## **Урок 6. Домашнє завдання**

### Рівень І

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

| **Вид тестування** | **Що перевіряється?** | **Коли застосовується?** | **Обмеження** | **Особливості** |
| --- | --- | --- | --- | --- |
| *Функціональне*  *тестування* | Перевіряє відповідність системи функціональним вимогам | Може бути використане на всіх рівнях тестування | Є вірогідність під час перевірки функціональності пропустити можливі логічні помилки в ПЗ | Даний вид тестування є простим і зрозумілим навіть для новачка |
| *Тестування безпеки* | Перевірка надійності системи від можливих ризиків та загроз | Коли необхідно перевірити вразливість ПЗ. Повинно бути запроваджено на етапі Planning (SDLC) | Це тестування при необмеженому асі і ресурсах взломає будь-яку систему. Ціна злому має бути вищою за ціну інформації. | Складається з шести аспекті (авторизація, аутентифікація, цілісність,, конфіденційність, безвідмовність, доступність) |
| *Тестування взаємодії* | Дає оцінку якості взаємодії системи або її компонентів з іншими компонентами або ПЗ | Під час інтеграційного рівня тестування | Не перевіряє взаємодію компонентів всередині самої системи | Охоплює тестування на сумісність та інтеграційне тестування |
| *Тестування навантаження* | Перевіряє продуктивність та швидкість відгуку системи в межах зростаючого навантаження | Відбувається на пізніх етапах процесу розробки, коли система майже готова до випуску | Визначає можливі вузькі місця в системі та передбачає контрольну поведінку системи, а не виявляє баги. | Має моделювати поведінку реальних користувачів , спираючись на психологію користувачів. |
| *Юзабіліті тестування* | Перевіряє зручність використання системи, та її ергономічність | Може бути проведене на всіх рівнях тестування | Потребує переважно ручного підходу, що робить тестування більш виснажливим | Передбачає багато користувацьке тестування, щоб більш ретельно визначити реакцію цільового ринку на продукт |
| *Тестування на відмову та відновлення* | Перевіряє спроможність системи до відновлення після помилок або збоїв | Має життєво важливе значення для перевірки готовності виробничої системи | Бюджет проекту - це перше, що має бути взято до уваги перед проведенням цього тестування, потребує оцінки доцільності проведення такого тестування для окремих продуктів | Надає системі можливість працювати безперебійно та забезпечує відсутність втрат під час простою. |
| *Димове тестування* | Стабільність та працеспроможність системи у цілому | На початковому етапі, щоб перевірити базову готовність системи до більш розширеного тестування | Поверхнева перевірка системи на працездатність, без заглиблення в деталі. | Може виконуватися як розробникми, так і тестувальниками |
| *Регресійне тестування* | Виявляє чи мали вплив на систему додані зміни в коді | При додаванні змін вже в існуючому, проекті | Перепровіє вже раніше пройдені тест-кейси | Може виконуватися паралельно з ретестами |
| *Ре-тест* | Перевіряє чи був виправлений раніше виявлений дефект | Після виправлення дефекту | Проводять тільки для зафейлених тест-кейсів. Не піддається автоматизації. | Проводять перед sanity testing, також пріоритет вище ніж у регресійного тестуванн, тому його проводять раніше. |

**2. Поясни, в чому різниця між регресією та ретестингом**

Ретестінг - це перевірка системи на наявність старих дефектів, після їх виявлення та виправлення. Регресійне тестування - це тестування вже раніше протестованої системи, що надає можливість впевнитися, що внесені зміни не викликали появу дефектів в частині програми, яка не змінювалась.

Якщо пов’язати ці два поняття, то можна сказати своїми словами, що ретест - це перевірка багу, після того як його пофіксили, а регресійне тестування - це перевірка чи не вплинуло виправлення багу (наприклад, для якого і проводиться ретест) на працездатність системи загалом. Ретест охоплює компонент/блоки системи, в яких раніше вже був виявлені дефекти, та перевіряє чи були вони виправлені. А регресія, в свою чергу, перевіряє чи працюють після виправлення багів інші компоненти/блоки системи, в яких не було виявлено дефектів під час попереднього тестування.

### Рівень ІІ

**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.

Я вважаю, що проведення тільки функціонального тестування, без перевірки нефункціональних вимог - неможливе. Неможливе, тому що під час розробки системи (у фазі Дизайну) з її подальшою Імплементацією (написання коду) враховуються всі вимоги, що були висунуті до системи у фазі Збору та аналізу вимог. Наприклад, маємо розробити застосунок інтернет-банкінгу, що має на увазі під собою наявність сторінки для авторизації. Написати код для сторінки, щоб вона працювала неможливо без урахування факторів безпеки, зручності (напр.: \* замість цифр у полі пароля, підсвічуються чи поля тощо) та інше. Всі вимоги до системи використовуються від самого початку, тобто код одразу пишеться з урахуванням всіх вимог, і якщо не враховувати нефункціональні вимоги, то клієнт може просто не авторизуватися у системі і подальше використання системи і її функцій неможливе.

Тестування системи *чи працює* вона взагалі, без урахування того *як саме* вона має працювати - неможливе.

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

Димове тестування (Smoke testing) розглядається як короткий цикл тестів, які проводяться для підтвердження того, що після збірки коду (нового або виправленого) система запускається та виконує основні функції.

Необхідність проведення димового тестування полягає в тому, що воно допомагає зрозуміти чи можна відправляти систему на розширене тестування. Наприклад: маємо нову збірку додатка і замість димового тестування відправляємо продукт одразу 3-ом тестувальникам. Вони починають його тестити, знаходять дефекти, і кажуть. що проблема у неправильній зборці. Але ми вже задіяли 3-ох тестувальників, які припустимо витратили по 5 годин, а отже ми загубили 15 годин продуктивної праці. Якщо б ми провели димове тестування та одразу знайшли дефекти у збірці, то могли б вже виправити ці дефекти.

Димове тестування не займає багато часу, воно не вимагає щоб було задіяно багато спеціалістів, тож допомагає економити час та ресурси, у разі якщо збірка не пройшла димове тестування.

На мою думку, димове тестування є не тільки доречним, а й необхідним з вищевказаних причин.

### Рівень ІІІ

**Напиши 5 функціональних тест-кейсів, які перевіряли б роботу застосунку.**

[Тест-кейси](https://docs.google.com/spreadsheets/d/1R7YaHSO7PMmUDppGrLOyj-UZEGgINuIhiCOwNxmMoxw/edit?usp=sharing)

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

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

1. Застосунок має бути адаптований під людей з дефіцитом кольорового зору -Accessibility Testing
2. Застосунок має бути недоступний для користувачів з РБ та рф. - Internationalization Testing (але не впевнена)
3. Застосунок має бути доступний на всіх ОС - Compatibility testing
4. Застосунок повинен продовжити працювати, після згортання/розгортання додатку - Stability Testing