GeekBrains

Введение в тестирование
Урок 3

Тест-кейсы

Оглавление

На этом уроке

1. Узнаем, что такое тест-кейс и его атрибуты
2. Узнаем как составлять тест-кейсы и работать с ними
3. Ознакомимся с примерами тест-кейсов
4. Узнаем, какие ошибки могут возникнуть при составлении тест-кейсов
5. Перечислим преимущества и недостатки тест-кейсов
6. Обсудим инструменты для работы с тест-кейсами
7. Узнаем о принципе работы тест-кейса и чек-листа
8. Познакомимся с регрессионным тестированием

Что такое тест-кейс?

На прошлом занятии мы рассмотрели чек-листы. Но они не всегда позволяют полноценно проверить функциональность. Если тестируемое ПО сложное, тестировщик недостаточно знаком с функциональностью или нужна определённая подготовка тестовой среды, чек-листов недостаточно — в них не получится уместить столько информации без потери читаемости.

В таких случаях нужен сценарий или инструкция, в которой пошагово описан алгоритм проверки — то есть тест-кейс.

Тестовый случай (тест-кейс) — это совокупность шагов, конкретных условий и параметров, нужных для проверки работы тестируемой функции или её части.

Проще говоря, тест-кейс — это небольшая инструкция по проверке работы функции. В нём описывается:

- какие шаги нужно выполнить (например, какие кнопки нажать, чекбоксы активировать, данные внести), чтобы проверить работу участка системы;
- какой результат должен быть у каждого шага.

В тест-кейсе есть подробная детализация для каждой из проверок, а в чек-листе — нет.

Атрибуты тест-кейса

У тест-кейса есть обязательные атрибуты (необходимые поля). Большая часть заполняется на этапе проектирования тест-кейса, несколько — после прохождения теста.

Атрибуты на этапе проектирования тест-кейсов

1. **ID тест-кейса** — уникальный идентификатор, присваивается автоматически.
2. **Название** — краткое описание сценария, который проверяет тест-кейс.
3. **Шаги** — последовательность действий, чтобы проверить работу функции или выполнение сценария.
4. **Ожидаемый результат** — как должна вести себя система после каждого шага тест-кейса, то есть то поведение, которое мы ожидаем получить после выполнения конкретного действия или последовательности действий.
5. **Приоритет тест-кейса** зависит от приоритета функций и сценариев, которые он проверяет. Определяет очерёдность выполнения теста во время тестирования. Например, при высоком приоритете тест-кейс будет в начале очереди на выполнение, а при низком — в конце. Если во втором случае что-то сломано, для нас это не так критично, как в первом.
6. **Тестовые данные** — данные, которые используются для проверки. Могут указываться в шагах тест-кейса или в отдельном файле с указанием ссылки на них. Это, например, данные для авторизации пользователя или данные тестовой карты для оплаты заказа.
7. **Предусловия** — действия, которые нужно выполнить, прежде чем приступать к тест-кейсу, а также настройки приложения и тестовой среды.

   Например, если мы проверяем отправку сообщения в чате с другом в социальной сети, мы точно знаем, что пользователь должен быть авторизован. Но сама авторизация как набор действий с вводом логина/пароля не имеет отношения к отправке сообщения другу. Таким образом, правильно будет указать в предусловии этого кейса, что пользователь должен быть авторизован, а страница чата — открыта. В самих шагах мы будем описывать действия, имеющие непосредственное отношение к отправке сообщения другу.
8. **Постусловия** — выполнение тест-кейса, как правило, переводит систему из одного состояния в другое: меняются настройки, производятся расчёты. Состояние, в которое нужно привести систему после прохождения тест-кейса, указывается в постусловии.

   Допустим, мы проводили тестирование кейса с редактированием данных пользователя на боевом сервере (на окружении, которым пользуются обычные

пользователи) и в процессе тестирования изменили часть данных на ненастоящие: вместо имени указали странную последовательность букв, в описании интересов написали несвязный текст, вместо фотографии человека прикрепили небо.

Но, так как мы тестируем на продакшене, наш тестовый профиль может попасть в выдачу обычным пользователям, и это будет выглядеть странно. Чтобы этого избежать, мы можем в постусловиях указать, что все изменённые значения надо вернуть в исходное состояние и указать, какие данные надо ввести в каждое поле.

Атрибуты на этапе исполнения тест-кейсов

1. **Фактический результат** — то, что мы получаем после выполнения всего тест-кейса или его конкретного шага. Это необязательный атрибут, его можно не указывать. Но если он указывается, в нём может быть как краткое описание того, что мы получили по факту прохождения шага / шагов, так и статус для каждого фактического результата.

   Если фактический результат совпадает с ожидаемым, в графе с фактическим ставится статус passed — тест-кейс успешно пройден, ошибок не обнаружено. Если результат отличается от ожидаемого — статус failed — проверка провалена, полученный результат не соответствует ожидаемому. Во втором случае тестировщик создаёт отчёт о дефекте (о нём поговорим позже). К тест-кейсу обязательно добавляется ссылка на дефект, обнаруженный при исполнении.

2. **Статус тест-кейса** обозначает результат исполнения этого кейса или причину, по которой он не может исполняться. Основные статусы:

   - **passed** — успешно пройден
   - **failed** — кейс не прошел проверку
   - **skipped** — пропуск проверки
   - **blocked** — проверка заблокирована
   - **untested** — кейс еще не брали в работу

passed ставится, когда исследуемый объект проходит проверку в соответствии с ТЗ.

failed – исследуемый объект не соответствует ТЗ. Например, функционал поломан, не работает или его нет (разработчик потерял кнопку).

skipped — пропуск проверки кейса. Например, из-за нехватки времени или если в кейсе есть неточности: он устарел, из документации вы понимаете, что его уже изменили в проекте.

blocked — проверка заблокирована поломанным функционалом, от которого прямо или косвенно зависит тестируемый. Например, есть функционал «авторизация пользователя» и функционал «редактирование данных пользователя». Разработчик допустил ошибку в коде и функционал авторизации поломан — ему мы выносим резолюцию **failed**. Проверить редактирование данных пользователя не можем — выставляем статус **blocked**. В комментарии указываем, почему не можем протестировать этот функционал.

Priority в тест-кейсе

Тестировщик может сортировать тест-кейсы на выполнение по приоритету. Это полезно, если время на тестирование ограничено: можно оставить только тест-кейсы с высоким приоритетом, а остальные не проходить.

Приоритет можно выразить в любом удобном виде:

- буквами: A, B, C, D, E;
- цифрами: 1, 2, 3, 4, 5;
- словами: «крайне высокий», «высокий», «средний», «низкий», «крайне низкий» (или английскими аналогами).

Количество градаций не зафиксировано, но часто лежит в диапазоне от 3 до 5. Формат может определяться командой и возможностями системы хранения тест-кейсов.

Приоритеты выделяют на основании:

- важности требования, пользовательского сценария или функции, с которыми связан тест-кейс;
- важности потенциального дефекта, который может быть обнаружен в процессе выполнения тест-кейса.

Составление тест-кейсов

Представим реальный рабочий кейс: в скором времени команда планирует зарелизить (выпустить) новую фичу (доработку или новый функционал).

Разработчики активно кодят фичу, а тестирование получило требования для знакомства с фичей и вычитки — в жизненном цикле тестирования мы называем этот этап анализом требований. Когда тестировщик разберётся, что, как и почему должно быть в новой фиче, пора приступать к написанию тест-кейсов на неё. В идеале это должно происходить до непосредственного тестирования реализации.

Как писать тест-кейсы по требованиям — вопрос довольно неоднозначный. Ответ зависит от того, в каком виде они будут представлены — в виде текста разной степени детализации, в виде схем, майнд-карт или use case (пользовательских сценариев).

Вне зависимости от вида требований можно выделить общий алгоритм:

1. Продумайте, какие именно сценарии и функции нужно покрыть тест-кейсами. Для этого определите, какие новые возможности и сценарии появятся за счёт новой фичи, с какими функциями из существующих связана новая фича.
2. Сформулируйте названия для тест-кейсов.
3. Определите для каждого сценария нужные предусловия: например, пользователь или другой объект системы должен быть в определённом состоянии.
4. Опишите шаги, по которым будете проверять сценарий, в соответствии с требованиями. Всегда внимательно следите, чтобы не было пропусков.
5. Опишите ожидаемый результат в соответствии с требованиями.
6. Заполните остальные атрибуты этапа, если нужно.

Тест-кейс готов. Ничего сложного, если знать правила составления и работы.

Правила работы с тест-кейсами

1. Один тест-кейс — одна проверка или один сценарий.
2. Заголовок точно описывает суть тест-кейса. По нему можно определить характер проверок, не открывая шаги теста: то есть что конкретно проверяется в функциональности. Если мы говорим про кейс с отправкой

сообщения, заголовок может быть следующим: «Отправка сообщения другу из раздела “Чаты”»

3. Названия для элементов приложения (кнопок, чекбоксов, элементов меню) точные. Тест-кейсы используют разные люди, всем должно быть понятно их содержание. Не общепринятые названия элементов допускаются только в тех случаях, когда они понятны для всей команды.

4. Простой технический стиль без объяснений базовых понятий работы ПО. Подробности описываются в требованиях или спецификации. Тест-кейс не должен быть сочинением, иначе его будет трудно и долго читать.

5. Нет пропущенных шагов: каждое следующее действие вытекает из предыдущего. Например, в кейсе на проверку подписки на пользователя сразу после шага «Запустить приложение» не должно быть шага «Подписаться на пользователя», ведь нужно ещё перейти на страницу пользователя, на которого будем подписываться.

   Простой способ проверить, нет ли пропуска — пройти тест-кейс строго по описанным шагам, ничего не додумывая. Представьте, что тест-кейс открыли не вы, а другой человек.

6. Нет описания очевидных действий (например, разблокировки девайса). Лучше начинать с ненулевого состояния и определённого экрана. О них сообщить в предусловии.

7. Нет зависимостей от других тест-кейсов. Часто для выполнения тест-кейса нужно попасть в определённое состояние пользователя. В таких случаях высок соблазн в предусловии сослаться на тест-кейс, в котором мы как раз попадаем в нужное состояние. Но это неправильно: спустя время тест-кейс, на который мы ссылаемся, может быть удален или изменён. Поэтому чётко описываем в предусловии, что должно быть сделано до выполнения тест-кейса, в шагах чётко прописываем все действия, даже если они частично дублируют действия из другого тест-кейса

8. Каждый тест-кейс уникален и не дублирует другой.

9. Обнаруженная ошибка становится очевидной. Шаги тест-кейса и ожидаемые результаты описываются таким образом, чтобы отклонение от них явно свидетельствовало о дефекте.

10. Гибкость для модификации. В тест-кейсе легко добавить, изменить или убрать шаг, в том числе в середине теста, а также изменить тестовые данные.

Ошибки в тест-кейсах

1. Заголовка нет или он сформулирован некорректно. Например, из заголовка «Кнопка “Сохранить”» или «Чат с поддержкой» непонятно, что мы проверяем в кейсе. То, что они есть, или их функциональность?

   Заголовок должен кратко, но чётко описывать суть проверки.

2. Ссылки ведут на разные или недействительные требования. В тест-кейсах можно ссылаться на требования, но если мы используем несколько ссылок, стоит убедиться, что их содержимое актуально и не противоречит друг другу.

3. Используются личные формы глаголов: «нажми», «перейдите», «укажи». В тест-кейсах, как и в чек-листах, следует использовать обезличенные формы глаголов: «нажать», «перейти», «указать».

4. Пунктуационные, орфографические, синтаксические ошибки. Текст с ошибками и опечатками читать сложнее, чем текст, не нарушающий правила языка.

5. «Выдумывание» особенностей поведения приложения без отсылки к требованиям. Тест-кейс должен описывать сценарий и ожидаемые результаты чётко в соответствии с требованиями. Тестировщик ничего не выдумывает.

6. Нет описания приготовления к выполнению тест-кейса, если требуются предусловия. Например, для проверки фишек верифицированного пользователя в соцсети нам нужен такой верифицированный пользователь. При тестировании у нас есть возможность создать его через админку (не так, как это происходит у обычного пользователя). Нам подойдёт этот вариант, так как мы проверяем не верификацию, а функцию, доступную такому пользователю.

   Соответственно, верификация пользователя — наше предусловие. В нём обязательно нужно либо сказать, что надо верифицировать пользователя по такой-то ссылке, либо дать конкретные данные для авторизации пользователем.

Примеры тест-кейсов

Тест-кейсы составляются исходя из требования. Поэтому сперва внимательно ознакомимся с требованием, а потом уже составим тест-кейсы.

Представим, что есть форма регистрации. Заголовок формы — «Начни бесплатно». Шрифт — Sans Serif, размер текста — 40px. Есть три поля:

1. **«Имя и фамилия».** Принимает на вход русский и английский текст. Когда пользователь начинает заполнять поле, плейсхолдер (текст-подсказка) скрывается.
2. **«Email».** Принимает на вход адрес электронной почты в формате email@mail.ru. Когда пользователь начинает заполнять поле, плейсхолдер скрывается.
3. **«Номер телефона».** Пользователь может выбрать страну и ввести свой номер телефона. При вводе номера автоматически добавится код выбранной страны. Вводимый номер будет разделяться пробелами.

Для отправки формы есть кнопка «Начать». Она становится активной (на неё можно нажать), если заполнены все поля.

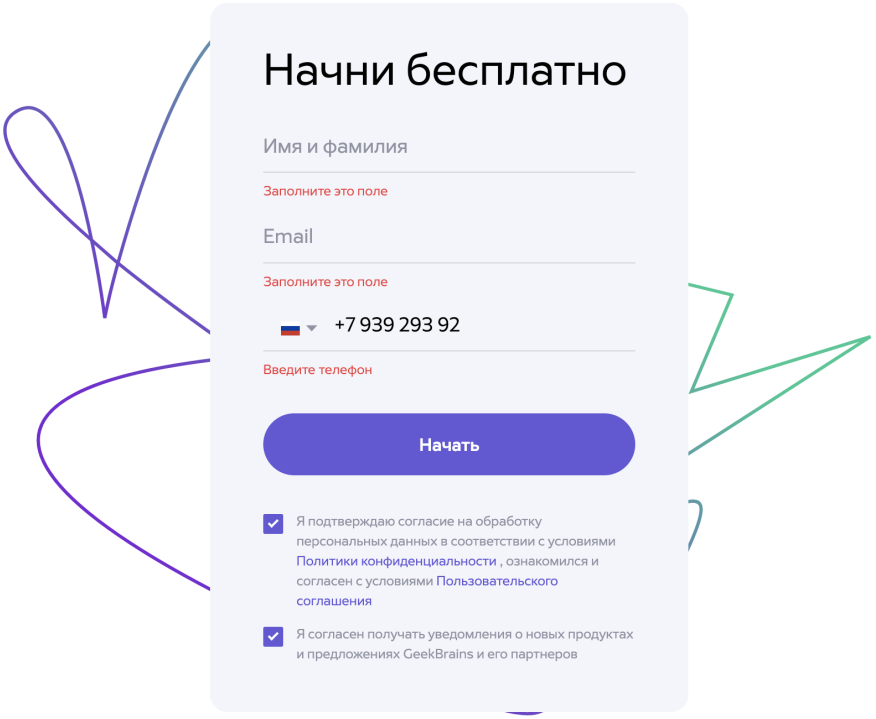
Есть два чекбокса:

- «Я подтверждаю согласие на обработку персональных данных в соответствии с условиями Политики конфиденциальности, ознакомился и согласен с условиями Пользовательского соглашения»;
- «Я согласен получать уведомления о новых продуктах и предложениях GeekBrains и его партнеров».

При определённых условиях отображаются ошибки:

- если поле «Имя и фамилия» не заполнено — «Заполните поле Имя и фамилия»;
- если поле «Email» не заполнено — «Заполните поле Email»;
- Если поле «Номер телефона» не заполнено — «Заполните поле Номер телефона».

Размер текста ошибки — 12px, шрифт — Sans Serif, цвет — #f2363b.



Реализация формы разработчиком

Первый тест-кейс будет проверять наличие заголовка формы «Начни бесплатно».

ID: 1

Название: отображение заголовка формы регистрации

Предусловие: открыта форма регистрации

Шаг 1: проверить отображение и наличие заголовка формы

Ожидаемый результат 1: заголовок формы «Начни бесплатно» отображается. Шрифт — Sans Serif, размер текста — 40px.

Теперь составим второй тест-кейс. Проверим заполнение и отображение поля «Имя и фамилия». Будет два разных тест-кейса. Начнём с отображения:

ID: 2

Название: отображение поля «Имя и фамилия»

Предусловие: открыта форма регистрации

Шаг 1: проверить наличие поля «Имя и фамилия»

Ожидаемый результат 1: поле «Имя и фамилия» отображается в соответствии с макетом

Шаг 2: проверить наличие плейсхолдера в поле «Имя и фамилия»

Ожидаемый результат 2: плейсхолдер «Имя и фамилия» отображается в поле, размер шрифта — 14px

Далее проверим заполнение поля:

ID: 3

Название: заполнение поля «Имя и фамилия»

Предусловие: открыта форма регистрации

Шаг 1: клик в поле «Имя и фамилия»

Ожидаемый результат 1: в поле отображается курсор

Шаг 2: ввести в поле Иван Иванов

Ожидаемый результат 2: плейсхолдер скрывается, введённый текст отображается

А если пользователь заполнил поле невалидными данными или оставил поле пустым?

ID: 4

Название: отображение ошибки «Заполните поле Имя и фамилия»

Предусловие: открыта форма регистрации

Шаг 1: клик на кнопку «Начать»

Ожидаемый результат 1: отображение ошибка «Заполните поле Имя и фамилия». Размер текста — 12px, шрифт — Sans Serif, цвет — #f2363b

Преимущества и недостатки тест-кейсов

Преимущества:

- За счёт полной детализации шагов тест-кейсы может проходить новичок. По ним удобно знакомиться с продуктом.
- Достаточно подробное описание бизнес-логики. При неидеальной организации хранения требований уточнение некоторых моментов проще будет найти в тест-кейсах.

Недостатки:

- Нужно больше времени на написание, чем для чек-листов.
- Сложно поддерживать. Если изменится бизнес-логика, будут переименованы разделы или произойдут другие изменения, затрагивающие пользовательский интерфейс и сценарии использования системы, нужно будет актуализировать все тест-кейсы, связанные с измененной частью.

Инструменты для работы с тест-кейсами

Тест-кейсы нужно структурировать, хранить и поддерживать. У тестировщиков должна быть возможность работать с ними совместно.

Системы для работы с тест-кейсами (test-case management system, TMS) — автоматизированные приложения, которые позволяют управлять тест-кейсами. К ним относятся:

- Jira + Zephyr,
- TestRail,
- Allure,
- Test IT,
- TestLink,
- Прочие.

Основные функции инструментов:

1. **Создание тест-кейсов** — приложения содержат специальные формы и шаблоны, которые ускоряют процесс разработки тест-кейсов.

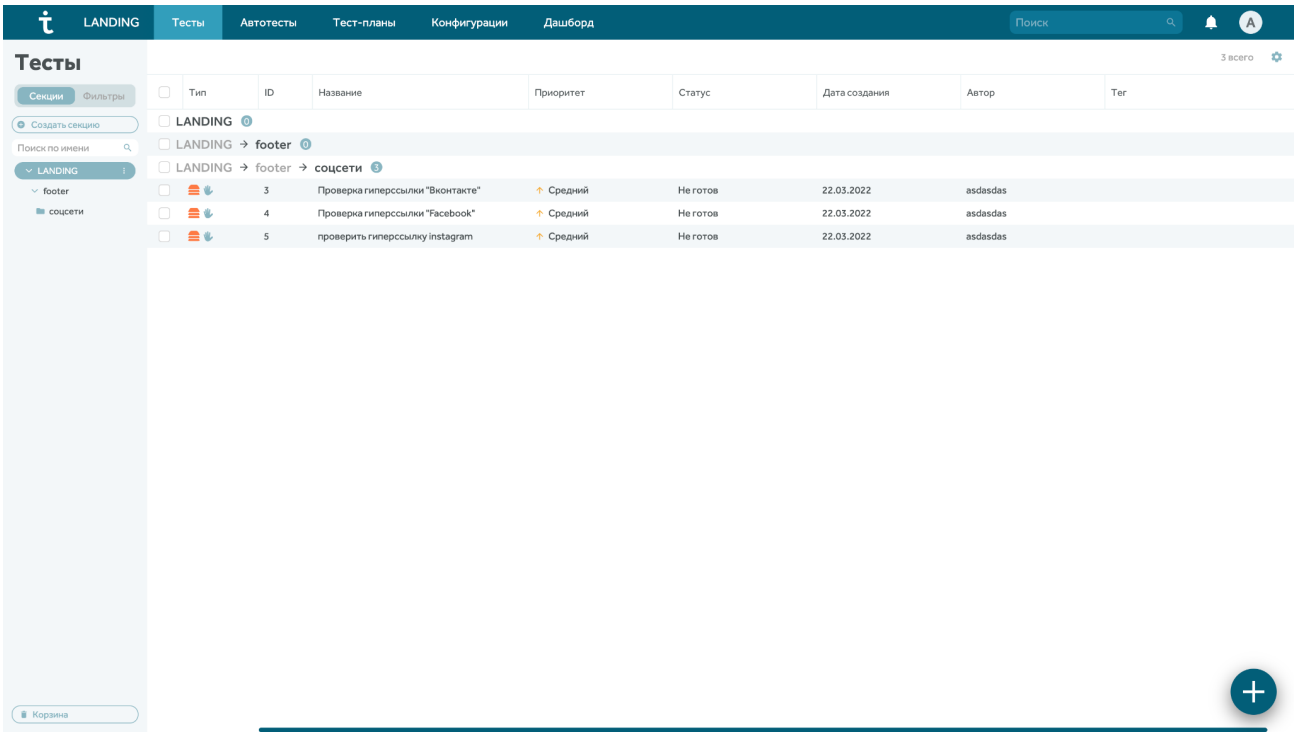
2. **Выполнение тест-кейсов и запись результатов** — тест-кейсы запускаются с использованием приложения, и результат их исполнения записывается в систему.

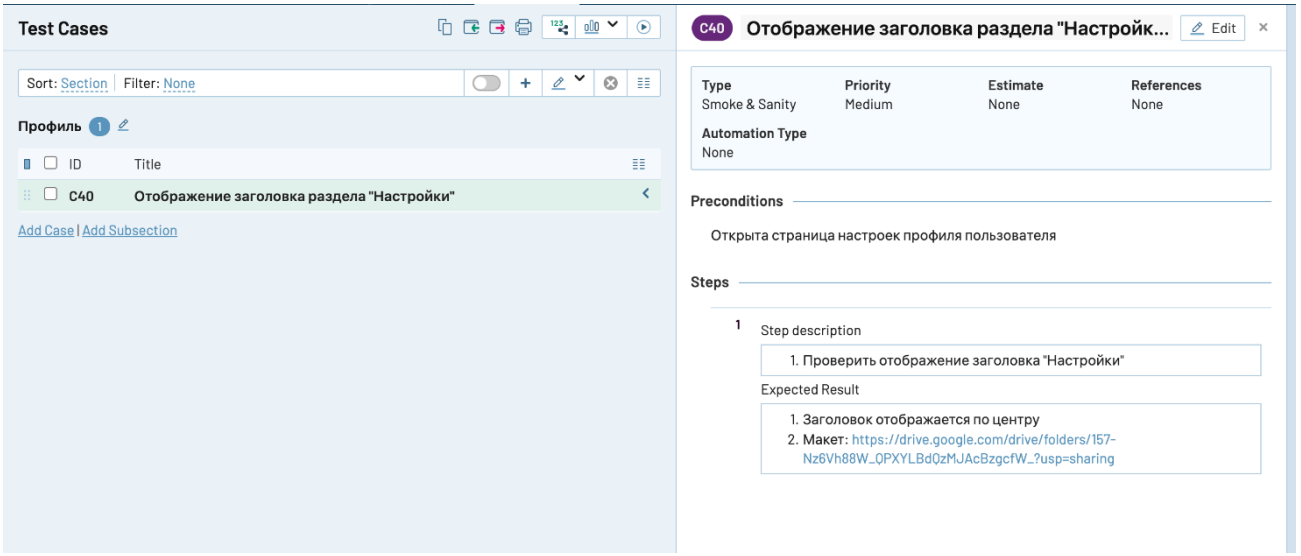
3. **Отслеживание дефектов** — возможность прикреплять ссылку на дефект, который обнаружен тест-кейсом. Это позволяет эффективнее отслеживать исправление дефектов.

4. **Отслеживание требований и других проектных документов** — в приложениях можно прикреплять ссылки на требования, по которым разрабатывается тест-кейс, и отслеживать полноту покрытия требований тест-кейсами.

5. **Защита тест-кейсов** — тест-кейсы создают для многократного использования, поэтому их важно защитить от несанкционированных изменений и удалений. Все действия тестировщиков с тест-кейсами отслеживаются и сохраняются в истории версий.

Давайте посмотрим, как будут выглядеть заведённые тест-кейсы в TMS:





Тест-кейс и чек-лист

И тест-кейс, и чек-лист — это упорядоченная последовательность проверок работы программы. Они представляют собой документацию к продукту и содержат информацию о его функциональности.

Различия тест-кейса и чек-листа:

	Тест-кейс	**Чек-лист**
Детализация	Проверки детализированы, расписаны по шагам	Проверки формулируются в общем виде. Часто пункт чек-листа = заголовок тест-кейса
Понятность	Понятен любому человеку	Понятен человеку, который знаком с продуктом
Эффект пестицида	Есть риск возникновения, так как каждый раз воспроизводятся одни и те же шаги	Риск снижается, так как каждый выполняет проверку по-своему

Ценность тест-кейсов и чек-листов

1. **Структурируют и систематизирую подход к тестированию.** Позволяют оценить объём предстоящей работы, распределить задачи между тестировщиками в команде, не пропустить важных проверок.
2. **Основа для метрик тестового покрытия.** Помогают оценить, сколько процентов требований протестируют, а на какие требования тесты ещё не появились. Иными словами, какой процент технического задания покрыт тестами.
3. **Основа для увеличения тестового покрытия.** Если тест-кейсов недостаточно, нужно добавить.
4. **Показатель соответствия ситуации плану**:
    - сколько тестов уже выполнено;
    - какие из них прошли успешно;
    - как много осталось проверить;
    - какие функциональности тестировались или нет.
5. **Поддерживают взаимопонимание** между заказчиком, разработчиками и тестировщиком. Написание тест-кейсов часто приводит к дополнительным вопросам по работе приложения. Это закрывает пробелы в знаниях о системе, устраняет недопонимание с заказчиком или разработчиком.
6. **Хранят информацию** для длительного использования и обмена опытом между сотрудниками и командами. Тест-кейсы и чек-листы используются, чтобы удобно передавать знания о системе другим членам команды, а также обращаться к ним при решении спорных вопросов — при условии, что тест-кейсы написаны качественно и корректно.
7. **Основа регрессионного тестирования**. Тест-кейсы делают повторные проверки регулярными и полноценными.

Наборы тест-кейсов

Test case — атомарный элемент тест-плана. Тест-план включает в себя следующие компоненты:

- **Test suite** — список кейсов, объединённых общим фактором: весь продукт, конкретная фича и так далее.
- **Test plan** — список тест-сьютов, выбранных для теста.
- **Test run** — проход («прогон») тест-сьютов, выбранных для тестирования в соответствии с тест-планом.

Регрессионное тестирование

В течение всего жизненного цикла продукта в него вносят изменения — правки, доработки, рефакторинг кода и новые фичи. Прежде чем отправить любые изменения в коде в релиз, нужно убедиться, что работа системы не стала хуже. То есть перед релизом мы проверяем не только что внесённые изменения действительно есть и работают корректно, но и проверяем, что старый функционал не был поломан.

Регрессионное тестирование (regression testing) — тестирование уже проверенной функциональности после внесения изменений в код для уверенности, что эти изменения не внесли или не активизировали ошибки в областях, которые не подвергались изменениям.

Для регрессионного тестирования выбирают уже ранее составленные тест-кейсы

Контрольные вопросы

1. Что такое тест-кейс и для чего он нужен?
2. Какие атрибуты заполняются на этапе создания тест-кейса?
3. Какие атрибуты заполняются на этапе выполнения тест-кейса?
4. В каком случае проверке ставится статус passed? failed? skipped? blocked?
5. Какие есть плюсы у тест-кейсов?
6. Какие минусы у тест-кейсов?
7. Что такое test suite?
8. Что такое test plan?
9. Что такое test run?
10. Что такое регрессионное тестирование и когда его проводят?

Дополнительные материалы

1. Пишем максимально эффективный тест-кейс
2. Что такое тест-кейс и как его писать
3. Что такое регрессионное тестирование?
4. Антирегрессионное тестирование – минимизируйте затраты