**Перший рівень — відпрацюй навички на базовому рівні.**

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

|  | Функціональне тестування | Нефункціональне тестування | Пов'язане зі змінами тестування |
| --- | --- | --- | --- |
| Що перевіряється | Перевіряє функцію пристрою, гарантуючи, що вхідні дані дають єдиний бажаний вихід.  Забезпечує початкову перевірку стабільності нової збірки програмного забезпечення та її критично важливої функціональності.  Включає тестування елементів керування інтерфейсом, таких як кнопки, меню та введення тексту, щоб переконатися, що потік взаємодії та вибрані функції є оптимальними для користувацького досвіду.  Перевіряє правильну роботу програмного забезпечення, коли компоненти об’єднуються і працюють як єдине ціле.  Включає: Unit testing, Component testing, Integration testing, API testing, UI testing, System testing,White-box testing, Black-box testing, Acceptance testing, Alpha testing, Beta testing, Production testing | Наскільки зручним є додаток для користувачів з обмеженими можливостями, наприклад, з вадами зору.  Наскільки часто додаток доступний і готовий до використання.  Тестує відновлення критично важливих для бізнесу додатків у надзвичайних ситуаціях.  Тестує додаток під великим навантаженням протягом тривалого періоду часу.  Тестує систему резервного копіювання додатку в разі збою системи.  Тестує сценарії на основі місцезнаходження в додатку.  Перевіряє здатність додатку оновлюватися.  Перевіряє, як додаток переноситься з одного програмного або операційного середовища в інше.  Тестує механізми безпеки додатку для виявлення вразливостей.  Перевіряє зручність використання програми.  Включає: Performance Tests, Load Tests, Stress Tests, Volume Tests, Security Tests, Upgrade & Installation Tests, Recovery Tests | Зазвичай виконується після отримання досить стабільної збірки програмного забезпечення або іноді, коли збірка програмного забезпечення могла зазнати незначних змін у коді або функціоналі.  Перевіряє чи раніше усунуті помилки були виправлені, і відловити помилки, які могли випадково з'явитися в новій версії.  Включає: Smoke Testing, Regression Testing, Build Verification Test, Sanity Testing |
| Коли застосовується | Коли тестується нова система.  Коли додаток перевіряється на відповідність його специфікаціям та бажаним вимогам.  Щоб перевірити, чи добре працює інтеграція системи та модулів.  Коли потрібно перевірити функціональність системи в цілому.  для визначення робочого процесу системи та її функцій.  Щоб переконатися, що наскрізні потоки дають очікуваний результат. | Коли потрібно перевірити чи програма працює відповідно до очікувань користувача, плавно та ефективно за будь-яких умов.  Для розуміння чи вона заявляється як надійний додаток. | Коли клієнт подає запит на зміну, що призводить до змін у кодовій базі.  Коли змінюється середовище тестування.  Коли критичні помилки, знайдені тестувальником на етапі тестування, виправляються розробником.  Коли змінюється інтерфейс додатку для покращення користувацького досвіду. |
| Обмеження | Менш надійне, ніж інші типи тестів, оскільки воно покладається на фактичну функцію програмного забезпечення, що тестується. Це означає, що може бути важко знайти помилки, які виникають під час звичайного використання.  Саме по собі воно не гарантує, що додаток готовий до використання в реальних умовах.  Можливість пропустити логічні помилки в додатках і висока ймовірність проведення надлишкового тестування. | Нефункціональні вимоги можуть бути дуже нечіткими, що ускладнює виконання нефункціональних тестів. | Визначення тестових кейсів у кожному модулі, до якого вносяться зміни, займає багато часу, дуже ймовірно, що ми пропустимо тестовий кейс, який є критично важливим для перевірки цієї зміни.Цілком можливо, що під час вибору та відбору тестових кейсів для виконання, ми можемо пропустити перевірку критичної функціональності. |
| Особливості | Функціональне тестування проводиться на основі бізнес-вимог.  Частіше здійснюється вручну.  Приклад: Метод тестування "чорного ящика". | Нефункціональне тестування проводиться на основі очікувань замовника та вимог до продуктивності.  Доцільніше тестувати за допомогою автоматизованих інструментів.  Приклад: Loadrunner. | Мета регресійного тестування, полягає в тому, щоб переконатися, що будь-які зміни в додатку або системах не призвели до розривів коду і що система продовжує функціонувати належним чином.  Регресійне тестування може бути ручним або автоматизованим. Якщо кількість тестових кейсів невелика, його можна провести вручну за менший час. Однак, якщо кількість регресійних тестів значна, автоматизоване тестування є ідеальним рішенням. |

**2. Поясни, в чому різниця між регресією та ретестингом (5 речень).**  
  
І регресійне тестування, і повторне тестування гарантують, що програмне забезпечення працює так, як очікується. Не дивлячись на це вони мають суттєві відмінності в контексті STLC.

Повторне тестування проводиться для перевірки того, що початкова помилка, яка була знайдена і виправлена, працює належним чином, в той час як регресійне тестування використовується для перевірки програми на наявність дефектів, які могли виникнути в результаті змін або інших невідомих помилок.

Retesting - це тестування програмного забезпечення, яке не працювало, але яке, на нашу думку, було виправлено. Ми тестуємо його, щоб підтвердити, що його дійсно виправлено.

Regression testing - це тестування програмного забезпечення, яке працювало, але тепер, через оновлення, може не працювати.



**Другий рівень — детальніше заглибся в практику.**

**1. Виконай завдання попереднього рівня.**

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

* **Якщо так – в яких випадках?**
* **Якщо ні – чому?**
* **Обґрунтуй свою відповідь.**

На мою думку, неможливо створити якісний продукт виконуючи тільки функціональне або нефункціональне тестування. Обидва мають працювати як доповнення один одного.   
Якщо звертати увагу тільки на функціональне тестування, як основний інструмент, то ми неминуче будемо мати справу з проблемою: коли продукт нарешті доставлений клієнту і кінцевий користувач починає користуватися програмним додатком, є ймовірність, що деякі недопрацювання, пов'язані з продуктивністю програми, виявляться на поверхні, ці проблеми не підпадають під покриття функціонального тестування, але вони можуть легко призвести до негативного впливу на користувацький досвід. Таким чином, нефункціональне тестування відіграє важливу роль, щоб зробити додаток швидким і надійним для кінцевих користувачів.   
Якщо не провести нефункціональне тестування перед запуском проекту, це може призвести до деяких критичних проблем, наприклад, коли багато користувачів одночасно входять в додаток, система стає повільною або час відгуку може збільшитися при збільшенні навантаження на додаток. Все це призводить до негативного досвіду користування продуктом.

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

Метою Smoke Testing є: виявити будь-які ранні дефекти в програмному продукті. Продемонструвати стабільність системи. Продемонструвати відповідність вимогам.  
Я вважаю, що воно є доречним завжди.   
Окрім виявлення помилок, димове тестування важливе, оскільки воно прокладає шлях для інших типів тестування. Без димового тестування ми не зможемо перейти до функціонального тестування, а це означає, що збірка не зможе прогресувати.

**Третій рівень — різнобічно опануй тематику уроку.**

**1. Виконай завдання двох попередніх рівнів.**

**2. 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.

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

Тест-кейси можна знайти [ТУТ](https://docs.google.com/spreadsheets/d/1Zv6Bhj3ybcGebKrqaYjjIcmWLbdkSpRy/edit?usp=sharing&ouid=109352105310201357541&rtpof=true&sd=true).

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

Нефункціональні вимоги до застосунку можна знайти [ТУТ.](https://docs.google.com/document/d/14DpuaJTaq6OkFwJ0PFz1neVpqSL143HY3L0kA10WFcI/edit?usp=sharing)

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

Для перевірки вищезгаданих вимог потрібно провести різні види тестування:

**Тестування продуктивності** - це оцінка того, як система працює з точки зору швидкості відгуку і стабільності під певним робочим навантаженням.

**Тестування надійності** - це методика тестування, яка пов'язана з перевіркою здатності програмного забезпечення функціонувати в заданих умовах навколишнього середовища, що допомагає виявити проблеми в дизайні та функціональності програмного забезпечення.

**Тестування безпеки** - це процес тестування програмного забезпечення, який гарантує, що програмне забезпечення вільне від будь-яких потенційних вразливостей або слабких місць, ризиків або загроз, щоб програмне забезпечення не могло завдати шкоди системі та даним користувача.

**Юзабіліті-тестування** - це метод перевірки функціональності веб-сайту, додатку або іншого цифрового продукту шляхом спостереження за реальними користувачами, які намагаються виконати завдання на ньому.

**Тестування інтероперабельності** - це тип тестування програмного забезпечення, який виконується для вивчення взаємодії програмного забезпечення з його компонентами або іншим програмним забезпеченням.