1. **Beet Seed**

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

* що перевіряється;
* коли застосовується;
* обмеження;
* особливості

**Виконання завдання:**

| **Назва тестування** | **Що перевіряється** | **Коли застосовується** | **Обмеження** | **Особливості** |
| --- | --- | --- | --- | --- |
| [Функціональне](https://tryqa.com/what-is-functional-testing-testing-of-functions-in-software/) | Перевіряються функції компонента або системи на відповідність специфікаціям та вимогам. | Застосовується під час перевірки окремих функцій або дій у коді, як правило, на основі специфікацій вимог. | Не охоплює нефункціональні аспекти, такі як продуктивність, безпека, і сумісність. | Використовуються методики, засновані на специфікаціях, такі як тестування на основі вимог та тестування на основі бізнес-процесів. |
| [Нефункціональне](https://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/) | Перевіряються характеристики якості компонента або системи, такі як надійність, зручність використання, продуктивність, сумісність, безпека і т.д. | Застосовується на всіх рівнях тестування для оцінки нефункціональних аспектів програмного забезпечення. | Не охоплює функціональні аспекти, такі як перевірка конкретних функцій або дій користувачів. | Включає такі види тестування, як тестування надійності, продуктивності, сумісності, безпеки, масштабованості, навантаження та стрес-тестування. |
| [Пов’язане зі змінами](https://tryqa.com/what-is-regression-testing-in-software/) | Перевіряються нові зміни або модифікації на предмет впливу на існуючу функціональність. | Використовується після додавання нових функцій, виправлення помилок, поліпшення продуктивності. | Може бути трудомістким і часозатратним без автоматизованих інструментів. | Виконується за допомогою технік, таких як корективне тестування, прогресивне тестування, повторне тестування та селективне тестування. |

**Завдання:** Поясни, в чому [різниця між регресією та ретестингом](https://tryqa.com/difference-between-regression-testing-and-retesting/) (5 речень).

**Виконання завдання:**

Регресійне тестування перевіряє, чи не вплинули нові зміни або виправлення на вже існуючу функціональність програми. Ретестинг фокусується на перевірці виправлених помилок для підтвердження, що вони дійсно були усунені. Регресійне тестування може виконуватися автоматизовано, тоді як ретестинг вимагає повторного виконання тестів вручну. Під час регресійного тестування перевіряються всі функції, включаючи ті, що пройшли тести раніше, тоді як ретестинг стосується лише виправлених помилок.

1. **Beet Sprout**

**Завдання:** 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.

**Виконання завдання:**

На мою думку, проведення тільки функціонального тестування для продукту без перевірки нефункціональних вимог можливе, але лише в певних випадках. А саме, це доречно для невеликих, простих систем, де продуктивність, безпека або зручність використання не є критичними. Наприклад, для внутрішніх інструментів, додатків або прототипів, де головне завдання — перевірка базової функціональності. Однак для більшості комерційних програм нефункціональні вимоги відіграють ключову роль у забезпеченні якості користувацького досвіду, надійності та безпеки, тому нехтування ними може призвести до серйозних проблем.

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

**Виконання завдання:**

На мою думку, [smoke тестування](https://tryqa.com/what-is-smoke-testing-when-to-use-it-advantages-and-disadvantages-2/) є необхідним і завжди доречним. Воно забезпечує перевірку основних функціональностей системи, що дозволяє виявити серйозні проблеми на ранніх етапах. Це тестування дозволяє розробникам і тестувальникам переконатися, що основні компоненти програми працюють правильно перед тим, як переходити до детального тестування. Димове тестування виконується швидко і допомагає заощадити час і ресурси, запобігаючи глибокому тестуванню недопрацьованих або несправних білдів.

1. **Mighty Beet**

**Завдання:** 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.

**Виконання завдання:**

1. Тест кейс“Завантаження фотографії котика”

Попередні умови:користувач має зареєстрований обліковий запис і ввійшов у систему.

Кроки:

1. Відкрити додаток і перейти до розділу завантаження фотографій.

Очікуваний результат: відкривається інтерфейс для завантаження фотографії.

1. Вибрати фотографію котика зі свого пристрою.

Очікуваний результат: фотографія відображається у попередньому перегляді.

1. Натиснути кнопку "Завантажити".

Очікуваний результат: фотографія успішно завантажується, і з'являється підтвердження.

Приблизний час на тестування: 5 хвилин.

2. Тест кейс “Заборона завантаження фотографії іншої тварини”

Попередні умови:користувач має зареєстрований обліковий запис і ввійшов у систему.

Кроки:

1. Відкрити додаток і перейти до розділу завантаження фотографій.

Очікуваний результат: відкривається інтерфейс для завантаження фотографії.

1. Вибрати фотографію собаки зі свого пристрою.

Очікуваний результат: відбувається процес валідації, з'являється попередження про заборону завантаження фотографій інших тварин.

1. Спробувати натиснути кнопку "Завантажити".

Очікуваний результат: завантаження блокується, і користувач отримує повідомлення про помилку.

Приблизний час на тестування: 5 хвилин.

3. Тест кейс “Додавання друзів”

Попередні умови: користувач має зареєстрований обліковий запис і ввійшов у систему.

Кроки:

1. Відкрити додаток і перейти до розділу пошуку користувачів.

Очікуваний результат: відкривається інтерфейс пошуку.

1. Ввести ім'я користувача, якого потрібно додати до друзів.

Очікуваний результат: система знаходить користувача і відображає його профіль.

1. Натиснути кнопку "Додати до друзів".

Очікуваний результат: запит на додавання до друзів надсилається успішно, і з'являється підтвердження.

Приблизний час на тестування: 5 хвилин.

4. Тест кейс “Ставлення "вподобайок" на фотографії”

Попередні умови:користувач має зареєстрований обліковий запис і ввійшов у систему.

Кроки:

1. Відкрити додаток і перейти до стрічки фотографій.

Очікуваний результат: відображається стрічка фотографій.

1. Обрати будь-яку фотографію котика.

Очікуваний результат: фотографія відкривається у повноекранному режимі.

1. Натиснути кнопку "Вподобати".

Очікуваний результат: "вподобайка" додається до фотографії, і кількість "вподобайок" збільшується на 1.

Приблизний час на тестування:3 хвилини.

5. Тест кейс “Залишення коментарів під фотографією”

Попередні умови:користувач має зареєстрований обліковий запис і ввійшов у систему.

Кроки:

1. Відкрити додаток і перейти до стрічки фотографій.

Очікуваний результат: відображається стрічка фотографій.

1. Обрати будь-яку фотографію котика.

Очікуваний результат: фотографія відкривається у повноекранному режимі.

1. Прокрутити вниз до розділу коментарів і ввести текст коментаря.

Очікуваний результат: коментар відображається у полі для введення.

1. Натиснути кнопку "Опублікувати".

Очікуваний результат: коментар успішно публікується і відображається під фотографією.

Приблизний час на тестування: 4 хвилини.

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

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

**Виконання завдання:**

1. Сумісність (Compatibility): програма повинна коректно працювати на різних версіях iOS та Android.

Перевірка: потрібно перевірити роботу програми на різних пристроях і версіях операційних систем iOS та Android, щоб переконатися в її стабільній роботі на всіх підтримуваних платформах.

1. Зручність використання (Usability): інтерфейс користувача повинен бути інтуїтивно зрозумілим і легким у використанні.

Перевірка: потрібно провести тестування з участю користувачів, надавши їм завдання, наприклад, завантажити фото, додати друзів, залишити коментар, і зібрати зворотний зв'язок щодо зручності виконання цих завдань.

1. Продуктивність (Performance): завантаження фотографій не повинно перевищувати 5 секунд.

Перевірка: потрібно завантажити різні фотографії (різного розміру та роздільної здатності) і виміряти час, який потрібен для їхнього завантаження.

1. Надійність (Reliability): система повинна працювати без збоїв протягом тривалого часу.

Перевірка: потрібно виконати тривале навантажувальне тестування, відтворюючи безперервну роботу програми, протягом 24 годин, перевіряючи на наявність збоїв або падінь.