**Лекція 18 Тестова документація**

Програма повного тестування може допомогти виявити проблеми до того, як вони проявлять себе в експлуатованому програмному продукті. Крім того, відповідна документація необхідна тим, хто проводить тестування, аби було зрозуміло, що потрібно перевіряти. Тому, такі правила повинні починатися з вимог щодо тестування і документування. Формулювання може виглядати таким чином:

*Всі власні розробки мають бути протестовані і задокументовані перед здачею їх в промислову експлуатацію.*

Це формулювання встановлює вимогу, але не тип тестування або формат документації. Дану пропозицію необхідно включити в правила для всього програмного забезпечення і в процедури.

Важливим аспектом процесу є визначення персоналу тестування, і тут важливим є те, що Незалежний **тестувальник часто може побачити більше**, інших і різних дефектів, ніж тестувальник, який працює в команді програмістів.

Бізнес-аналітики, спеціалісти з маркетингу, дизайнери та програмісти вносять власні припущення в специфікацію та реалізацію тестованого елемента, **незалежний тестувальник вносить інший набір припущень** у тестування та огляди, що часто допомагає виявити приховані дефекти та проблеми, пов’язані з спосіб мислення групи.

Незалежний **тестувальник має скептичне ставлення до професійного песимізму**, відчуття того, що, якщо є сумніви щодо спостережуваної поведінки, вони повинні запитати: чи це недолік

Для визначення потреб персоналу для тестування потрібно зважати на такі аспекти.

**Додаток або бізнес-сфера:** тестувальник повинен розуміти заплановану поведінку, проблему, яку система вирішить, процес, який вона автоматизує.

**Технологія:** тестувальник повинен знати про проблеми, обмеження та можливості вибраної технології впровадження, щоб ефективно та результативно виявляти проблеми та розпізнавати функції та особливості , які «імовірно не вдасться ».

**Тестування:** тестувальник повинен знати тестування, щоб ефективно та результативно виконувати поставлені тестові завдання. Специфічні навички в кожній галузі та необхідний рівень навичок залежать від проекту, організації, програми та пов’язаних із цим ризиків.

**Тестова документація** – це документація артефактів, створених до або під час тестування програмного забезпечення. Вона допомагає команді тестування оцінити необхідні зусилля для тестування, покриття тесту, відстеження ресурсів, прогрес виконання. Тестова документація робить планування, перевірку та виконання тестування легкими, а також такими, що їх можна перевірити. Склад тестової документації представлено на рис.1.



Рисунок 1 - Склад тестової документації

Ця документація повинна включати питання:

* **Бізнес-цілі** — у цьому розділі записуються бізнес-цілі та ключові показники ефективності ((key performance indicators, KPI). Описує, як проект приносить прибуток і які технології використовуються для отримання бажаних результатів.
* **Бізнес-модель** — цей розділ описує продукт з точки зору клієнта — включаючи бажаний досвід, потреби, цілі та переваги від використання програмного забезпечення. Складається з діаграм, статистики та графіків.
* **Технічні умови** —функціональність технічного середовища проекту, пристроїв, операційних систем, вимог до апаратного забезпечення тощо та перевіряйте, чи добре працює функціональність продукту за цих умов.
* **Характеристики системи** — ці звіти визначають зручність використання, безпеку, доступність, підтримку між платформами.

Документація поділяється на групи відповідно до цільового використання: внутрішнього та зовнішнього.

**Документація внутрішнього тестування програмного забезпечення**

* + Тест стратегії
  + Тестові дані
  + Плани випробувань
  + Тестові сценарії
  + Тестові випадки
  + Матриця відстеження

**Документація зовнішнього тестування**

* + Зовнішні звіти
  + Підсумковий звіт про тестування
  + Звіти про помилки

Розглянемо в загальних рисах та пригадаємо призначення наданих на рис.1 тестових документів.

**Тест стратегії**

* Документ високого рівня, який визначає рівні (типи) тестування, які потрібно виконати для проекту.
* Опис повного підходу до тестування продукту.
* По ходу проекту розробники, дизайнери, власники продукту можуть повертатися до документа та перевіряти, чи фактична продуктивність відповідає запланованим діям.

При створенні тестової стратегії виконуються такі кроки:

1. ***Збір інформації*** *.* Вивчіть продукт і його контекст; отримати інформацію з: бесід із зацікавленими сторонами, дослідницького тестування, читання документації.
2. ***Аналіз інформації*** *.* Виберіть найбільш важливу інформацію.
3. ***Прийняття рішень*** *.* Приймайте рішення про те, як тестувати продукт. Набір цих рішень ляже в основу тестової стратегії.
4. ***Презентація*** *.* Поясніть рішення та обґрунтуйте їх правильність.

Склад документу надано на рис.2.

**Тестовий план**

Тестовий план - документ, який описує стратегію, ресурси, середовище, обмеження та графік процесу тестування. Це найповніший тестовий документ, необхідний для обґрунтованого планування. Такий документ поширюється між членами команди та надається всім зацікавленим сторонам.

План тестування — це детальний документ, який описує стратегію тестування, цілі, графік, оцінку, результати та ресурси, необхідні для тестування програмного продукту.

**План тестування** — це план проекту для виконання тестової роботи.

Це не специфікація дизайну тесту, збірка тестів або набір тестових процедур

**Мета процесу тестування** – це відповіді на запитання.

* Що входить у сферу дії, а що виходить за рамки цього тестування?
* Які цілі тесту?
* Які важливі ризики проекту та продукту?
* Які обмеження впливають на тестування (наприклад, бюджетні обмеження, жорсткі дедлайни тощо)?
* Що є найбільш критичним для цього продукту та проекту?
* Які аспекти продукту більше (чи менш) тестуються?
* Яким має бути загальний розклад виконання тестів і як ми маємо визначити порядок виконання конкретних тестів?

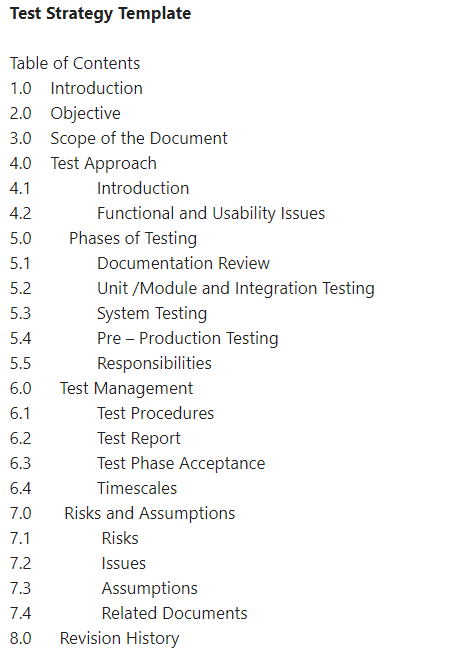
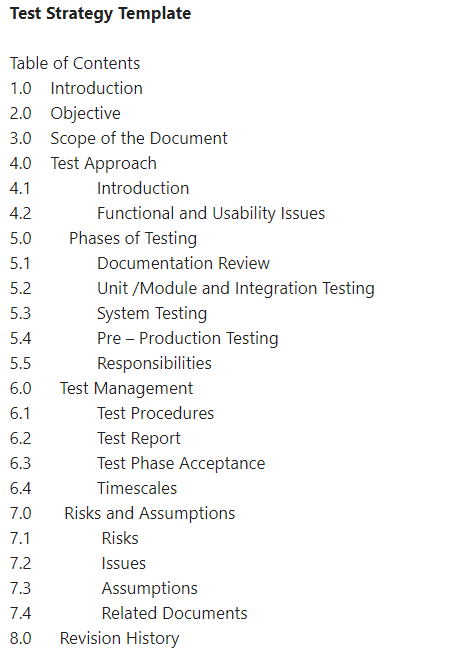


Рисунок 2 – Склад документу Тестова стратегія

**Зміст тестового плану** повинен надавати відповіді на ці запитання, відповідно, повинен висвітлювати .

* + Визначення обсягу, цілей і ризиків тестування
  + Визначення загального підходу до тестування
  + Інтеграція та координація тестової діяльності в діяльність життєвого циклу програмного забезпечення
  + Ресурси, необхідні для виконання різних тестових дій, і те, як тестування здійснюватиметься
  + Планування аналізу, проектування, впровадження, виконання та оцінювання тестів; Вибір метрик для моніторингу та контролю тестування
  + Бюджетування тестової діяльності
  + Визначення рівня деталізації та структури тестової документації

За стандартами перелік позицій тест-планів наданий на рис.3.

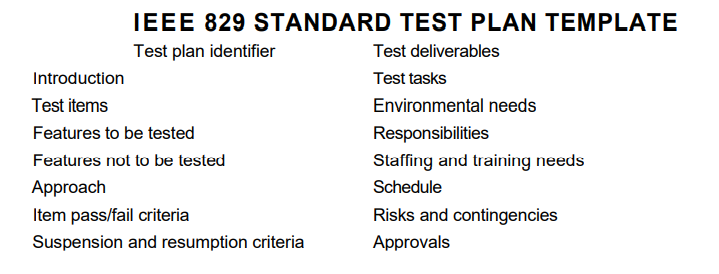


Рисунок 3 – Склад документу Тестовий план

**Види тест- плану**

* **Master Test Plan** - має кілька рівнів тестування. Він включає повну стратегію тестування.
* **План фазового тестування** – стосується будь-якої фази стратегії тестування. Наприклад, список інструментів, список тестів тощо.
* **Спеціальні плани тестування** — призначені для основних типів тестування, наприклад тестування безпеки, тестування навантаження, тестування продуктивності тощо. Іншими словами, спеціальний план тестування, призначений для нефункціонального тестування.

Всі тест-плани мають атрибути (див. рис.4).

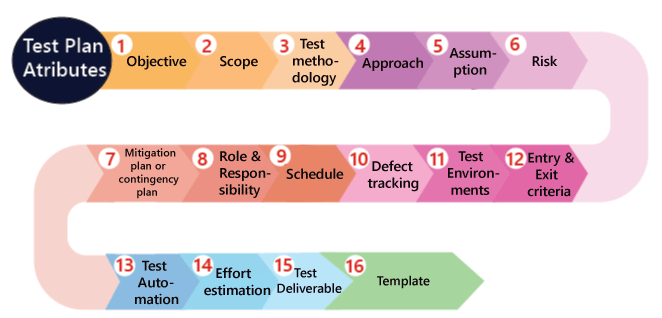


Рисунок 4 – Атрибути документу Тестовий план

Тестовий план повинен визначати критерії входження та виходу з процесу тестування. Фактори, які слід враховувати при прийнятті таких рішень, часто називають **«критеріями входу» та «критеріями виходу».** Для таких критеріїв типовими факторами є:

* **Придбання та постачання** : наявність персоналу, інструментів, систем та інших необхідних матеріалів.
* **Елементи тестування:** стан, у якому мають перебувати елементи для тестування, щоб розпочати та завершити тестування.
* **Дефекти:** відома кількість наявних, швидкість прибуття, прогнозована кількість, яка залишиться, і кількість вирішених.
* **Тести:** номер запущений, пройдений, невдалий, заблокований, пропущений...
* **Покриття:** частини бази тестування, програмного коду або обох, які були протестовані та які ні.
* **Якість:** статус важливих характеристик якості для системи.
* **Гроші:** вартість пошуку наступного дефекту на поточному рівні тестування порівняно з вартістю його пошуку на наступному рівні тестування (або у виробництві).
* **Ризик:** небажані наслідки, які можуть виникнути внаслідок надто раннього відправлення (наприклад, приховані дефекти чи неперевірені ділянки) або надто пізнього (наприклад, втрата частки ринку).

**Тестовий сценарій**

Тестовий сценарій — це елемент або подія програмної системи, які можуть бути перевірені за допомогою одного або кількох тестових випадків. У сценаріях тестувальники розбивають функціональні можливості та інтерфейс продукту за модулями та забезпечують оновлення статусу в реальному часі на всіх етапах тестування. Модуль може бути описаний одним оператором або вимагати сотні статусів, залежно від його розміру та сфери застосування.

Тестовий сценарій визначається як будь-яка функція, яку можна перевірити. Як тестувальник ви можете поставити себе на місце кінцевого користувача та з’ясувати реальні сценарії та випадки використання програми, що тестується. Тестовий сценарій допомагає визначити найважливіші наскрізні транзакції або реальне використання програмного забезпечення. Приклад тестового сценарію надано на рис.5.

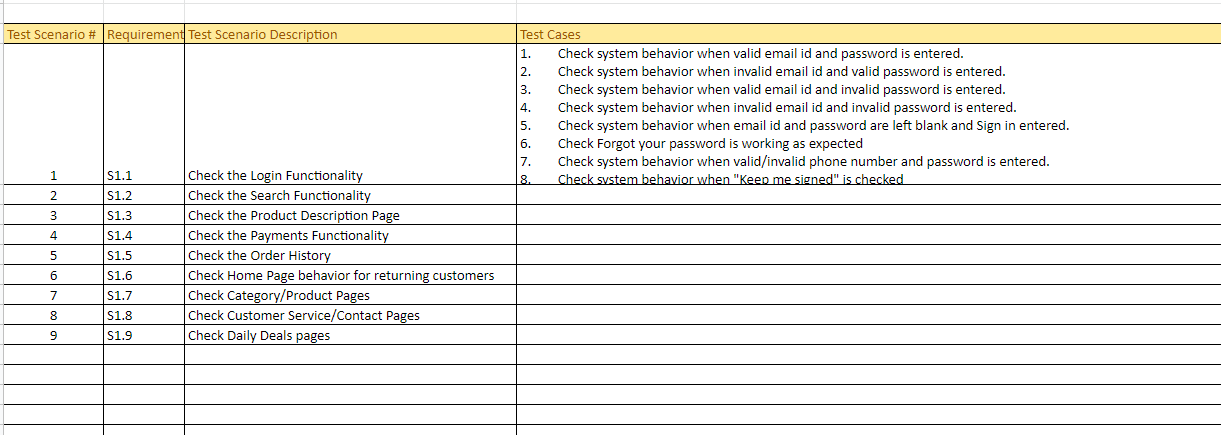


Рисунок 5 – Приклад тестового сценарію

**Тестові випадки**

***Test Case*** *— це набір умов або змінних, за яких тестувальник визначатиме, чи працює програма, програмна система або одна з її функцій, як це було спочатку встановлено.*

Тестовий випадок (test case) - група вхідних значень, попередніх умов виконання, очікуваних постумов виконання та результатів. Тест випадок розроблений для тестового сценарію. Якщо тестовий сценарій описує об’єкт тестування (що), сценарій описує процедуру (як).

Ці документи охоплюють покрокові вказівки, детальні умови та поточні вхідні дані тестового завдання. Тестові випадки мають свої види, які залежать від типу тестування — *функціональні, UI, фізичні, логічні випадки тощо* .

Тестові випадки порівнюють доступні ресурси та поточні умови з бажаними результатами та визначають, чи можна запустити функціональність чи ні.

***Під час написання тестів враховуеться наступне:***

* *Нехай заголовок буде коротким.*
* *Додайте чіткий опис.*
* *Будьте чіткими та лаконічними.*
* *Включіть очікуваний результат.*

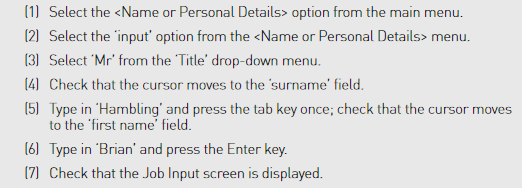


Рисунок 6 – Приклад тесту

Атрибути Тестового випадку:

|  |  |
| --- | --- |
| **ID набору тестів** | Ідентифікатор набору тестів, до якого належить цей тест. |
| **Ідентифікатор тестового випадку** | Ідентифікатор тесту. |
| **Підсумок тесту** | Короткий зміст / мета тесту. |
| **Пов’язана вимога** | Ідентифікатор вимоги, до якої відноситься/відстежується цей тестовий приклад. |
| **передумови** | Будь-які передумови або попередні умови, які мають бути виконані перед виконанням тесту. |
| **Процедура тестування** | Покрокова процедура виконання тесту. |
| **Тестові дані** | Тестові дані або посилання на тестові дані, які будуть використовуватися під час проведення тесту. |
| **Очікуваний результат** | Очікуваний результат тесту. |
| **Статус** | Пройшов або не пройшов. Інші статуси можуть бути «Не виконано», якщо тестування не виконується, і «Заблоковано», якщо тестування заблоковано. |
| **Зауваження** | Будь-які коментарі щодо тесту або виконання тесту. |

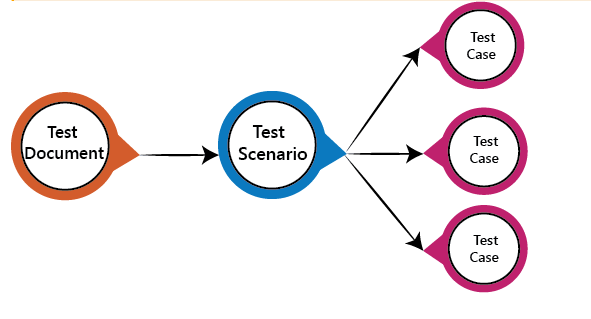


Рисунок 7 – Місце тестових прикладів в загальному процесі тестування

***Контрольний список/лист (Checklist)***

***Контрольний список – це каталог елементів/завдань, які записуються для відстеження. Цей список може бути або впорядкованим у послідовності, або може бути випадковим.***

**Навіщо нам потрібні контрольні списки?** : Для відстеження та оцінки завершення (або незавершення). Записувати завдання, щоб нічого не пропустити.

**Як ми створюємо контрольні списки?:** Ну, це не може бути простіше. Просто запишіть все по пунктах.

***Контрольні списки можна створювати на основі досвіду, знань про те, що важливо для користувача, або розуміння того, чому та як програмне забезпечення дає збій.***

Тестування на основі контрольного списку виконується за схемою.

1. Зробіть кожен пункт чітким і лаконічним.
2. Згрупуйте свої предмети за категоріями.
3. Зробіть кожен елемент дієвим.
4. Не пропускайте нічого зі списку.
5. Переконайтеся, що ваші нотатки, докази чи інші результати використовуються для покращення продуктивності.

***Склад контрольного списку***.

* Загальна інформація (наприклад, посилання та дані для входу на веб-сайт).
* Назва розділу (підрозділу).
* Список перевірок (з необхідним рівнем деталізації)ю
* Вид перевірки (позитивна, негативна).
* Тестові дані.
* Очікувана поведінка системи.
* Інформація про тестове середовище: пристрій, браузер, роздільна здатність екрана тощо.
* Результат перевірки (пройшов, не пройшов, не виконано).
* Номер помилки або посилання на неї (якщо тест не пройдено).

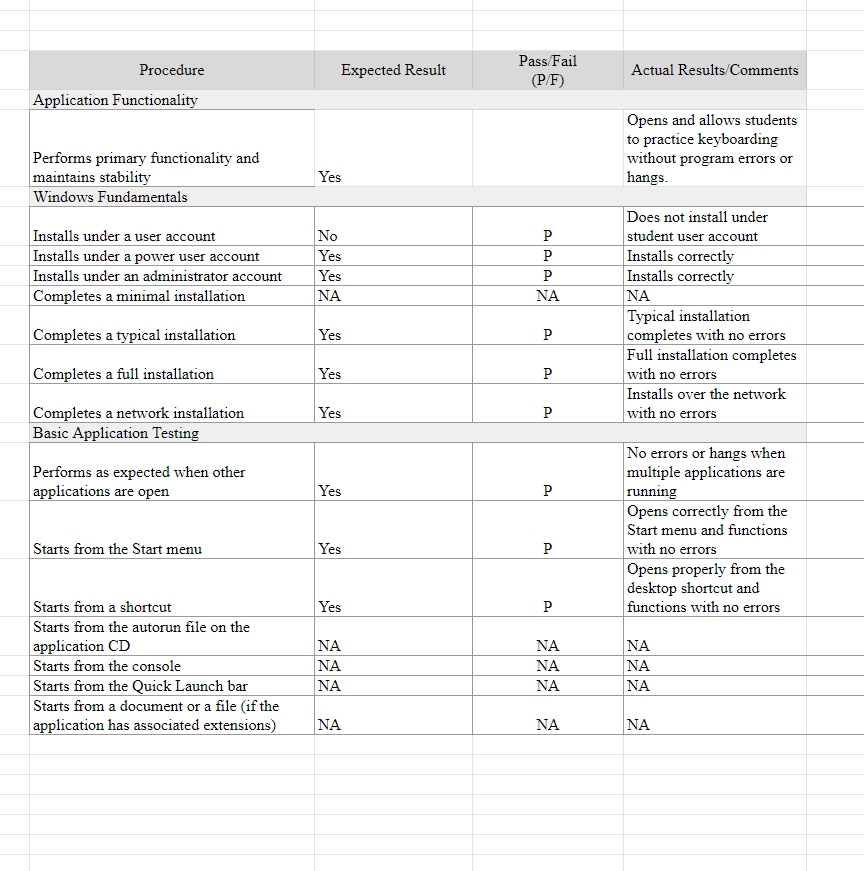


Рисунок 8 – Приклад контрольного списку тестування

## Баг-репорт

Bug report — це технічний документ, який містить повний опис дефекту/багу з інформацією як про саму помилку (короткий опис, серйозність, пріоритет тощо), так і про умови її виникнення. Баг-репорт має містити правильну єдину термінологію, що описує елементи призначеного для користувача інтерфейсу і події, що спричиняють баг.

Складники баг-репорту:

* короткий опис проблеми;
* назва проєкту;
* компонент додатка, в якому виник дефект;
* версія версії/білду, в якому знайдено баг;
* серйозність (градація ступеня впливу на додаток);
* пріоритет (черга виправлення помилки);
* статус (помилки в життєвому циклі);
* автор (хто створив баг-репорт);
* призначення (хто повинен виправити дефект).

Сам опис багу може містити такі частини:

* передумови (опціонально);
* кроки відтворення (опис шляху, який призводить до виникнення дефекту);
* фактичний результат (результат, до якого приходимо, коли виконаємо всі кроки);
* очікуваний результат (результат, який мав бути відповідно до вимог).

Корисно додавати до документа скриншоти, тестові файли або інші документи, що допоможуть програмістам розв’язати проблему. Бажано відразу описувати баг детально, коректно та зрозуміло, щоб розробникам доводилося менше ставити уточнювальних питань чи взагалі ставити статус *Can’t reproduce*.

# Матриця відстеження/простежуваності вимог (Requirements Traceability Matrix - RTM)

Ця документація з тестування програмного забезпечення відображає тестові випадки та їхні вимоги. Усі записи мають власні ідентифікатори — члени команди та зацікавлені сторони можуть відстежувати хід виконання будь-яких завдань, просто ввівши ідентифікатор у пошуковий запит. Кожна вимога в документі RTM пов’язана з пов’язаним із нею тестовим прикладом, щоб можна було проводити тестування відповідно до згаданих вимог

**Основні цілі матриці**

* Переконайтеся, що програмне забезпечення розроблено відповідно до зазначених вимог.
* Допомагає знайти першопричину будь-якої помилки.
* Допомагає відстежувати розроблені документи на різних етапах SDLC

Traceability matrix — подвійна таблиця, що перевіряє відповідність функціональних вимог продукту (functional requirements) і підготовленим тест-кейсам. На перетині відповідних рядка та стовпця ставиться відмітка, яка позначає, що ця вимога покривається тест-кейсом.

Таким чином цей інструмент дає візуальне зображення таких параметрів:

* наявність у системі ще не покритих тест-кейсами вимог (якщо у вимог немає жодного перетину з тест-кейсами);
* надлишкове тестування — якщо вимога має кілька перетинів з тест-кейсами.

Що має містити матриця покриття вимог? У кожному рядку має бути:

* номер та опис задачі з таск-трекера;
* модуль, до якого належить задача (опціонально);
* приймальний критерій (Acceptance criteria);
* пріоритет;
* номер та опис відповідного тест-кейсу.

Матриця коректно виконує свою роль лише за умови її постійної актуалізації. Інакше вона не тільки стає марною, а й може заплутувати тестувальників.

На рис.9 надано зразок матриці відстеження.

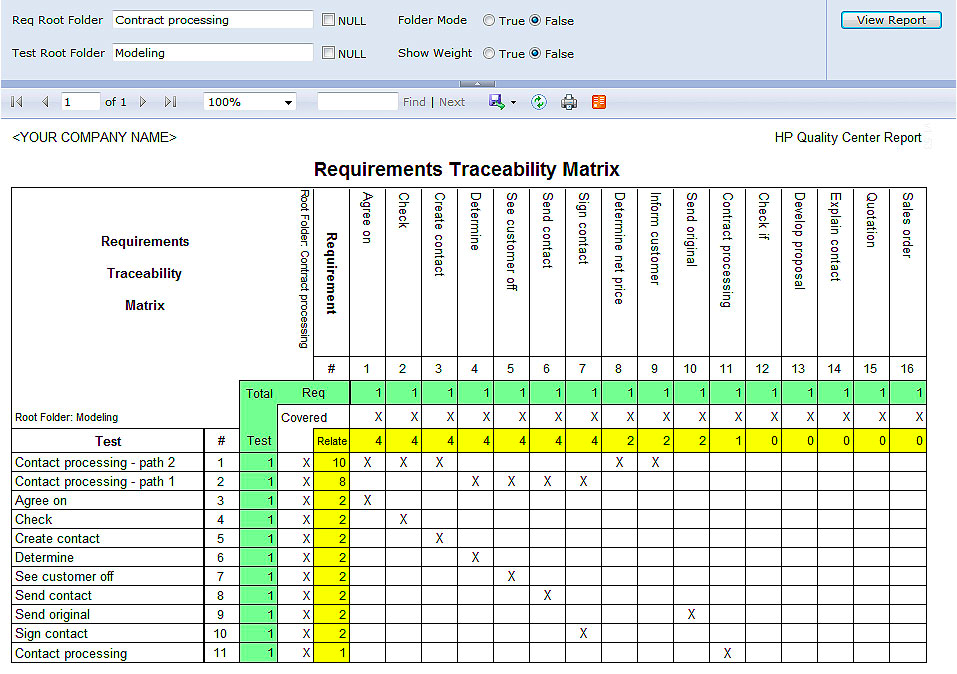


Рисунок 9 - Приклад матриці відстеження

**Таблиця рішень (Decision Table**

**Таблиця рішень -** 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.

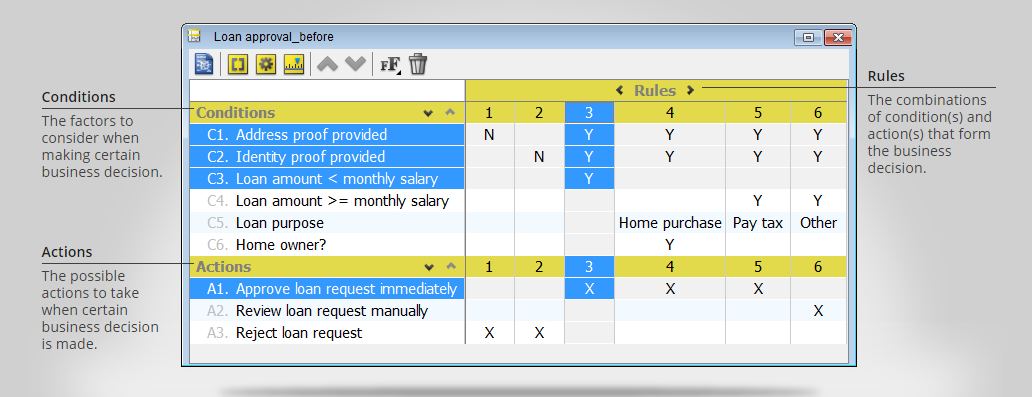


Рисунок 10 - Приклад Таблиці рішень

**Таблиця/ діаграма переходу системи (State Transition)**

**Діаграма переходу системи (приклад див. рис.11)** показує, до якого стану (станів) перейде система на основі різних комбінацій вхідних даних.

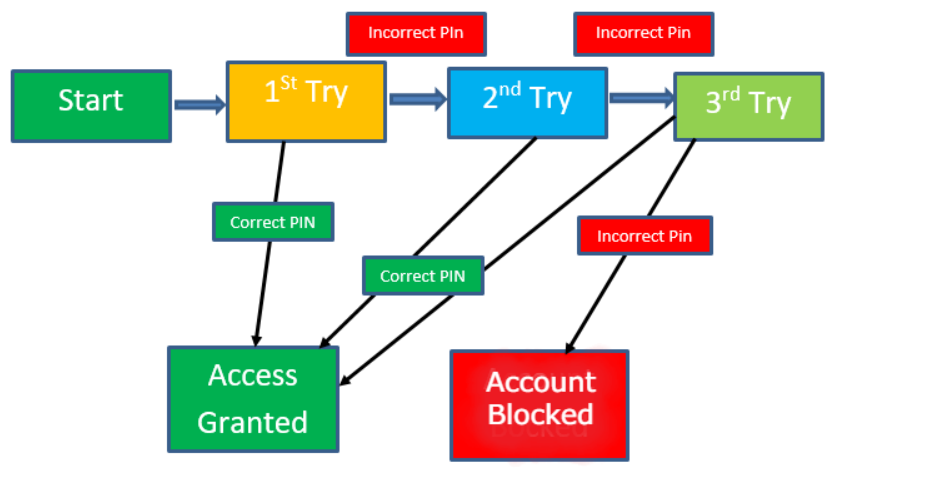


Рисунок 10 – Приклад **діаграми переходу системи**

## 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 відповідної експлуатаційної документації.
* 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

## Висновки

* Тестова документація — це документація артефактів, створених до або під час тестування програмного забезпечення.
* Ступінь формальності тестування залежить від 1) типу програми, що тестується 2) стандартів, яких дотримується ваша організація 3) зрілості процесу розробки.
* Важливими типами тестових документів є політика тестування, стратегія тестування, план тестування, тестовий приклад тощо.
* Команда контролю якості повинна бути залучена до початкової фази проекту, щоб тестова документація створювалася паралельно
* Основна причина створення тестової документації полягає в тому, щоб зменшити або усунути будь-які невизначеності щодо діяльності тестування.
* Вартість документації може перевищувати її вартість, оскільки це займає багато часу

**Питання для самоконтролю:**

**1.** Особливості документування.

2.   Яким чином проводиться генерування даних.

**3.** Якзабезпечити конфіденційність даних?

**4.** Які вимоги висуваються до документації?