**Перший рівень**

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

| Назва тестування | Що перевіряється | Коли перевіряється | Обмеження | Оособливості |
| --- | --- | --- | --- | --- |
| Функціональне тестування | 1) Functional testing - розглядає певну поведінку системи в цілому чи її компонентів, на основі їхньої специфікації  2) Security and Access Control testing - перевіряє безпеку системи, наявні ризики пов’язані із захистом додатку, конфіденційної інформації, тощо.  3) Interoperability testing - перевіряє сумісність і взаємодію додатків із компонентами системи. | Перевіряють конкретну дію або функцію коду, здатність використовувати її функціонал належним чином та її працездатність. Коротко кажучи, перевіряє можливість системи виконувати задані функції. | Перевіряє тільки певний функціонал системи чи її компонентів без розуміння витримки самої системи. Загалом використовується при перевірках вимог та бізнес-процесах. | Використовуються на усіх чотирьох рівнях тестування, імітує фактичне використання системи. |
| Нефункціональне тестування | 1) Performance testing - складається з певної кількості різних варіантів тестування, із яких можна позначити - тестування в умовах стресу, тобто перегрузки системи, тестування працездатності системи при тривалому середньому навантаженні, тестування при великому навантаженні на систему.  2) Usability testing - перевіряє зручність та інтуїтивність використання системи, а також її привабливість до користувачів.  3) Failover recovery testing - перевіряє можливість системи відновитися та виконувати вимоги після певних збоїв. (коли ми крашимо систему та перевіряємо її можливість відновлення та нормалізації функціонування) | Перевіряється для розуміння працездатності усієї системи, її можливості відновлюватися, адаптуватися, витримувати певне навантаження, а також зручність у використанні. Загалом перевіряє, якщо можна так сказати, міцність працювання системи. | Перевіряє характеристики програмне забезпечення, які частіше не пов’язані з певними функціями та діями. | В цілому це тестування того, як саме працює система при нагрузках, імітація певної кількості користувачів, масштабованості програм від навантаження, отримання оцінки продуктивності з роботи системи за певних обставин. |
| Пов’язане зі змінами тестування | 1) Smoke-testing та Sanity testing - дуже схожі перевірки, які часто використовують на етапі розробки. Вони перевіряють, відповідним чином, поверхневу та внутрішню середу системи (або її частину). За ISQB - вони аналогічні.  2) Regression testing та Re-testing - аналогічно перевіряють систему після певних змін у ній, відповідно, перше перевіряє працездатність системи після будь-яких змін, коли друге - працездатність системи після виправлення дефектів.  3) Build verification test - перевіряє відповідність критеріям якості щодо початку подальшого тестування. | Перевіряє можливість функціонування системи після внесення змін, при первинному тестуванні раннішньої системи та розуміння можливості подальшого тестування. Я гадаю, що це тестування відповідає на питання: “Як система/компонент працює після Х дії/дій?” | Ключовий аспект у перевірці програмного забезпечення, бо відповідає за взаємодію та інтерфейс компонентів чи самої ОС. | Тестування дійсно показує чи виправлена проблема та чи працює система після установки ПЗ чи певних змінах. |

1. Різниця між регресією та ре-стестингом

На мою думку основні відміни між регресією та ре-тетсінгом є наявність саме виправлених багів. Регресія проводиться після будь-яких змін у системі, щоб перевірити, що її функціональність працює як і раніше, саме коли ре-тестінг зазвичай використовується при виправленні дефектів та перевірці виправленого функціоналу.   
 Як приклад до регресії, можна привести такий: заміна у автомобілі деяких деталей на більш сучасні (якщо автомобіль працює належним чином, та функції нових деталей успішно виконуються, то регресивне тестування пройшло належним чином).  
 Приклад ре-тестингу: у автомобілі проходить заміна тормозної системи через її несправність. Якщо функціональність тормозної системи працює належним чином, то можна зробити висновок, що ре-тестінг виконано вдало.

**Другий рівень**

1) Як ти вважаєш, чи можливе для продукту проведення тільки функціонального тестування, без перевірки нефункціональних вимог?

На мою думку чіткої відповіді на це питання немає, все залежить від проекту. Але якщо, мі беремо приклад - автомобіль (що є вже більш складним у системі та реалізації вимог, ніж, наприклад, пристрій портативної зарядки), то тоді що функціональне, що нефункціональне тестування дуже важливо.

Якщо також далі розглядати на прикладі автомобілю, то наші функціональні тести не можуть бути зроблені без нефункціональних, і навпаки. Добре, навіть якщо і якісь з цих тестів не зробити, що реально, то це не гарантує виконання певних належних перевірок, що спричиняє подальші ризики та втрату ресурсів, а наприкінці і не проходження продуктом верифікації чи валідації.

Щоб показати наглядно, то ми розглядаємо на прикладі тієї ж омиваючої системи у автомобілі. Якщо функціональне тестування на вимоги та працездатність функцій не було виконано, то система не буде працювати належним чином, навіть якщо перевірка навантаження та безпеки цієї системи були проведені коректно, так працює і навпаки, система може давати збій при деяких випадках, незважаючи на повне ідеальне проведення тестування функціоналу продукту. Тому я гадаю, що важливість проведення обох тестувань є дуже високою, щоб отримати гарно працюючий та задовольняючий замовника продукт.

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

Я вважаю, що проведення цього тестування є дуже важливим на первинних етапах розробки системи, щоб перевірити її працездатність належним чином. Звісно, димове тестування не завжди можуть використовувати, але на мою думку, це тестування є дуже важливим, щоб на ранніх етапах розробки системи не допустити помилок, та передавати на подальші, більш серйозні тести, вже перевірений продукт, яких не крашнеться при першій можливості. Я гадаю, що проведення цього тестування можна зрівняти з кнопкою “START” у циклі усього тестування.

**Третій рівень**

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.

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

Посилання на тест-кейси залишаю:

[тест кейси додатку](https://docs.google.com/spreadsheets/d/1Y3bmUH-Ik10vODh9-3hVUst2curxZgCiXQ0HrYx-fy4/edit#gid=0)

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

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

а) Мінімальна роздільна здатність - 720 пікселів. Максимальний розмір відео коротше 10 хвилин - 650 МБ. Максимальний розмір відео коротше 60 хвилин - 3,6 ГБ.   
Перевірка: тестування сумісності  
Перевіряє чи програма сумісна з апаратним забезпеченням, базою даних та іншим програмним забезпеченням.

б) Підтримка 100 різних мов світу.

Перевірка: інтернаціоналізації та локалізації  
Перевіряє адаптування до різних країн і регіонів за будь-яких змін.

в) Одночасний перегляд публікації 100 тис. користувачів

Перевірка: продуктивності  
Перевіряє як швидко працює цей аспект при певному навантаженні

г) Користувачі можуть користуватися додатком не виходячи з нього протягом невизначенного часу ( за виняток серверних робіт, оновлень додатка, тощо).  
Перевірка: На довговічність  
Перевіряє систему під навантаженням протягом значного періоду часу, щоб дізнатися поведінку системи в таких умовах.

д) Додаток може використовуватися на системах Iphone, Android та у браузерах Google Chrome, Microsoft edge, Firefox, Apple’s Safari

Перевірка: Відповідності

Перевіряє як система дотримується стандартів, та наявність їх відхилень.