1.1. Склади порівняльну таблицю функціонального, нефункціонального і пов'язаного зі змінами видів тестування.
Порівняння має містити такі блоки:

	функціональне	**нефункціональне**	**пов'язане зі змінами**
що перевіряється	чи виконує програмний продукт свої функції відповідно до визначених вимог та специфікацій „Що" робить продукт. "can the user do this" or "does this particular feature work"	оцінює якості програмного продукту, такі як продуктивність, надійність, безпека, ефективність, зручність для користувача та інші характеристики, які не відносяться безпосередньо до функціональності програми. "Як" система працює", поведінка програми.	чи не вплинули останні зміни в програмному продукті на його попередню функціональність
коли застосовується	використовується на різних етапах розробки програмного продукту та після його випуску	використовується на різних етапах розробки програмного продукту та після його випуску	після внесення змін у продукт
обмеження	• Орієнтованість на функціональність: перевіряє, на відповідність функціональним вимогам та специфікаціям, але не оцінює інші характеристики (продуктивність, безпека, доступність, користувацький досвід ін. нефункціональні аспекти). • Залежність від документації: Функціональне тестування ґрунтується на наявності докладної документації, яка описує функції програми. Якщо така документація відсутня або не актуальна, то тестування може бути ускладненим. • можливість упущення логічних помилок у програмному забезпеченні; • ймовірність надмірного тестування.	• Висока вартість: особливо якщо потрібно створювати спеціалізоване середовище для тестування продуктивності або проводити складні тести на безпеку. • Суб'єктивність: зокрема щодо зручність використання та доступність для користувачів з обмеженими можливостями. • Складність вимірювання:, продуктивність та ефективність, можуть бути складні для об'єктивного вимірювання та порівняння. • Залежність від архітектури та інфраструктури: зокрема. тести на продуктивність та масштабованість • Важливість спеціалізованого знання: Для ефективного тестування безпеки або інших нефункціональних аспектів може бути важливим наявність фахівців із спеціалізованим знанням.	• Покриття тестами: Не завжди можливо передбачити всі можливі сценарії та взаємодії, які виникають після змін. Залежності інших систем: Якщо зміни впливають на інші системи або компоненти, тестування може бути ускладненим через необхідність координації з цими системами і оточеннями. • Час: Велика кількість змін або складність програмного продукту може вимагати багато часу для повного тестування. • Високий ризик невдалого тестування: Помилки, виявлені після внесення змін, можуть мати серйозний вплив на програму або систему. • Масштабність тестування: Зміни можуть вплинути на велику кількість функціональних блоків, і тестування всіх може бути складним завданням..
особливості	• допомагає переконатися, що програма відповідає своїм функціональним вимогам і працює правильно для користувачів	• допомагає забезпечити, що програма не лише відповідає функціональним вимогам, але і відповідає вимогам щодо продуктивності, безпеки та інших нефункціональних аспектів.	вимагає уважності і виваженості, а також спеціального планування та координації, щоб забезпечити, що нові зміни не порушують стабільність та функціональність програмного продукту

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

Regression testing	**Retesting**
1. Регресійне тестування виконується тільки при додаванні нової фічі (додаткова функціональність ПЗ) або істотній зміні функціоналу системи.	1.Ретест виконується в тому ж оточенні й з тими ж даними, але на новому білді.
2.Регрес можна проводити паралельно з повторним тестуванням.	2.Повторне тестування має вищий пріоритет та має бути виконано до регресійного.

3.Тест-кейси можуть бути автоматизовані.	3.Тест-кейси не можуть бути автоматизовані.
4.В рамках регресійного тестування тест-кейси, які були відмічені раніше як «Passed», повинні бути перевірені повторно.	4.В рамках повторного тестування (ретест) перевіряються тест-кейси тільки зі статусом «Failed».

2.1. 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.**

Вибір видів тестування залежить від конкретного проекту. Проте, для випуску якісного продукту варто поєднувати різні види тестування. І функціональне, і нефункціональне. Тому, загалом, відповідь “НІ”.

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

Димове тестування проводиться на ранніх етапах розробки, поверхнево перевіряє основну функціональність на відсутність критичних чи блокуючих дефектів. Дає відповідь чи готовий продукт до повномасштабного тестування чи потребує доопрацювання.

Доцільність проведення димового (smoke) тестування залежить від конкретної ситуації та умов проекту:

- Розмір та складність проекту: Димове тестування особливо корисне на великих та складних проектах, де базовий функціонал може бути складним і містити багато взаємодіючих компонентів. На менших проектах це може бути менш важливим.
- Складність інтеграції: Якщо проект включає в себе багато компонентів, які повинні взаємодіяти між собою, димове тестування може допомогти виявити проблеми на ранніх етапах, коли їх виправити легше.
- Частота змін і релізів: Якщо проект має часті зміни або релізи, то димове тестування може бути корисним для впевненості в тому, що нові зміни не вплинули на базовий функціонал.
- Вартість і час: Димове тестування може вимагати витрати часу та ресурсів, тому важливо оцінити, чи вони виправдовуються під час конкретного проекту.
- Наявність інших видів тестування: Якщо в проекті вже є обширні автоматизовані тести, які покривають базовий функціонал, то димове тестування може бути менш критичним.
- Стратегія тестування: Димове тестування може бути однією зі складових стратегії тестування, яку обирає команда розробників, разом з іншими видами тестів, такими як модульне тестування, інтеграційне тестування, функціональне тестування тощо.

3.1. 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 кейси розміщені у TestRail / Natalia Ruda / HW Види тестування

3.2. Напиши, які нефункціональні вимоги ти хотів/ла б застосувати для продукту твого стартапу. Опиши перевірки, які б їх перевіряли (3-5 прикладів).

- Load testing, - перевірити продуктивність застосунку при плановій кількості одночасної присутності користувачів (10 тис ос), одночасній реєстрації 100 нових користувачів; одночасному завантаженні 100 фотографій користувачами
- Stress testing - перевірити продуктивність застосунку при одночасній присутності користувачів до 20 тис ос; одночасній реєстрації 1000 нових користувачів; одночасне завантаження 1000 фотографій користувачами
- Volume testing - перевірити як застосунок обробляє 1000 фотографій у фотоальбомі користувача (чи є зависання у відображенні фото, швидкодія пошуку). При нормі 500 фото.
- Usability Testing - перевірка зручності навігації, пошуку, фільтрів, завантаження фото, видалення фото.