ТЕСТИРОВЩИК / QA

Чеклист QA

Практическая база для тестирования продукта: требования, тест-дизайн, тестовая среда, дефекты, регрессия, релизная приемка и автоматизация.

Анализ требований

QA начинается не с кликов по интерфейсу, а с проверки требований на полноту, ясность, противоречия и тестируемость.

Проверить требования на тестируемость

Найти неясные формулировки, пропуски и противоречия до начала тестирования.

Что это дает: Чем раньше QA задает вопросы, тем дешевле исправлять требования и дизайн.

Как выполнить:

1. Прочитайте user story и acceptance criteria.
2. Отметьте неоднозначные слова: быстро, удобно, корректно, понятно.
3. Сформулируйте вопросы владельцу требования.

Критерии приемки:

- Неясные требования вынесены в список вопросов.
- Ответы зафиксированы в задаче.
- Критерии приемки можно проверить.

Определить риски качества

Понять, где дефект нанесет наибольший вред пользователю или бизнесу.

Что это дает: Risk-based testing помогает тестировать глубже там, где цена ошибки выше.

Как выполнить:

1. Определите критические пользовательские сценарии.
2. Оцените вероятность и влияние дефекта.
3. Выделите области для углубленных проверок.

Критерии приемки:

- Есть список рисков.
- Критические сценарии получили высокий приоритет.
- Команда согласовала уровень риска релиза.

Подготовить тестовые данные

Создать данные для позитивных, негативных и граничных сценариев.

Что это дает: Хорошие тестовые данные делают проверки воспроизводимыми и сокращают время на ручную подготовку.

Как выполнить:

1. Опишите нужные роли, статусы, тарифы, заказы и ограничения.
2. Подготовьте данные для edge cases.
3. Проверьте, что данные можно восстановить после теста.

Критерии приемки:

- Данные доступны в тестовой среде.
- Есть инструкция восстановления.
- Негативные сценарии не требуют ручного хака.

Тест-дизайн

Тест-дизайн помогает выбрать достаточный набор проверок: не тестировать все подряд, но закрыть важные риски.

Составить тест-кейсы для критических сценариев

Описать проверки так, чтобы их мог выполнить другой QA или разработчик.

Что это дает: Тест-кейсы сохраняют знание о продукте и помогают повторять проверки без потери качества.

Как выполнить:

1. Выберите сценарии из требований и рисков.
2. Опишите шаги, данные и ожидаемый результат.
3. Укажите приоритет и тип проверки.

Критерии приемки:

- Критические сценарии покрыты.
- Ожидаемый результат конкретен.
- Тест-кейсы связаны с требованиями.

Применить техники тест-дизайна

Использовать классы эквивалентности, граничные значения, таблицы решений и pairwise.

Что это дает: Техники тест-дизайна дают хорошее покрытие без бесконечного перебора вариантов.

Как выполнить:

1. Разбейте входные данные на классы эквивалентности.
2. Добавьте граничные значения.
3. Для сложной логики используйте таблицу решений.

Критерии приемки:

- Проверки обоснованы техникой.
- Границы и комбинации покрыты.
- Лишние дубли удалены.

Связать требования и проверки

Построить traceability между требованиями, тестами и дефектами.

Что это дает: Traceability показывает, что именно покрыто тестами и где изменение требования требует обновить проверки.

Как выполнить:

1. Свяжите тест-кейсы с user stories.
2. Отмечайте дефекты на уровне требования.
3. Проверяйте покрытие перед релизом.

Критерии приемки:

- У требований есть связанные проверки.
- Понятны непокрытые зоны.
- Дефекты связаны с затронутым требованием.

Выполнение проверок

На этапе выполнения QA проверяет продукт системно: с понятными данными, окружением, фиксацией результата и воспроизводимыми дефектами.

Выполнить smoke-тесты

Быстро проверить, что сборка пригодна для дальнейшего тестирования.

Что это дает: Smoke экономит время: если базовые сценарии сломаны, глубокое тестирование бессмысленно.

Как выполнить:

1. Определите минимальный набор критических сценариев.
2. Выполняйте smoke после деплоя на тестовую среду.
3. Останавливайте тестирование при критическом падении.

Критерии приемки:

- Smoke-набор опубликован.
- Результат зафиксирован.
- Критические падения сразу эскалированы.

Оформить качественный bug report

Зафиксировать дефект так, чтобы разработчик мог быстро понять и воспроизвести проблему.

Что это дает: Хороший bug report сокращает цикл исправления и уменьшает количество уточнений.

Как выполнить:

1. Укажите окружение, шаги, фактический и ожидаемый результат.
2. Добавьте скриншот, видео, логи или request/response.
3. Оцените severity и priority.

Критерии приемки:

- Дефект воспроизводится по шагам.
- Есть доказательство проблемы.
- Severity и priority объяснены.

Провести retest исправленного дефекта

Проверить, что конкретный дефект исправлен и не повторяется на указанных данных.

Что это дает: Retest подтверждает исправление, но не заменяет регрессионную проверку соседних сценариев.

Как выполнить:

1. Возьмите исходные шаги воспроизведения.
2. Проверьте исправление на той же среде и версии.
3. При необходимости добавьте смежные проверки.

Критерии приемки:

- Дефект закрыт только после успешного retest.
- Версия исправления указана.
- Смежные риски проверены или вынесены отдельно.

Регрессия и релиз

Перед релизом важно проверить не только новую функцию, но и критические старые сценарии, которые могли сломаться.

Спланировать регрессию

Выбрать набор проверок, который защищает важные старые сценарии перед релизом.

Что это дает: Регрессия снижает риск, что новая функциональность сломает работающие части продукта.

Как выполнить:

1. Определите затронутые модули.
2. Выберите critical path и интеграционные сценарии.
3. Согласуйте время и владельцев регрессии.

Критерии приемки:

- Regression scope утвержден.
- Проверки привязаны к рискам релиза.
- Результат доступен команде.

Подготовить QA verdict по релизу

Сформулировать состояние качества и риски, чтобы команда могла принять release decision.

Что это дает: QA verdict не решает за бизнес, но дает честную информацию о качестве и известных рисках.

Как выполнить:

1. Соберите результаты smoke, regression и critical bugs.
2. Опишите known issues и workaround.
3. Дайте рекомендацию: go, go with risks или no-go.

Критерии приемки:

- Вердикт опубликован до релиза.
- Риски понятны stakeholders.
- Известные дефекты имеют владельцев.

Разобрать дефекты после релиза

Понять, почему дефект дошел до production и как улучшить процесс.

Что это дает: Разбор production defects улучшает требования, тест-дизайн, автоматизацию и релизный контроль.

Как выполнить:

1. Опишите путь дефекта от требования до релиза.
2. Определите, какая проверка могла его поймать.
3. Добавьте изменение в процесс или тестовый набор.

Критерии приемки:

- Root cause описан.
- Есть preventive action.
- Новый контроль добавлен в чеклист или автотесты.

Автоматизация

Автоматизация полезна, когда она снижает повторяемую ручную работу и дает быстрый сигнал о регрессии, а не создает отдельный хрупкий проект.

Выбрать кандидатов для автоматизации

Определить проверки, которые дают ценность при регулярном автоматическом запуске.

Что это дает: Автоматизировать стоит стабильные, повторяемые и важные сценарии, а не все ручные проверки подряд.

Как выполнить:

1. Оцените частоту выполнения проверки.
2. Проверьте стабильность функциональности.
3. Сравните стоимость автоматизации и ручного выполнения.

Критерии приемки:

- Есть список кандидатов с приоритетом.
- Кандидаты связаны с рисками.
- Нестабильные зоны не автоматизированы преждевременно.

Контролировать flaky-тесты

Выявлять нестабильные автотесты и быстро возвращать доверие к тестовому набору.

Что это дает: Flaky-тесты разрушают доверие к автоматизации: команда перестает реагировать даже на реальные падения.

Как выполнить:

1. Отмечайте тесты, которые падают без изменения продукта.
2. Изолируйте причины: данные, ожидания, сеть, асинхронность.
3. Чините или временно выводите тест из quality gate.

Критерии приемки:

- Flaky-тесты видны в отчете.
- Есть владелец исправления.
- Quality gate не зависит от заведомо шумных проверок.

Интегрировать проверки в CI

Запускать нужные уровни тестов автоматически на pull request, merge и релизных сборках.

Что это дает: CI дает быстрый сигнал о регрессии и помогает ловить проблемы до ручной приемки.

Как выполнить:

1. Определите, какие тесты запускать на каждом событии.
2. Разделите быстрые и долгие наборы.
3. Публикуйте отчет так, чтобы команда видела причину падения.

Критерии приемки:

- CI запускает тесты по правилам.
- Отчет доступен команде.
- Падение quality gate блокирует небезопасный merge или релиз.