# 1. Склади порівняльну таблицю функціонального, нефункціонального і пов’язаного зі змінами видів тестування.

# 

| **блоки порівняння** | **функціональне**  **тестування** | **нефункціональне тестування** | **тестування пов’язане зі змінами** |
| --- | --- | --- | --- |
| **що перевіряється** | перевіряється відповідність ф-х вимог ПЗ його реальним х-м. перевіряє те, чи володіє продукт потрібним для замовника функціоналом | перевірення неф-х вимог системи, таких як продуктивність, масштабованість, зручність використання, надійність, тощо. | це перевірка змін після виявлення чи усунення багів, а також, якщо замовник захоче щось змінити у ПЗ |
| **коли застосовується** | коли треба протестувати функціональні х-ки додатка та перевірку на невідповідності між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації | застосовується тоді, коли треба перевірити зовнішний вигляд ПЗ, його надійності та продуктивності, а також зручності у використанні кінцевим користувачем | застосовується тоді, коли команда вносить кісь зміни у роботу, зовнішній вигляд, тощо, ПЗ. |
| **обмеження** | є ймовірність надмірного тестування, а також можливість упущення логічтисних помилок у ПЗ | ти можеш побачити недолік у дизайні та сказати про неї, але якщо користувач саме так і хотів (типу фіолетовий текст на чорному фоні) - ти нічого не зможеш вдіяти і будеш обмежен у цьому | можна проводити тільки після затвердження якоїсь зміни. перевіряєш тільки забаговану ф-ію |
| **особливості** | для якісного тестування потрібне кокретне оформлення вимог і обмежень о якості, та конкретний опис моделі функціонування пз, який підійде замовнику.  імітує фактичне користування системою | характеризує систему з точки зору зручності використання її кінцевим користувачем.  є оцінка реакцї захисних властивостей програми. допомагає оцінити зовнішній вигляд програми, якщо це треба. | треба дивитися за тим, чи не зламало виправлення забагованної ф-її іншу працюючу ф-ю ПЗ. рекомендується робити автоматизацію регресійних тестів, для прискорення подальшого процесу тестування і виявлення дефектів на ранніх стадіях розробки програмного забезпечення. |

# 2. Поясни, в чому різниця між регресією та ретестингом (5 речень).

регресія - це дія, спрямована на виявлення помилок у вже протестованих версіях ПЗ. для пошуку і виправлення саме регресивних помилок, які з’являються при додаванні чогось нового у вже існуючий білд, або, також, при виправленні вже знайдених багів.

а ретестинг - це перевірка вже знайдених і виправленних багів. чи їх справді виправили і тепер працює та ф-я, котра була забаговна. цей тест можно проводити неодноразово, щоб впевнитися, що ф-я працює правильно

основна різниця заключається у тому, що ретестинг не спрямований на пошук регресивних помилок, які міг спрямувати вже виправленний баг, а просто перевіряє саме цю виправленну функціональність.

# 3. Як ти вважаєш, чи можливе для продукту проведення тільки функціонального тестування, без перевірки нефункціональних вимог?

мені здається, що таке тестування можливе, але дуже недоречне.

тому що усі функціональні вимоги можуть працювати дуже добре і все буде класно, а нефункціональні просто будуть заважати роботі сайту (якщо там будуть якісь серйозні баги). наприклад, ось нефункціональна вимога: зовнішній вигляд сайту - також важлива частина, котру треба протестувати. бо, грубо кажучи, ніхто не буде і не зможе користуватися супер ідеально працюючим сайтом, якщо у нього чорні букви написані на чорному фоні). чи сайтом, котрий прогружає одну сторінку 30 хвилин.

# 4. Як ти розумієш необхідність проведення smoke (димового) тестування? Чи завжди воно є доречним?

Димове тестування це вид тестуванння який використовується для виявлення помилок, котрі можуть з’явитися на самих ранніх етапах роботи ПЗ.

це короткий цикл тестів, котрий проводиться після кожного нового випуску білду. виконується для перевірки основного ф-лу. після удачної перевірки цього ф-лу система переходить вже до тестування більш серйозними видами тестів.

також воно добре піддається автоматизації.

я вважаю, що димове тестування необхідне для того, щоб попередити виявлення самих великих проблем одразу після випуску білда, бо якщо програма не пройде димове тестування, тестувати її далі просо немає сенсу, бо воно може бути просто неможливим. це дуже економить час, якщо все ж таки знайдено багу. я вважаю що димове тестування є доречним завжди, особливо якщо його автоматизувати - багато часу воно не займе.

# 4. Ти – засновник/ця стартапу і плануєш випустити на ринок мобільний застосунок для обміну світлинами котиків для iOS та Android пристроїв.

# Користувачі можуть завантажувати фотографії котиків. Але не можуть завантажувати фотографії інших тварин/людей/об’єктів. Користувачі можуть додавати друзів, ставити “вподобайки”, залишати коментарі.

### Завдання: Напиши 5 функціональних тест-кейсів, які перевіряли б роботу застосунку.

[тест кейс 1.1](https://docs.google.com/spreadsheets/d/1YNungrU_R2wSJ3_0SVzmoO2xUtQ0Cbp-ZoU8b66qaWM/edit?usp=sharing)

[тест кейс 1.2](https://docs.google.com/spreadsheets/d/1HrpIDYIq2fl1yQB8zEpQyAhe0zkirLfiwcTAM7jhpjE/edit?usp=sharing)

[тест кейс 1.3](https://docs.google.com/spreadsheets/d/11w3I6GG7kopbdJZPSfYMPZ_-zPhZu0XxPVdG-YypK28/edit?usp=sharing)

[тест кейс 1.4](https://docs.google.com/spreadsheets/d/1Zj6av6WC_MqmsUYkgc4o25v5ML6XpsvUiJBDs9wTp_A/edit?usp=sharing)

[тест кейс 1.5](https://docs.google.com/spreadsheets/d/1PXjZmdL7eOiZP6LHYAWBivZx8J8ypJbpsr-mgmn5eek/edit?usp=sharing)

# 5. Напиши, які нефункціональні вимоги ти хотів/ла б застосувати для продукту твого стартапу.

## Опиши перевірки, які б їх перевіряли (3-5 прикладів).

1. щоб система опрацьовувала запит не повільніше 3-х секунд (секундомер чи спеціальна програма що зчитує час опрацювання запросу)
2. коли приходить сповіщення лунає звук “мяу” (треба щоб на телефоні був увімкнутий звук та дозволені сповіщення від стосунку, тоді, коли на аккаунт підписався інший користувач, або хтось поставив лайк світлині - буде чутно звук сповіщення)
3. червоно білий дизайн у днівній версії та сіро фіолетовий у нічній (подивитися чи влаштовує дизайн, чи зручний він у користуванні)
4. екран загрузки у вигляді заповнення лапки котика кольором (закрити застосунок повністю, а потім відкрити, та подивитися на екран загрузки)