Завдання 1.1

	Функціональне тестування	Нефункціональне тестування	Тестування пов'язане зі змінами
Що перевіряється	Перевіряє, що програмне забезпечення функціонує так, як очікується, відповідно до заданих вимог. Тестуються конкретні функціональні можливості. Відповідає на питання: **Що система робить?** Чи працює ця конкретна функція?	Тестування нефункціональних вимог системи, таких як продуктивність, безпека, масштабованість, зручність використання, надійність та інші атрибути якості програмного забезпечення. Відповідає на питання **Як добре система працює?**	Передбачає перевірку **впливу змін** на існуючу програмну систему. Цей тип тестування спрямований на те, щоб переконатися, що внесені зміни, чи то нові функції, оновлення, виправлення або модифікації, не впливають негативно на загальну функціональність, продуктивність або якість системи.
Коли застосовується	Протягом усього життєвого циклу розробки програмного забезпечення, щоб гарантувати, що продукт відповідає заданим функціональним вимогам. Функціональні тести можуть проводитися **на усіх рівнях тестування** (компонентному, інтеграційному, системному, приймальному).	Зазвичай його проводять після функціонального тестування, щоб переконатися, що програмне забезпечення відповідає бажаним нефункціональним вимогам. Нефункціональне тестування проводиться на різних етапах життєвого циклу розробки програмного забезпечення. Грунтовне тестування проводиться **від рівня інтеграції і вище.**	Використовується щоразу, коли в програмну систему вносяться зміни. (щоб переконатися, що зміни були впроваджені правильно і не порушують жодної існуючої функціональності) **На будь-якому рівні тестування**
Обмеження	***Обсяг***: Функціональне тестування фокусується на перевірці поведінки програмного забезпечення на основі заданих вимог. Однак воно може не охоплювати всі можливі сценарії, граничні випадки або взаємодію з користувачем, що призводить до потенційних прогалин у тестовому покритті. ***Глибина:*** Функціональне тестування зазвичай тестує продукт на рівні користувацького інтерфейсу, що може не виявити більш глибокі проблеми в системі, такі як вузькі місця в продуктивності, вразливості безпеки або проблеми інтеграції. Логічні помилки в програмному забезпеченні можуть бути пропущені під час проведення функціонального тестування. ***Повторювані завдання:*** Функціональне тестування часто передбачає повторне виконання тестів. Це може зайняти багато часу і призвести до ігнорування певних дефектів через людські помилки або втому. Існує висока ймовірність надлишкового тестування. ***Обмежена автоматизація:*** Хоча функціональне тестування можна автоматизувати для підвищення ефективності та узгодженості, не всі аспекти функціонального тестування можна легко автоматизувати, наприклад, візуальну перевірку або перевірку складної бізнес-логіки. ***Залежність від вимог:*** Функціональне тестування значною мірою залежить від правильності та повноти задокументованих вимог. Якщо вимоги неточні, неоднозначні або неповні, це може призвести до неефективного тестування і потенційного пропуску критичних дефектів.	***Суб'єктивність:*** Деякі нефункціональні аспекти, такі як зручність використання або досвід користувача, можуть бути суб'єктивними і змінюватися залежно від індивідуальних уподобань користувача. Ця суб'єктивність може ускладнити визначення чітких критеріїв проходження/непроходження таких тестів. ***Складність:*** Нефункціональне тестування часто включає в себе складні сценарії, такі як навантажувальне тестування або тестування безпеки, які можуть бути складними в налаштуванні та виконанні. Така складність може вимагати спеціалізованих інструментів, досвіду та ресурсів. ***Взаємозалежності:*** Нефункціональні аспекти системи, такі як продуктивність або масштабованість, можуть бути взаємозалежними і залежати від різних факторів. Тестування одного аспекту ізольовано може не відображати реальну поведінку системи. ***Вартість:*** Проведення комплексного нефункціонального тестування, особливо для таких аспектів, як продуктивність або безпека, може бути ресурсоємним з точки зору часу, зусиль і вартості. Балансування між потребою в ретельному тестуванні та наявними ресурсами може бути складним завданням.	***Обсяг:*** Тестування, пов'язане зі змінами, може зосереджуватися переважно на сферах, на які безпосередньо впливають зміни, і потенційно не враховувати непрямий вплив на інші частини системи. Це може призвести до того, що деякі проблеми залишаться невиявленими. ***Проблеми регресії:*** Зміни, внесені під час тестування, іноді можуть ненавмисно призвести до появи нових помилок або викликати проблеми регресії в раніше працюючих частинах програми. Виявити всі проблеми регресії може бути складно, особливо якщо обсяг тестування обмежений. ***Обмеження ресурсів:*** Обмежений час, бюджет або ресурси можуть обмежити обсяг тестування, пов'язаний зі змінами. Це може призвести до неповного тестового покриття, залишаючи деякі аспекти неперевіреними. ***Залежність від документації:*** Ефективне тестування змін залежить від точної та актуальної документації змін. Якщо документація неадекватна або неточна, це може призвести до прогалин у процесі тестування.

Особливості	***Тестування на основі вимог:*** Функціональне тестування виконується на основі функціональних вимог або специфікацій програмного додатку. Тестові кейси призначені для перевірки того, що програмне забезпечення функціонує так, як очікується відповідно до цих вимог. ***Тестування "чорного ящика":*** Функціональне тестування часто проводиться з зовнішньої точки зору без знання внутрішнього коду програми. Тестувальники зосереджуються на входах, виходах і поведінці програмного забезпечення. ***Перевірка конкретних функцій:*** Функціональне тестування перевіряє конкретні функції, можливості або модулі програмного додатку. Тестові кейси призначені для перевірки окремих функцій, щоб переконатися, що вони працюють за призначенням і що програмне забезпечення поводиться правильно на основі різних дій користувача. ***Тестування на основі даних:*** Функціональне тестування може включати тестування на основі даних, коли тестові кейси виконуються з різними наборами тестових даних для перевірки функціональності за різних умов.	***Атрибути якості:*** Нефункціональне тестування оцінює такі атрибути якості, як продуктивність, масштабованість, надійність, зручність використання, безпека та відповідність вимогам. ***Взаємодія з користувачем:*** Нефункціональне тестування оцінює загальний досвід користувача, включаючи такі фактори, як час відгуку, швидкість і простота використання. ***Масштабованість і продуктивність:*** Нефункціональне тестування перевіряє здатність програмного забезпечення працювати в різних умовах, таких як навантаження, стрес, паралельність і масштабність. ***Безпека та відповідність вимогам:*** Нефункціональне тестування гарантує, що програмне забезпечення є безпечним, відповідає нормативним вимогам, а також захищає дані користувачів і конфіденційність. ***Надійність і стабільність:*** Нефункціональне тестування перевіряє надійність, стабільність і доступність програмного забезпечення за різних обставин.	***Валідація змін:*** Основна увага при тестуванні, пов'язаному зі змінами, приділяється перевірці модифікацій, внесених в програмне забезпечення. Це передбачає перевірку того, що зміни були впроваджені правильно і не вносять ніяких нових дефектів. ***Сфокусований обсяг:*** Тестування, пов'язане зі змінами, спеціально націлене на області програмного забезпечення, які були модифіковані або на які вплинули нещодавні зміни. Воно спрямоване на те, щоб переконатися, що ці модифікації не створюють нових проблем і не порушують існуючу функціональність. ***Повторюваний характер:*** Регресійне тестування часто передбачає запуск одних і тих же тестів кілька разів, щоб переконатися, що зміни, внесені в програмне забезпечення, не мають непередбачуваних наслідків.

Завдання 1.2

Різниця між регресією та ретестингом

Регресійне тестування виконується для того, щоб переконатися, що нещодавні зміни в коді не вплинули негативно на існуючу функціональність програмного забезпечення. А ретестинг проводиться для перевірки того, що певний дефект або проблема, виявлена під час попереднього циклу тестування, була виправлена правильно. Регресійне тестування ми проводимо після внесення змін до коду, впровадження нових функцій або виправлень, і перевіряємо, чи все працює так як треба, чи не вплинули ці зміни на роботу систему, чи не з’явилися нові дефекти у вже протестованому продукті після внесених змін, чи «не зламалося» те, що раніше працювало. А ретестинг проводиться після того, як команда розробників виправила виявлений дефект, і ми ретестимо лише конкретний функціонал, в якому була виявлена і виправлена проблема. При ретестингу використовується той же тест-кейс, який виявив дефект. Тест-кейси регресійного тестування можуть бути отримані з функціональних вимог або специфікацій, і проводяться незалежно від того, що виправили розробники.

Отже, ретест перевіряє, що дефект виправлений, а регресійне тестування перевіряє, що свіжі зміни в коді не надали побічних ефектів на існуючу функціональність.

Завдання 2.1

У деяких випадках продукт може проходити функціональне тестування без явного тестування нефункціональних вимог. Передусім, це залежить від розміру і складності проекту. Така ситуація може виникнути у наступних сценаріях:

- Якщо проект розробляється тільки під одну систему, точково від одну платформу, на дуже простих проектах, на так званих нативних додатках, що інтегруються та встановлюються в конкретну систему (наприклад, вбудований калькулятор), то можна обійтися лише функціональним тестуванням.

- Якщо обмежені ресурси: Через обмеженість ресурсів проект може надавати пріоритет функціональному тестуванню для перевірки основних можливостей і функціональності програмного забезпечення. Нефункціональне тестування, таке як тестування продуктивності або безпеки, може бути відкладене або зведене до мінімуму.

- Ранні етапи розробки: На початкових етапах розробки основна увага може бути зосереджена на забезпеченні відповідності програмного забезпечення функціональним вимогам, перш ніж заглиблюватися в нефункціональні аспекти. Нефункціональне тестування можна відкласти на пізніші етапи.

Однак важливо зазначити, що нехтування нефункціональним тестуванням може мати згубні наслідки для загальної якості та продуктивності програмного продукту. Нефункціональні вимоги, такі як продуктивність, безпека, зручність використання та масштабованість, мають вирішальне значення для створення міцного та надійного продукту, який відповідає очікуванням користувачів. Ігнорування нефункціонального тестування може призвести до таких проблем, як низька продуктивність, вразливість системи безпеки або проблеми з юзабіліті, які можуть вплинути на задоволеність користувачів і успіх продукту на ринку. Тому, хоча функціональне тестування є важливим, воно повинно доповнюватися ретельним нефункціональним тестуванням, щоб забезпечити загальну якість і надійність програмного продукту.

Завдання 2.2

Smoke testing - тестування, яке проводиться на початковому етапі та спрямоване на перевірку готовності продукту до проведення більш розширеного тестування. Його мета перевірити «стабільність» системи, щоб дати зелене світло проведенню ретельнішого тестування. Якщо ми не проведемо димове тестування на ранніх стадіях, надалі можуть виникнути дефекти. І на етапах пізніше, це може вже дорого коштувати. І дефекти знайдені у пізніших етапах можуть стати стопорами там, де це може вплинути на випуск результатів. Простіше кажучи, димове тестування - це дешевший і швидший спосіб переконатися, що основні найкритичніші функції програмного забезпечення працюють належним чином, перш ніж воно буде протестоване далі і завантажене для загального доступу.

Хоча димове тестування - це швидкий і ефективний спосіб забезпечити стабільність роботи програми та виявити основні дефекти на ранніх стадіях, варто пам’ятати, що воно не може виявити всі проблеми з програмним забезпеченням. Натомість його слід використовувати в поєднанні з іншими методами тестування, щоб забезпечити найкращий досвід користування програмою для споживачів. Тим не менш, димове тестування є важливою частиною процесу тестування програмного забезпечення і чудовим способом оптимізувати зусилля з тестування.

Можуть бути випадки, коли проект дуже маленький або коли є серйозні часові обмеження. У таких випадках команда може зосередитися на інших видах тестування або оптимізувати процес тестування відповідно до вимог проекту, ґрунтуючись на конкретних потребах і обмеженнях проекту.

Але все ж таки я вважаю, що димове тестування повинно бути настійно рекомендовано, оскільки воно забезпечує початковий рівень впевненості в стабільності роботи програми після змін або розгортання, і все ж таки повинно проводитися для кожної збірки, оскільки воно допомагає виявити дефекти на ранніх стадіях. Це стосується нових розробок, а також основних і другорядних випусків системи, після внесення великих змін.