительной среде, как правило, в формате json.

Agile (Agile software development) — гибкий подход к разработке программного обеспечения, который часто применяют в небольших командах и больших организациях.
Ценности Agile:
1. Люди и взаимодействие важнее процессов и инструментов.
2. Работающий продукт важнее исчерпывающей документации.
3. Сотрудничество с заказчиком важнее согласования условий контракта.
4. Готовность к изменениям важнее следования первоначальному плану.

Scrum — это каркас разработки, набор правил для организации гибкого рабочего процесса, который заключается в командном подходе, работе итерациями, фокусировке на цели каждой итерации и нестандартном распределении обязанностей внутри коллектива.
Клиент-серверная архитектура - сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг (серверами), и заказчиками услуг (клиентами). Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой посредством сетевых протоколов:
Ethernet - доступ к сети
IP - установка связи
TCP - передаётся объём данных
HTTP - определяет формат запроса и формат передаваемых данных.

HTTP - протокол прикладного уровня передачи данных, является протоколом клиент-серверного взаимодействия. Сообщения, отправленные клиентом называются запросами, а сообщения, полученные от сервера, называются ответами.

Структура запроса:
1. Строка запроса: метод, url, версия протокола
2. Заголовки: служебная информация, параметры, сведения : для обработки запроса. (Хост, юзер-агент, акцепт-language, тип контента)
ПУСТАЯ СТРОКА
3. Тело запроса: сами передаваемые данные.

Структура ответа:
1. Строка ответа: версия http, код состяния (ответа)
2. Заголовки
ПУСТАЯ СТРОКА
3. Тело ответа

Чем GET отличается от POST:

1. Запрос GET является идемпотентным, т.е. не меняет состояние на сервере, а просто получает от него данные. Запрос POST же вносит новую информацию на сервер. Поэтому GET может быть закэширован, остается в истории браузера и его можно добавить в закладки.
2. В GET запросе информация передается URL, в POST вся информация передается в теле запроса.
3. Данные безопаснее передавать в теле запроса, поэтому авторизовываться лучше через POST.

Коды состояния (код ответа):
102 — запрос в обработке
200 — запрос успешно обработан
201 — создан ресурс
204 — нет содержимого
301 — URI перемещен на постоянной основе
400 — неверный синтаксис запроса
401 — неавторизованно
403 — запрещено, нет прав доступа
404 — нет запрашиваемого ресурса на сервере
500 — внутренняя ошибка сервера

Веб-форма Регистрация (логин, пароль):

1. Есть ли требования к данной веб-форме? Если нет, то нужно протестировать на основе опыта и собственных представлений о работе данной формы.
2. Есть ли сроки для проведения тестирования? Следовательно, smoke-тестирование или расширенное тестирование.
3. Начать тестирование необходимо с положительных тестов.
4. Полезным инструментом для заполнения полей является Bug Magnet
5. В какую систему происходит регистрация? Это, например, может влиять на требование к сложности пароля.

Проверка	**Пример**	**Результат**
Тестирование методом черного ящика (позитивные тесты)		
Регистрация с допустимым логином и паролем	newuser, NewUser, New-user, new_user, new123user	Происходит авторизация
Классы эквивалентности: символов в логине или пароле [1; +∞)	Вводим количество символов в логине, в пароле 6 и 0.	Происходит авторизация при 6, сообщение об ошибке при 0
Анализ граничных значений: Максимальное(+-1) и минимальное(+-1) количество символов в логине, в пароле	Логин min: 0, 1, 2 Логин max: см. в DevTools(+-1) Пароль min: 0, 1, 2 Пароль max: см. в DevTools(+-1)	Происходит авторизация, кроме случаев min(-1) и max(+1) — при которых сообщение об ошибке.
Таблица принятия решений: логин +, пароль +	newuser, password newuser, -	Авторизация происходит в первом

логин +, пароль - логин -, пароль + логин -, пароль -	-, password -, -	случае, в остальных — сообщение об ошибке
Форма отражается согласно требованиям при различных разрешения экрана	-	Форма отражается корректно
Тестирование методом черного ящика (негативные тесты)		
Регистрация с недопустимым логином	new@user, «;!@#&*», пробелы, admin с первой буквой русской	Сообщение об ошибке
Тестирование безопасности: регистрация с логином, содержащим XSS и SQL инъекции	<skript>alert(123)</skript>, Select * From	Сообщение об ошибке
Регистрация под уже существующим пользователем	newuser	Сообщение об ошибке
Создать логин на японском, турецком, китайском	ユーザー	Сообщение об ошибке
Ввести данные в форму неправильно 5, 10, 50 раз	user@mail	Блокирует возможность заполнения на 5 мин после 10 попытки
Предугадывание ошибки: вставить данные извне	Скопировать логин и пароль из закладок	Данные вставляются в поля
Тестирование методом серого ящика через API (позитивные тесты)		
Классы эквивалентности		
Анализ граничных значений		
Таблица принятия решений		
Тестирование методом серого ящика через API (негативные тесты)		
Регистрация с недопустимым логином	new@user, «;!@#&*», пробелы, admin с первой буквой русской	Сообщение об ошибке
Тестирование безопасности: регистрация с логином, содержащим XSS и SQL инъекции	<skript>alert(123)</skript>, Select * From	Сообщение об ошибке
Регистрация под уже существующим пользователем	newuser	Сообщение об ошибке
Создать логин на японском, турецком, китайском	ユーザー	Сообщение об ошибке
Дополнительные тесты		
Тестирование локализации: правильно ли меняются названия полей в зависимости от выбранного языка	Перевести на китайский, поля корректно отражаются на китайском языке	Поля корректно отражаются на языке страницы
Юзабилити тестирование: удобно ли пользоваться данными формами, понятно ли пользователю интуитивно куда вводить	-	Пользователю удобно пользоваться формой

информацию		
Тестирование совместимости: попробовать: попробовать отразить форму с разных устройств (мобильный, ноутбук, ТВ), с разных браузеров, с разных ОС	-	Можно работать с разных устройств, браузеров, ОС
Можно ли кнопкой TAB переключаться между полями	-	Кнопка TAB переключает поля
Тестирование восстановления после сбоев:отключить интернет когда нажимаешь кнопку зарегистрироваться, а потом включить	-	Приложение зарегистрировало пользователя, можно войти
Проверить функцию «Запомнить меня на сайте» через Куки в DevTools	-	Куки сохраняют информацию о пользователе
Проверить, что используется протокол HTTPS	-	Данные должны передаваться по зашифрованному протоколу