№1.1

| Характеристика | Функціональне тестування | Нефункціональні | Пов'язані зі змінами тестування |
| --- | --- | --- | --- |
| Що перевіряє | Коректність виконання функцій | Продуктивність, безпека, сумісність, юзаібіліті. | Вплив змін на функціональність |
| Коли застосовується | На всіх етапах розробки та перед випуском | Під час перевірки продуктивності та якості, стабільності | Після змін в программі або виправлення багів |
| Обмеження | Може бути не ефективним при тестуванні продуктивності безпеки тощо | Не виявляє проблеми з конкретними частинами функцій | Ретестинг перевіряє виправлені баги.  Ретестинг забирає багато часу |
| Особливості | Сценарії використання, відповідність вимогам | Оціює стабільність продуктивність, юзабіліті | Ретестинг перевіряє виправлені баги.  Регресивне перевіряє що зміни не вплинули на інші системи |

№1.2

Ретестинг можна назвати частиною регресійного тестування, оскільки в принципі вони схожі. Однак, регресійне тестування охоплює ширший спектр програмного забезпечення та перевіряє, як виправлення в різних модулях можуть вплинути на програму в цілому. Ретестинг, навпаки, зосереджується на перевірці конкретних виправлень і перевіряє, чи був усунутий виявлений баг чи ні. Регресійне тестування часто проводиться після кожного випуску нової версії програмного забезпечення, тоді як ретестинг може бути здійснений у процесі розробки для перевірки окремих виправлень. Крім того, ретестинг допомагає визначити, чи була виправлена помилка без впливу на інші частини системи.

№2.2

На мою думку, проводити тільки функціональне тестування не має сенсу, оскільки сайт може функціонувати ідеально з точки зору функціональності, але на продакшені може зламатися від навантаження всього двох користувачів. Це відбувається тому, що не було проведено нефункціональне тестування, яке виявляє модулі, що працюють неефективно або зовсім не функціонують під навантаженням. Таким чином, без нефункціонального тестування можна пропустити важливі аспекти, які впливають на стабільність і продуктивність системи. Тільки комплексний підхід до тестування забезпечує надійність та високу якість продукту.

№2.3

Smoke тестування є надзвичайно доречним під час отримання нових збірок програмного забезпечення для виявлення слабких місць на початку тестування. Воно дозволяє швидко визначити основні проблеми та недоліки, які можуть вплинути на стабільність системи. Крім того, smoke тестування допомагає виправити критичні помилки на ранніх етапах і уникнути значних витрат на детальне тестування збірки, яка має серйозні дефекти. Таким чином, це тестування забезпечує ефективність процесу розробки і сприяє випуску більш стабільного та надійного програмного забезпечення.

№3.2

| **Test Scenario** | Завантажити фотографії котиків |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | **Pass / Fail / Not executed / Suspended** | |
|
| 1 | Увійти до системи | | Користувач увійшов | |  | |
| 2 | Перейти до форми завантаження фото | | Форма відкрилась | |  | |
| 3 | Обрати файл з фотографєю котика та натиснути завантажити | | Фото успішно завантажилось | |  | |
|  |  |  |  |  |  |  |
| **Test Scenario** | Завантаження обєкта або тварини яка не є котиком |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | **Pass / Fail / Not executed / Suspended** | |
|
| 1 | Увійти до системи | | Користувач увійшов | |  | |
| 2 | Перейти до форми завантаження фото | | Форма відкрилась | |  | |
| 3 | Обрати файл з фотографєю не котика та натиснути завантажити | | Фото успішно не завантажилось | |  | |
|  |  |  |  |  |  |  |
| **Test Scenario** | Додавання обєкта під назвою друг в список друзів |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | **Pass / Fail / Not executed / Suspended** | |
|
| 1 | Увійти до системи | | Користувач увійшов | |  | |
| 2 | Знайти користувача | | Користувач знайдено | |  | |
| 3 | Натиснути на кнопку додати в друзі | | Обєкт друг додано до друзів | |  | |
|  |  |  |  |  |  |  |
| **Test Scenario** | Встановлення вподобайки |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | **Pass / Fail / Not executed / Suspended** | |
|
| 1 | Увійти до системи | | Користувач увійшов | |  | |
| 2 | Обрати фото кота | | Фото обрано | |  | |
| 3 | Встановити подобайку | | Подобайка встановлена | |  | |
|  |  |  |  |  |  |  |
| **Test Scenario** | Завантаження обєкта або тварини яка не є котиком |  |  |  |  |  |
|  |  |  |  |  |  |  |
| **Step #** | **Step Details** | | **Expected Results** | | **Pass / Fail / Not executed / Suspended** | |
|
| 1 | Увійти до системи | | Користувач увійшов | |  | |
| 2 | Знайти фото кота | | Фото успішно знайдено | |  | |
| 3 | Знайти поле для коментарів і ввести ваш дужее важливий коментар | | Коментар успішно введений | |  | |
| 4 | Натиснути кнопку додати коментар | | Коментар успішно доданий | |  | |

№3.4

Перелік не функціональних вимог:

Продуктивність

* Вимога: Застосунок повинен завантажуватися не більше ніж за 3 секунди.
* Перевірка: Використання інструментів для вимірювання часу завантаження застосунку на різних пристроях.

Масштабованість

* Вимога: Застосунок повинен підтримувати одночасне використання великою кількістю користувачів без зниження продуктивності.
* Перевірка: Симуляція одночасного доступу багатьох користувачів і перевірка, як це впливає на продуктивність застосунку.

Сумісність

* Вимога: Застосунок повинен коректно працювати на різних версіях iOS та Android, а також на різних моделях пристроїв.
* Перевірка: Тестування на різних платформах і пристроях для виявлення можливих проблем.

Юзабіліті

* Вимога: Інтерфейс застосунку повинен бути зручним у використанні.
* Перевірка: Проведення тестування з реальними користувачами для отримання їх відгуків.

Безпека

* Вимога: Застосунок повинен забезпечувати захист даних користувачів.
* Перевірка: Проведення тестів на безпеку для виявлення можливих вразливостей та забезпечення захищеності даних.