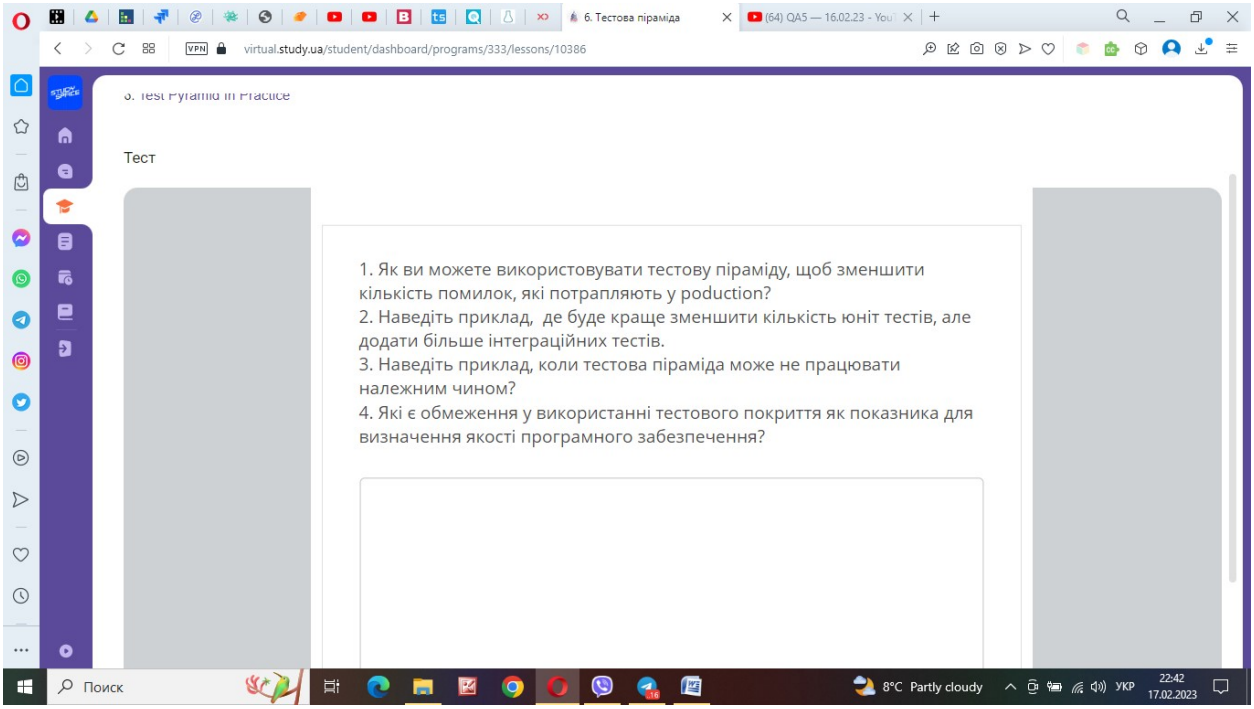
6 Test pyramid



1. По-перше, треба пам’ятати, що тести обов’язково потрібні на кожному рівні піраміди (unit tests, integration tests, end-to-end tests). Кількість тестів на кожному з рівнів може різнитися і це залежить від типу продукту, який розробляється, проте не можна виключити жоден з типів тестів. Unit tests проводяться розробниками на модульному рівні тестування та повинні бути у найбільшій кількості, оскільки це дозволяє знайти баги на початковому етапі розробки і таким чином зменшується ризик їх проникнення на вищі рівні. Не менш важливим є проведення integration tests, (які також проводяться розробниками або за допомогою автоматизованих тестів), оскільки саме ці тести відповідають за взаємозв’язок між модулями програмного забезпечення. Integration тестів також повинно бути проведено багато, проте вже в меншій кількості, ніж unit tests. Метою end-to-end tests, проведенням яких займаються вже тестувальники, є тестування системи в цілому через інтерфейс. Ці тести також є важливими, оскільки end-to-end тестами можна протестувати те, що не можуть unit tests та integration tests. End-to-end тести об’ємніші та повільніші, аніж unit tests та integration tests, саме тому їх кількість повинна бути найменшою.

Плануючи кількість тестів на кожному з рівнів піраміди, потрібно відштовхуватись від типу програмного забезпечення, яке розробляється.

2. Інтеграційні тести будуть вкрай важливими для таких програм, у яких використовуються сторонні сервіси. Наприклад, на сайті Kyivstar при поповненні рахунку використовується стороння система оплати Portmone, тому перевірити інтеграцію, взаємодію між двома системами – одна з найважливіжих задач. Або до прикладу, якщо на сайті є сервіс збереження медіа-файлів, то ці медіа-файли можуть знаходитися на сторонньому сервері і потрібно перевірити інтеграцію між сайтом і стороннім сервісом. Також бувають зовнішні сервіси аутентифікації (аутентифікація через соціальні мережі - Google, Facebook і.т.п), додатки для прослуховування музики (такі як Spotify, Google-Music), додатки для визначення геолокації.

Прикладом необхідності збільшення інтеграційних тестів у програмах без використання сторонніх сервісів може бути присутність у програмному забезпеченні таких параметрів, як зв'язок з базою даних, підтримка різних форматів файлів, відправка поштових повідомлень.

Загалом, завжди краще мати більше співвідношення модульних тестів до інтеграційних тестів. Однак є деякі випадки, коли може мати сенс мати менше модульних тестів і більше інтеграційних тестів. Наприклад, якщо ви працюєте над застарілою системою, де код дуже складно модульно тестуватиі, можливо, буде доцільніше зосередитися на інтеграційних тестах, щоб отримати краще охоплення.

3. Піраміда тестування може не працювати належним чином, якщо, наприклад, проводиться забагато end-to-end тестів, а юніт та інтеграційним тестам приділяється набагато менше уваги. End-to-end тести повільніші, більш громіздкі та коштують дорожче, отже процес розробки вповільнюється та здорожується. До того ж, помилок може виникати забагато, бо було проведено недостатню кількість тестів на модульному та інтеграційному рівнях тестування.

Також причиною того, що піраміда тестування може працювати неналежним чином є погана комунікація в команді, коли розробники рідко співпрацюють з мануальними тестувальниками та автоматизаторами тестів. Виходить, що різні працівники можуть виконувати однакові тести, а ще й по декілька разів, що знижує ефективність та продуктивність.

Ще однією причиною роботи піраміди тестування неналежним чином є дуже мала кількість або навіть повна відсутність тестів на одному з рівнів піраміди, а саме інтеграційних тестів. Перевіряються окремі компоненти (юніт-тести), а потім вже система в цілому (end-to-end тести), проте інтеграція між компонентами не перевіряється або перевіряється у недостатньому об’ємі.

Піраміда тестування може не працювати належним чином, якщо проводиться замало юніт-тестів, а більша увага приділяється інтеграційним тестам. Дуже часто розробники вважають, що юніт-тести непотрібні, а регресію відновити здатні лише інтеграційні тести. Такий підхід призводить до того, що на інтеграційному рівні виникають помилки і все одно доводиться повертатися до попереднього(модульного) рівня тестування і виправляти ці помилки. Це призводить до уповільнення роботи над проектом, отже, продукт стає більш дорожчим, а робота – менш ефективною.

Одним із прикладів, коли тестова піраміда може не працювати належним чином, коли є дуже велика та складна програма. У цьому випадку може бути важко створити модульні тести, які охоплюють усі функціональні можливості, і може бути ефективнішим зосередитися на створенні меншої кількості, але більш комплексних інтеграційних тестів.

4. Одним із обмежень у використанні тестового покриття як показника для визначення якості ПО є той факт, що 100 % тестового покриття не може бути згідно з принципом тестування «Вичерпне тестування неможливе».

Що стосується проведення метрики покриття коду, то ця метрика не завжди дає гарний результат того, наскільки ефективно може бути протестована програма в залежності від кількості рядків коду.