**Лекція 15. Тестова документація**

### Види документів та для чого вони призначені (****найпоширеніші тестові артефакти****)

**Формальні** - ті, що є частиною проекту. Вони передаються замовнику і всі члени команди мають доступ до перегляду, коментування, або редагування:

* **ТЗ** (технічне завдання) - це документ, що встановлює призначення, показники якості, візуалізацію, технічні, економічні та інші спеціальні вимоги до продукту. Найчастіше з ТЗ приходить в компанію від замовника або майбутнього власника продукту. Але саме тестувальник може його верифікувати та адаптувати документ. Йдеться про те, що ТЗ дає чітке розуміння що потрібно розробити, для чого та який функціонал і дизайн має мати майбутній продукт. ТЗ дозволяє донести суть того що слід створити команді. Допомагає зрозуміти, яким саме функціоналом повинен володіти продукт, іноді із зазначенням використовуваних технологій і методами його реалізації.
* **Вимоги** (**Software requirements specification**) – це документ основа основ, того що буде реалізовано. У загальному вимоги описують перелік побажань замовника, і те що повинен робити продукт. Документ, що описує весь функціонал продукту, деталі, економічні показники. Тобто це взірець, відповідно до якого має бути розроблений продукт. Щоб тестувальники могли перевірити продукт/функціонал згідно вимог, хтось має ці вимоги написати. QA - саме ці спеціалісти, що створюють ці вимоги. Тобто вимоги - превентивні методи запобігання дефектів/багів в тому чи іншому середовищі.
* **Технічна документація** – зазвичай містить повний опис логіки конкретної частини продукту, що розробляється і варіанти, сценарії використання предмета розробки користувачами.
* **Тест-кейси** - **Test Case** (або ж тестовий випадок) - це професійна документація тестувальника, де прописаний сценарій перевірки того чи іншого аспекту продукту та очікуваний ідеальний результат. Тестувальники описують всі можливі варіанти перевірок, аби потім проходити по цих сценаріях та не пропустити помилок у ПЗ. **Test Case** - це описана послідовність певних дії (кроків) і очікуваний результат для перевірки роботи певного функціоналу системи. Також необхідно, щоб опис кейса був таким, щоб виконати його міг будь-хто (тестувальник, розробник, аналітик, замовник).
* **Баг-репорти** - це, в сутності, тест-кейс, тобто вже пройдений сценарій на практиці, але з фактичним результатом, що не збігається з очікуваним. Цим документом тестувальник повідомляє розробнику, що в створюваній системі є помилка і ПЗ працює неправильно. Це технічний документ, що описує ситуацію або послідовність дій, яка призвела до некоректної роботи об'єкту тестування, із зазначенням причин і очікуваним результатом

**Неформальні** - ті, що призначені для внутрішнього користування команди, або окремого члену команди. Можуть бути необов’язковими:

* **Тест-план** (**Test Plan**) - це документ, який описує весь обсяг робіт з тестування, починаючи з опису об'єкта тестування, стратегії, розкладу, критеріїв, початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення. Описує стратегію, яка буде застосовуватися для тестування програми, ресурсів, що будуть використовуватися, тестове середовище, в якому буде проводитися тестування, а також обмежень тестування та графіку тестування. Тест-план є важливою складовою будь-якого грамотно-організованого процесу тестування, так як містить у собі всю необхідну інформацію процесу тестування. Як правило, за написання Тест-плану, розробку Тест-дизайну відповідальний керівник групи з питань Забезпечення Якості (QA) або досвідчений Senior QA engineer.

**План Тестування повинен включати в себе наступне:** вступ, обгрунтування необхідності тестування, **Check List**, перелік функціоналу, який тестуватиметься, зазначення підходу, який використовуватиметься під час тестування ПЗ, перелік результатів, які необхідно перевірити, включення ризиків пов’язаних із тестуванням, графік виконання завдань та етапів.

* **Чек-листи - Check List** - перелік завдань на визначений термін, що потребує відмітки про виконання. Це конкретний список того, що потрібно перевірити. Від допомагає планувати терміни закінчення робіт у майбутньому й сьогоденні. **Check List —** це частина **Test Plan,** конкретний список того, що потрібно перевірити. Від допомагає планувати терміни закінчення робіт у майбутньому й сьогоденні. У ньому можна відмічати скільки часу необхідно для перевірки і скільки було витрачено. Відображає ступінь готовності продукту. Check List із результатами наочно показує у будь-якого співробітника компанії поточний стан продукту, що розробляється. Зберігає історію раніше проведених тестів. Не дає забути, які тести необхідно виконати в першу чергу, які в другу, які в третю і т. д. Це породжує впевненість, що за певний запланований час найважливіші тести будуть проведені, а результати по ним — отримані. Також можна буде легко звірити, які саме тести проходили з помилками, і перепровірити їх ще раз, лише їх наприклад. Чек-лісти варіанти записати у Google Sheets або у вигляді інтелектуальної карти.
* **Матриці тресабіліті (Traceability Matrix)** - це таблиця, що показує покриття тестами вимог. Тобто який функціонал вже перевірили, а який ще потребує перевірки. Також цей документ візуалізує відсоткове співвідношення протестованості між частинами ПЗ. Ця таблиця використовується для відстеження вимог під час життєвого циклу розробки програмного забезпечення. Основними цілями створення матриці є: впевненість у тому, що програма розроблена відповідно до зазначених вимог; допомогти знайти причину будь-якої помилки, гарний помічник який підказує які документи слід відстежувати на різних етапах циклу розробки програмного забезпечення.
* **Test Scenario** – повідомляє про те, яку ділянку і у якому порядку в програмі буде перевірено.

Розглянемо використання документів**.**

Фундаментальний процес тестування виглядає так: 

Виконання тестів необхідно, але не менш важливі і супроводжуючі дії – планування і документування процесу. В обов’язки тестувальників входить розробка тестових сценаріїв, а також підготовка тестування і оцінка його результатів. Становлення ідеї фундаментального тестового процесу на всіх рівнях тестування зайняло роки. У рамках цього процесу можна виділити ключові кроки:

* Планування та управління;
* Аналіз та проектування;
* Впровадження та реалізація;
* Оцінка критеріїв виходу і написання звітів;
* Дії по завершенню тестування.

Тут дії описані в логічній послідовності, але в умовах реального проекту вони можуть накладатися, відбуватися одночасно або навіть повторюватися. Зазвичай, відбувається адаптація цих кроків під потреби конкретної системи або проекту. Розглянемо їх.

1. **Планування та управління**

Планування тестування включає дії, спрямовані на визначення основних цілей тестування і завдань, виконання яких необхідне для досягнення цих цілей.

У процесі планування ми переконуємося в тому, що ми правильно зрозуміли цілі та побажання замовника і об’єктивно оцінили рівень ризику для проекту, після чого ставимо цілі і завдання для, власне, тестування.

Для більш ясного опису цілей і завдань тестування складаються такі документи як тест-політика, тест-стратегія і тест-план.

**Тест-політика** – високорівневий документ, що описує принципи, підходи і основні цілі компанії в сфері тестування.

**Тест-стратегія** – високорівневий документ, що містить опис рівнів тестування і підходів до тестування в межах цих рівнів. Діє на рівні компанії або програми (одного або більше проектів).

**Тест-план** – документ, що описує засоби, підходи, графік робіт і ресурси, необхідні для проведення тестування. Крім іншого, визначає інструменти тестування, функціональність, яку потрібно протестувати, розподіл ролей в команді, тестове оточення, техніки тест-дизайну, що використовуються, критерії початку та закінчення тестування та ризики. Тобто, це докладний опис всього процесу тестування.

У будь-якій діяльності, управління не закінчується плануванням. Нам потрібно контролювати і вимірювати прогрес. Саме тому управління тестуванням – безперервний процес.

**Управління тестуванням** – зіставлення поточної ситуації в процесі тестування із планом та складання звітності.

У свою чергу, дані, отримані в ході контролю над процесом, враховуються при плануванні подальших дій.

1. **Аналіз та проектування**

Аналіз та проектування тестів – це процес написання тестових сценаріїв і умов на основі загальних цілей тестування.

У процесі аналізу і проектування ми розробляємо тестові сценарії на підставі загальних цілей тестування, визначених під час планування.

**Тестовий сценарій** – документ, що визначає встановлену послідовність дій при виконанні тестування.

1. **Впровадження та реалізація**

Під час виконання тестування відбувається написання тест-кейсів, на основі написаних раніше тестових сценаріїв, збирається необхідна для проведення тестів інформація, готується тестове оточення і запускаються тести.

**Тест-кейс** – документ, що містить набір вхідних значень, перед- та післяумови, а також очікуваний результат проведення тесту, розроблений для перевірки відповідності визначеної функціональності системи заданим для цієї функціональності вимогам.

**Тестове оточення** – апаратне і програмне забезпечення та інші засоби, необхідні для виконання тестів.

1. **Оцінка критеріїв виходу і написання звітів**

Критерії виходу визначають, коли можна завершувати тестування. Вони необхідні для кожного рівня тестування, оскільки нам необхідно знати, чи достатньо було проведено тестів.

При оцінці критеріїв виходу необхідно:

* Перевірити, чи було проведено достатню кількість тестів, чи досягнута потрібна ступінь забезпечення якості системи;
* Переконається в тому, що немає необхідності проводити додаткові тести. Якщо все ж така необхідність є, можливо, буде потрібно змінити встановлений критерій виходу.

Після закінчення тестування відбувається написання звіту, який буде доступний всім зацікавленим сторонам. Адже не тільки тестувальники повинні знати результати виконання тестів, – ця інформація може бути необхідна багатьом учасникам процесу створення ПЗ.

1. **Дії після завершення тестування**

При завершенні тестування ми збираємо, систематизуємо і аналізуємо інформацію про його результати. Вона може стати в нагоді пізніше – при випуску готового продукту. Можуть бути й інші причини для згортання тестування, наприклад, дострокове закриття проекту або завершення певного етапу розробки.

Основні цілі цього етапу:

* Переконатися, що вся запланована функціональність дійсно була реалізована;
* Перевірити, що всі звіти про помилки, подані раніше, були, так чи інакше, закриті;
* Завершення роботи тестового забезпечення, тестового оточення та інфраструктури;
* Оцінити загальні результати тестування і проаналізувати досвід, отриманий в його процесі.

Щодо тестування існує стандарт ДСТУ ISO/IEC/IEEE 29119-1:2017 Інженерія систем і програмних засобів. Тестування програмних засобів. Частина 1. Поняття та визначення. Мета серії стандартів ISO/IEC/IEEE 29119 — визначити узгоджену на міжнародному рівні множину стандартів для тестування програмних засобів, які може застосовувати будь-яка організація під час тестування програмних засобів.

Є багато різних типів програмних засобів, організацій програмування та методологій. Сфера програмування охоплює інформаційні технології (IT), персональні (ПК), вбудовані, мобільні та наукові комп’ютери та багато інших категорій. Організації програмування варіюються від маленьких до великих, від місцевих до світових і від комерційних до орієнтовних на державну службу. Методології програмування охоплюють об’єктно-орієнтовану, традиційну, керовану даними та гнучку методології. Ці та інші чинники впливають на тестування програмних засобів. Ця серія стандартів може підтримувати тестування в різних контекстах.

Цей стандарт містить терміни, на яких побудовані інші частини сериї стандартів тестування програмних засобів ISO/IEC/IEEE 29119, що полегшує їх застосування, та в яких наведені приклади практичного застосування. Частина 1 є довідковою, подаючи визначення, опис понять тестування програмних засобів та способів застосування процесу тестування програмних засобів, визначеного в цьому стандарті, і настанову до інших частин стандарту ISO/IEC/IEEE 29119.

Спочатку обговорено поняття тестування програмних засобів. Описано роль тестування програмних засобів у контексті організації та проекту. Пояснено тестування програмних засобів у загальному життєвому циклі програмних засобів, впроваджуючи спосіб, який дає змогу встановити процеси та під- процеси тестування програмних засобів для конкретних об’єктів тестування або з конкретною метою тестування. Описано, як тестування програмних засобів вписується в різні моделі життєвого циклу. Продемонстровано застосування різних методів планування тестування; а також як автоматизацію можна застосовувати для підтримки тестування. Також обговорено залучення тестування.в керуванні дефектами. У додатку А описано роль тестування в ширшій сфері верифікації та валідації. Додаток В містить короткий вступ у сферу вимірів, які застосовують для відстеження та контролювання тестувань. У додатку С наведено приклади застосування стандарту в різних моделях життєвого циклу. Додаток D містить детальні приклади підпроцесів тестування. Додаток Е містить додаткову інформацію про ролі та обов’язки, з якими зазвичай стикаються групи тестувальників і незалежні тестувальники. У кінці стандарту наведено бібліографію.

**Детальний розгляд тестових артефактів**

**Тест-кейси** — це база, професійна документація тестувальника, де прописаний сценарій перевірки того чи іншого аспекту (фічі) та очікуваний результат. Структура даного артефакту полягає в наступній парі:

* *Виконувана дія* (Action) — *Очікуваний результат* (Expected result).

Крім цього, тест-кейс може містити й додаткову інформацію, реалізовану у вигляді наступних атрибутів:

* Попередні умови (Preconditions) — передумов: містять важливу інформацію та кроки, необхідні для попередньої підготовки операційної системи, тестового застосунку, мобільного пристрою, браузера тощо. Виконання цих умов завжди передує проходженню тест-кейса. Якщо вимоги, зазначені в передумовах, не виконані, тест-кейс буде неможливо відтворити по зазначених кроках, або ж результати тестування будуть спотворені.
* Опис (Description) — опису: лаконічно відповідає на питання «Навіщо нам цей кейс, яка його мета?»
* Тестові дані (Test Data) — включають: вхід, що задовольняє попереднім вимогам (Preconditions), наприклад: 100 згенерованих записів у базі даних, набір користувачів з різними рівнями доступу до системи, текст, який має понад 300 символів тощо.
* Скриншотів (Screenshots): візуальна репрезентація тестових атрибутів.
* Постумови (Postconditions) — включають: зміни, які повинні бути внесені в систему після виконання дій тест-кейса — це може передбачати видалення тестових даних з бази, скидання налаштувань профілю до замовчуваних значень тощо.

Тест-кейси бувають різного формату та деталізації. Наприклад, досить популярним є формат BDD тест-кейсів (Behavior-Driven Development - розробка, орієнтована на поведінку, це такий підхід до створення програмного забезпечення, що зосереджується на тому, як програма має поводитися в конкретних ситуаціях). Вони пишуться з використанням так званої мови Gherkin i схематично мають наступний вигляд:

Scenario #1: Successful Log in to the website

Given user brings up the login pop-up

When user clicks Sign-in

And user enters a valid email and password

And user clicks Sign-in

Then user should be successfully logged into the site

Хорошою практикою вважається створювати тест-кейси паралельно з розробкою, щоб як тільки задача мігрує в стан готовності до тестування ‘Ready For Test’, QA може братися до роботи. До того ж розробники заздалегідь можуть бачити тестові випадки, які повинні бути враховані на стадії розроблення.

**Баг-репорти** — це, в сутності, тест-кейс, тобто вже пройдений сценарій на практиці, але з фактичним результатом, що не збігається з очікуваним.

Взірцевий репорт складається з наступних елементів:

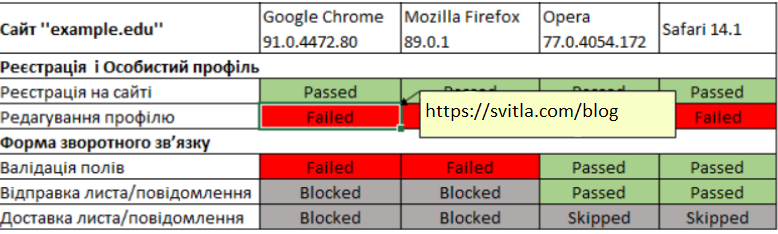
1. Title.
2. Environment.
3. Steps to Reproduce.
4. Actual + Expected Result.
5. Screenshots/Logs.
6. Severity + Priority.

Заголовок баг-репорту повинен бути інформативним або ж, хочеться використати англіцизм, — self-descriptive. Перераховані кроки повинні дозволяти точно відтворити проблему, а фактичний результат вичерпно описувати поточну поведінку функціонала. Також бажано, щоб на скриншоті був не лише UI, а й фрагмент консолі/ мережі — це відчутно спростить життя команді розробника.

*Якщо баг критичний — не забудьте одразу поділитись ним в чаті з припискою «АЛЯЯЯЯРМ!».*

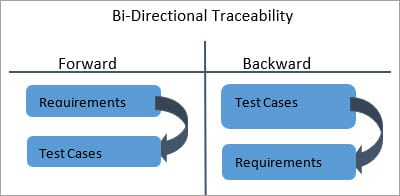
**Чеклісти** — перелік завдань на визначений термін, що потребують відмітки про виконання. Вони організовані дуже просто: складаються з переліку блоків, сторінок, які слід протестувати. Пункти повинні бути максимально короткими, однозначними та водночас зрозумілими тестувальникам, які ще, можливо, не знайомі із застосунком.

Як правило, кожен чек-лист має кілька стовпців. Кожен стовпець призначений для тестування на окремій платформі. Слід завжди вказувати назву пристрою, назву браузера та його версію. При проходженні чек-листа тестувальник зазначає статус навпроти кожного пункту, який протестовано, посилання на баг-репорти (якщо такі існують) і, за необхідності, додає примітки.

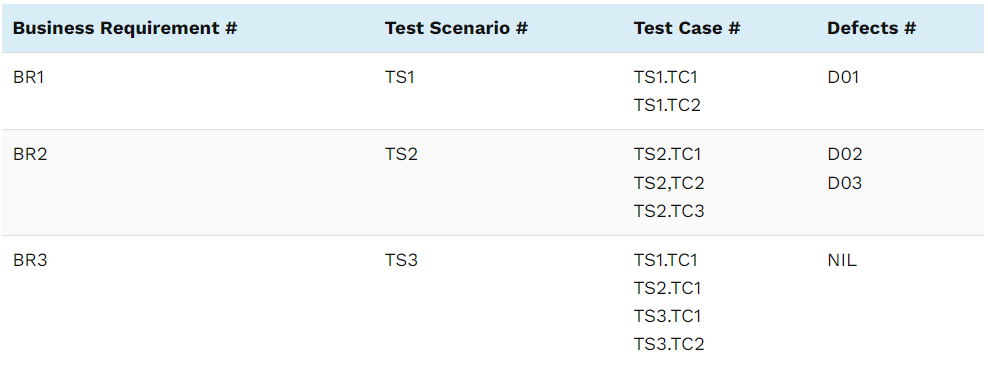


Зазвичай, чек-листи використовують, коли потрібно врахувати багато тестових випадків, а часу на опис повноцінних тест-кейсів немає або ж вимоги на проєкті змінюються так часто, що витрачати час на тест-кейси просто недоцільно.

**Матриця відповідності вимог** — таблиця, що містить відповідність функціональних вимог продукту та підготовлених тестових сценаріїв. Буває трьох типів: **Forward, Backward і Bi-Directional Traceability**.



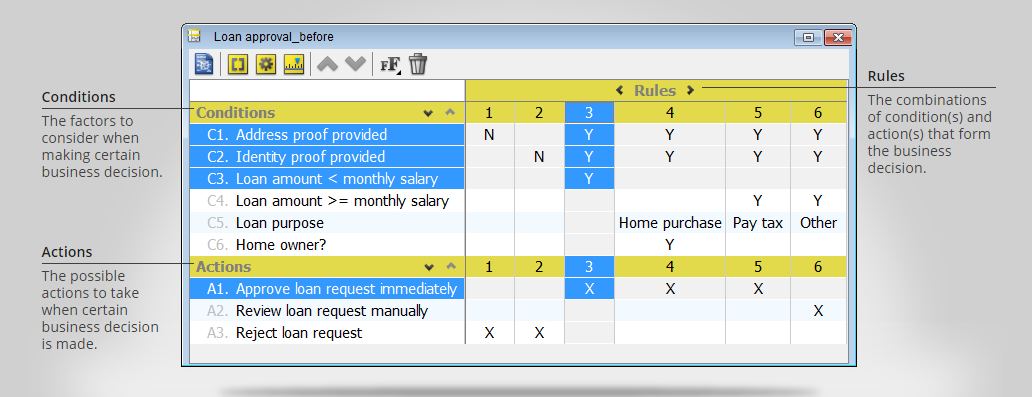
Таблиця має наступну структуру: Business Requirement #, Test Scenario #, Test Case #, Defects #.



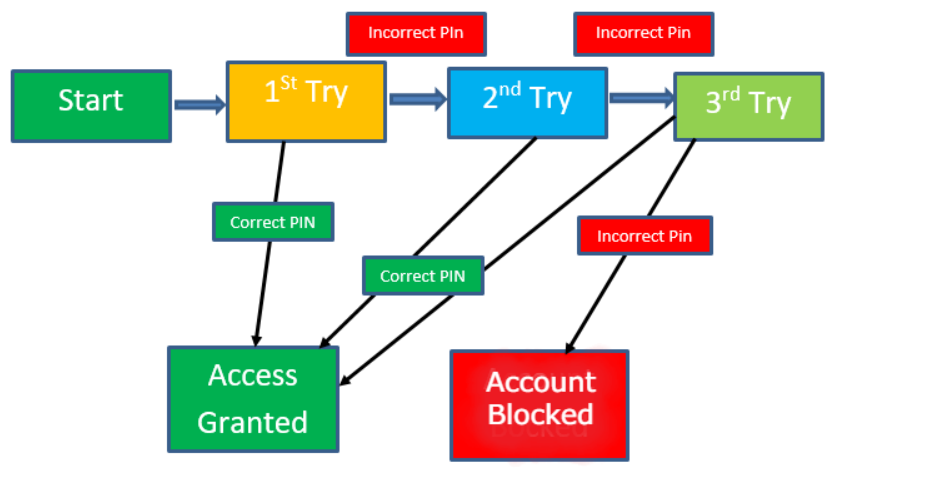
Матриця відповідності вимог дозволяє візуалізувати покриття продукту тестами та відшукати «дірки» в нашому тестовому покритті - test coverage.

**Таблиця рішень (Decision Table)**: ефективний метод тестування системи, що використовується для аналізу її відповіді на різні вхідні дані.

Як створювати: по горизонталі — вписуємо умови, що впливають на результат, а нижче дію, яку потрібно виконати. По вертикалі — правила: конкретна комбінація вхідних умов.



**Таблиця/ діаграма переходу системи (State Transition):** показує, до якого стану (станів) перейде система на основі різних комбінацій вхідних даних.



**Тестова стратегія** — це документ високого рівня, який містить вказівки та принципи, пов’язані з проведенням процесу тестування.

**Тест-план** — це документ, який описує весь обсяг робіт з тестування, починаючи з опису об’єкта тестування, стратегії, розкладу, критеріїв, початку і закінчення тестування, до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

За рівнем конкретизації завдань, тест-плани поділяються на

**Master Test Plan** i **Product Acceptance Plan** (набір дій, пов’язаних з Acceptance Testing). За посиланням нижче можна ознайомитись зі зразком вищезгаданого документа: [Test Plan Template (based on ISO/IEC/IEEE 29119)](https://studylib.net/doc/25666990/test-plan-sample--based-on-iso-iec-ieee-29119-) – англомовний зразок на 38 стор..

В чому полягає різниця між тест-планом і тестовою стратегією?

| **Атрибут** | **Тестова стратегія** | **Тест-план** |
| --- | --- | --- |
| Ціль | Визначити загальні принципи, яких слід дотримуватися під час процесу тестування. | Визначити, як тестувати, що тестувати, коли і хто тестуватиме. |
| Мета | План процесу тестування на довгостроковій основі. | Створюється для того, щоб виявити можливі невідповідності в кінцевому результаті. |
| Повторюваність | Може використовуватись багатьма проєктами. | Використовується лише одним проєктом. |
| Можливість внесення змін | Статичний документ. | Гнучкий документ. |
| Компоненти | Обсяг, цілі, формат документації, тестові процеси, звіти (репортинг) тощо. | ID, перелік того, що тестувати, тест-техніки, Pass/Fail критерії, результати тестування, розклад тощо. |
| Існування | Може бути частиною тест-плану. | Існує окремо. |

### Який мінімум тестової документації очікується на проєкті

Насправді відповідь варіюється в залежності від розмірів, специфікації, рівня складності та розгалуженості тощо. Зазвичай: тест-кейси й відмінні баг-репорти це, в нашому випадку, два кити, на яких будується ефективна діяльність тестувальника. Будь-хто, приєднавшись до команди, розумітиме принцип функціонування застосунку; будь-хто, взявши в роботу баг-репорт, знатиме, яким чином відтворити проблему і в чому полягає відхілення від очікуваного результату.

Коли ми визначаємо кількість тест-артефактів, необхідних нашому проєкту, потрібно орієнтуватись на наступне:

1. Перегляньте існуючу документацію. Це допоможе зрозуміти обсяг тестування і типи тестових одиниць, які знадобляться.
2. Виходячи з вимог проєкту та ризиків, пов’язаних з використанням продукту, визначте типи тестів, які потрібно буде виконати, наприклад, юніт, інтеграційні, системні, приймальні, регресійні тести тощо.
3. Визначте необхідний рівень деталізації: залежно від складності бізнес-логіки, технічної складності проєкту (продукту) та необхідного рівня деталізації може знадобитися створити різноманітні тестові документи, такі як тест-плани, тестові сценарії, підготувати тестові дані.
4. Визначте тестове середовище, необхідне для проєкту, як-от обладнання, програмне забезпечення та мережеві конфігурації, і переконайтеся, що тестові документи відображають необхідні вимоги до середовища.
5. Задокументуйте процес тестування, включаючи методологію тестування, інструменти та методи, що використовуються, і створіть підсумкові звіти про тестування, щоб надати зацікавленим сторонам огляд витрачених ресурсів щодо тестування та досягнутих результатів.
6. Звучить боляче, але не забувайте оновлювати документацію при будь-яких змінах у влаштуванні проєкту.

Відсутність тестової документації, своєю чергою, передбачає наявність негативних наслідків. А саме: підвищений ризик присутності помилок у кінцевому продукті; зниження ефективності роботи; невиправдані витрати, якщо дефекти виявлені на пізніх стадіях розробки — йдуть в комплекті з невдоволенням замовника, несхвальним відгуком та заплямованою репутацією компанії.

Підсумовуючи вищесказане, тестові артефакти є критично важливим аспектом розробки програмного забезпечення, і нехтування їх створенням може призвести до появи несприятливого проєктного середовища.

## Найкраща практика для отримання тестової документації

* Команда контролю якості повинна бути залучена до початкової фази проекту, щоб тестова документація створювалася паралельно
* Не просто створюйте та залишайте документ, а оновлюйте його щоразу, коли потрібно
* Використовуйте контроль версій, щоб керувати документами та відстежувати їх
* Спробуйте задокументувати те, що вам необхідно для розуміння вашої роботи, і те, що вам потрібно буде надати своїм зацікавленим сторонам
* Ви повинні використовувати стандартний шаблон для документації, як-от файл Excel або doc
* Зберігайте всі документи, пов’язані з проектом, в одному місці. Він має бути доступним для кожного члена команди для довідки, а також для оновлення за потреби
* Ненадання достатньої кількості деталей також є типовою помилкою під час створення тестового документа

## Переваги тестової документації

* Основна причина створення тестової документації полягає в тому, щоб зменшити або усунути будь-які невизначеності щодо діяльності тестування. Допомагає усунути неоднозначність, яка часто виникає під час розподілу завдань
* Документація пропонує не лише систематичний підхід до тестування програмного забезпечення, але він також виступає як навчальний матеріал для новачків у процесі тестування програмного забезпечення
* Це також хороша стратегія маркетингу та продажів, щоб продемонструвати тестову документацію, щоб продемонструвати зрілий процес тестування
* Тестова документація допомагає запропонувати клієнту якісний продукт у визначені терміни
* В розробці програмного забезпечення, тестова документація також допомагає налаштувати або налаштувати програму за допомогою документа конфігурації та посібників оператора
* Тестова документація допомагає вам підвищити прозорість із клієнтом

## Недоліки тестової документації

* Вартість документації може перевищувати її вартість, оскільки це займає багато часу
* Часто її пишуть люди, які не вміють добре писати або не знають матеріалу
* Відслідковувати зміни, які вимагає клієнт, і оновлювати відповідні документи втомлює.
* Погана документація безпосередньо відображає якість продукту, оскільки може виникнути непорозуміння між клієнтом і організацією

**Питання для самоконтролю:**

### Які основні компоненти формату звіту про дефекти?

### Що таке тестовий результат?

1. **Що таке звіт про помилку?**
2. **Що таке Тест план і check-list і для чого вони використовуються?**
3. **Визначте відміну Use case від test case**