**Тема 6**

Перший рівень

|  | **Функціональне тестування** | **Нефункціональне тестування** |
| --- | --- | --- |
| що перевіряється | задана поведінка | продуктивність (навантажування, працювання системи в умовах стресу, стабільність та надійність) |
|  | інтерфейс | встановлення |
|  | безпека | відновлення |
|  | взаємодія додатків | система на різних платформах |
| коли застосовується | в процесі комплексного тестування для перевірки відповідності реальних функцій продукту його функціональним вимогам. | в процесі тестування для перевірки нефункціональних вимог системи |
| обмеження | тестування тільки функціональних функцій | тестування тільки нефункціональних функцій |
| особливості | функціональні тести перевіряють, чи відповідає ПЗ визначеним вимогам | нефункціональне тестування з'ясовує наскільки добре працює система загалом |

2. **Ретестинг** - це коли ми знаходимо помилки, формуємо баг репорти, які пізніше передаються розробникам на виправлення. Після цього відбувається ретестинг, тобто перевірка виправлених помилок. Повторне тестування виконується з тими самими даними в тому самому середовищі просто з новою збіркою.

**Регресія** - це тестування вже виправлених помилок для того, щоб переконатися, що система нормально працює з новими функціями, виправленими помилками або будь-якими змінами в функціональності. Це виконається для того, щоб підтвердити, що зміни не вплинули на решту функціонала програми.

Другий рівень

2. Я вважаю, що обидва типи тестування важливі, адже вони дозволяють краще зрозуміти особливості системи. Функціональне тестування можливе без перевірки нефункціональних вимог, адже основне завдання функціонального тестування це перевірити, що весь функціонал системи відповідає зазначеним вимогам. Проте, не менш важливим є нефункціональне тестування, тому, якщо ми хочемо забезпечити хорошу якість програми, потрібно проводити обидва тестування.

3. Я вважаю, що димове тестування є важливим, адже ми маємо впевнитися, що основний функціонал працює і вже після цього продовжувати подальше тестування. Найбільш доречно використовувати даний метод, коли вважаємо програму не зовсім стабільною, адже це допоможе протестувати надзвичайно важливі функції на початковому етапі для того, щоб виявити помилки якомога раніше.