**Лабораторна робота № ЛР11-1 Підготовка тестів для перевірки ПЗ – перевірка інтерфейсу користувача**

**Мета** – ознайомитися з особливостями GUI тестування та UI тестування. Набути навичок проведення автоматизованого GUI тестування та UI тестування інтерактивного прототипу веб-додатку.

**Методичні рекомендації до виконання лабораторної роботи.**

1. Опрацювати теоретичні відомості.
2. Створити декілька тест-кейсів для тестування графічного інтерфейсу обраного додатку або власного проекту.
3. Провести тестування, обравши будь-який засіб тестування, який запропоновано в даній лабораторній роботі (або обрати самостійно).
4. Оформити звіт з виконання лабораторної роботи та захистити
5. Робота повинна бути виконана згідно критеріїв оформлення документації та повинна містити

* Назва лабораторної роботи.
* Прізвище, група
* Назва проекту.
* Результати роботи оформлюються у вигляді таблиці:

По закінченню практичну роботу потрібно здати на перевірку викладачеві, надіславши електронною поштою на адресу [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com) . Якщо викладач знаходить помилки чи неточності, він може повернути роботу на доопрацювання.

Файл з роботою повинен мати назву в такому форматі:

**QA<Номер групи><Номер лекції / практичної / лабораторної>[-<Номер завдання>][літера позначення типу роботи L – лекція, P –практична, R – лабораторна]<Прізвище англійською>**.. Наприклад, **QA4101Р**buts.doc.

Не копіюйте фрагментів з різних інформаційних джерел, подумайте і викладіть свою точку зору. При наявності робіт-"близнюків" відповідь буде зараховуватися першому за часом надсилання.

Тему в заголовку листа записати

**QA <Номер групи>-ЛР<Номер лабораторної>-<Прізвище >**

**Строк виконання цієї роботи ІПЗ-41 – 27.04.2024**

**Теоретичні відомості**

Інтерфейс користувача, або UI, — це платформа, яку ви використовуєте для взаємодії з певним програмним забезпеченням. Інтерфейс користувача – це місце, де ви можете вводити інструкції, вводити дані або переглядати інформацію з екрана чи монітора.

Існує багато різних типів інтерфейсу користувача, включаючи графічний інтерфейс користувача (GUI) та інтерфейси командного рядка, які показують лише код і текст.

Інтерфейс користувача проти графічного інтерфейсу користувача

Щоб допомогти вам краще зрозуміти ці дві форми взаємодії комп’ютера, погляньте на пряме порівняння між інтерфейсом і графічним інтерфейсом нижче:

**UI:**

• Абревіатура інтерфейсу користувача

• Це тип платформи, який дозволяє користувачам взаємодіяти з пристроями

• Це форма взаємодії людини і машини

• Він використовується всіма і часто працює у фоновому режимі, тому ви не знаєте, що використовуєте його

• Типовими прикладами є MS-DOS або Unix

**GUI:**

• Абревіатура графічного інтерфейсу користувача

• Це тип платформи, яка використовує графіку, щоб допомогти користувачам орієнтуватися у функціях пристрою

• Це підклас UI

• Його зазвичай використовують пересічні повсякденні користувачі, наприклад споживачі

• Типовими прикладами є Windows 10, iOS і Android

Тестування інтерфейсу користувача зазвичай є найефективнішим до випуску програми для виробництва. Це робиться для того, щоб кінцевий користувач мав найкращий досвід із якомога меншою кількістю помилок і дефектів.

Кінцеві користувачі не є найкращими тестувальниками програмного забезпечення, тому важливо вирішувати будь-які проблеми, перш ніж вони дійдуть до них.

Тестування інтерфейсу користувача — це корисний спосіб оцінити, як програма справляється з певними діями, як-от використання клавіатури та миші для взаємодії з меню. Це допомагає перевірити візуальні елементи програми, щоб переконатися, що вони відображаються правильно. Тестування інтерфейсу користувача також є чудовим способом оцінити продуктивність і переконатися, що немає помилок або проблем із функціональністю програми.

**Типи UI тестів**

Існує безліч різних тестів інтерфейсу користувача, які слід розглянути залежно від програми, що тестується. Тести інтерфейсу користувача мають потенціал для перевірки багатьох функцій у програмах, тому вибір правильного типу тесту може допомогти виявити конкретні проблеми. Деякі з найпоширеніших методологій функціонального та нефункціонального тестування включають наступне:

***1. Регресійне тестування***

Регресійне тестування — це тип тестування інтерфейсу користувача, який розглядає будь-які зміни в коді програми або веб-сайту.

Це гарантує, що всі функціональні можливості програми відповідають призначенню після внесення змін до частин коду.

Йому не потрібно проводити жодних хитрих тестів, він просто запускає код, щоб переконатися, що всі залежності та функції працюють так само, як і до внесення змін.

***2. Функціональне тестування***

Функціональне тестування має на меті перевірити програму, щоб переконатися, що вона відповідає всім функціональним вимогам.

Воно перевіряє всі окремі функції програми, а потім перевіряє результат, щоб переконатися, що вона працює належним чином.

Цей тип тестування інтерфейсу користувача зазвичай зосереджується на тестуванні чорного ящика, який не розглядає вихідний код. Він має тенденцію перевіряти такі речі, як інтерфейс користувача, будь-які пов’язані API, зв’язок між клієнтом і сервером або безпеку.

***3. Приймальні випробування***

Приймальне тестування, іноді відоме як тестування прийнятності користувача (UAT), — це форма тестування інтерфейсу користувача, яка виконується кінцевим користувачем програми для перевірки системи перед переходом до робочого режиму.

Цей тип тестування інтерфейсу користувача найчастіше зустрічається на останніх етапах тестування після перевірки інших областей.

Приймальне тестування використовується для перевірки загального потоку програми від початку до кінця. Він не розглядає проблеми поверхневого рівня, такі як орфографічні помилки чи естетичні проблеми. Він використовує окреме середовище тестування, щоб імітувати виробниче середовище, гарантуючи готовність переходити до наступного етапу.

***4. Модульне тестування***

Модульне тестування спрямоване на перевірку окремих компонентів програми, щоб переконатися, що вона працює належним чином.

Зазвичай це виконується на етапі кодування, тому виконання цього типу тестування інтерфейсу користувача зазвичай покладається на розробників.

Модульне тестування працює шляхом відокремлення частини коду, щоб переконатися, що він працює належним чином. Цей окремий фрагмент коду може бути певним модулем, функцією, об’єктом або будь-якою іншою окремою частиною програми.

***5. Тестування продуктивності***

Тестування продуктивності має на меті оцінити оптимізацію програми, розглядаючи такі речі, як швидкість, стабільність, чуйність і масштабованість програми під час використання.

Цей тип тестування інтерфейсу користувача спрямований на пошук проблемних областей у додатку або вузьких місць у потоці даних. Три основні сфери, на які він дивиться, це швидкість, масштабованість і стабільність програми.

***6. Тестування GUI***

Інструменти тестування графічного інтерфейсу перевіряють графічний інтерфейс користувача програми, щоб переконатися, що всі функції працюють належним чином.

Це включає перегляд графічних ресурсів і елементів керування програми, таких як кнопки, панелі інструментів і значки. Графічний інтерфейс користувача – це те, з чим взаємодіє кінцевий користувач і що бачить під час використання програми.

**Тестування графічного інтерфейсу** - це тип тестування програмного забезпечення, який перевіряє графічний інтерфейс користувача програмного забезпечення. Мета тестування графічного інтерфейсу користувача (GUI) полягає у забезпеченні функціональних можливостей програмних додатків відповідно до специфікацій шляхом перевірки екранів та елементів керування, таких як меню, кнопки, піктограми тощо. GUI - це те, що бачить користувач.

Користувач не бачить вихідний код. Інтерфейс видно користувачеві. Особливо увага приділяється структурі дизайну, зображенням того, чи працюють вони належним чином чи ні.

Вимоги до зовнішнього вигляду інтерфейсу та форм взаємодії з користувачем:

* Вимоги до розміщення елементів управління на екранних формах
* Вимогу до змісту та оформленню повідомлень, які виводяться на екран
* Вимоги до форматів вводу
* Вимогу до реакції системи на ввід даних користувачем
* Вимоги до часу відгуку системи на команди користувача

Тестування графічного інтерфейсу в основному передбачає:

• Тестування розміру, положення, ширини, висоти елементів.

• Тестування повідомлень про помилки, які відображаються.

• Тестування різних розділів екрана.

• Тестування шрифту, читабельний він чи ні.

• Тестування екрану в різних роздільних здатностях за допомогою збільшення та зменшення масштабу, як 640 x 480, 600x800 тощо.

• Тестування вирівнювання текстів та інших елементів, таких як піктограми, кнопки тощо, знаходяться у належному місці чи ні.

• Тестування кольорів шрифтів.

• Тестування кольорів повідомлень про помилки, попереджувальних повідомлень.

• Перевірка того, чи має зображення хорошу чіткість чи ні.

• Тестування вирівнювання зображень.

• Перевірка правопису.

• Перевірка привабливості інтерфейсу.

• Тестування смуг прокрутки відповідно до розміру сторінки, якщо така є.

• Тестування відключених полів, якщо такі є.

• Тестування розміру зображень.

• Перевірка заголовків, чи правильно він вирівняний чи ні.

• Тестування кольору гіперпосилання.

**Рівні покриття для тестування**:

* + Функціональне покриття – покриття вимог до інтерфейсу користувача
  + Структурне покриття – кожний інтерфейсний елемент повинен бути використаний в тестових прикладах хоча б один раз
  + Структурне покриття з врахуванням стану елементів інтерфейсу – необхідно не тільки використати всі елементи інтерфейсу, а задати йому всі можливі стани.
  + Структурне покриття із врахуванням стану елементів інтерфейсу та внутрішнього стану системи – поведінка деяких інтерфейсних елементів може залежати від внутрішнього стану системи.

**Методи проведення тестування**:

* + Вручну - реалізація операторів описаних тестових сценаріїв. Переваги: виконання людиною, не будуть враховуватися «косметичні» зміни. Недоліки: значні ресурси.
  + За допомогою інструментарію (автоматизація) – використання програмних інструментів, які емалюють поведінку тестувальника-оператора.

**Фактори, що впливають на тестування зручності використання інтерфейсів**:

* + Легкість навчання
  + Ефективність навчання
  + Запамятовуваність навчання
  + Помилки
  + Загальна задоволеність.

**Етапи тестування зручності використання інтерфейсу користувача**:

* + Дослідницьке – після формулювання вимог до системи та розробника прототипу інтерфейсу
  + Оціночне – після розробки низькорівневих вимог та деталізованого прототипу інтерфейсу.
  + Валідаційне – ближче до етапу завершення розробки
  + Порівняльне – на будь-якому етапі з метою порівняння декількох версій.

Інструменти тестування графічного інтерфейсу

Нижче наведено список популярних засобів тестування графічного інтерфейсу :

• Ranorex

• Selenium

• QTP

• SilkTest

• TestComplete

• Squish GUI Tester

**Основні переваги тестування інтерфейсу користувача**.

***1. Покращує функціональність***

Важливо тестувати програми, щоб переконатися, що вони працюють належним чином, щоб у разі виникнення збоїв, помилок чи інших проблем їх можна було вирішити до випуску.

Якщо програма досягає кінцевих користувачів і має помилки, повна помилок або зламана, вона не виконуватиме роботу, яка від неї очікується. Це, у свою чергу, створює занадто багато проблем для кінцевих користувачів, і вони, швидше за все, перестануть ним користуватися.

***2. Це полегшує використання***

Інструменти автоматизації тестування інтерфейсу також є корисним способом оптимізації та спрощення програми.

Навіть якщо все кодування працює належним чином, погано спроектований інтерфейс може заплутати кінцевих користувачів і швидко вимкнути їх, знизивши рівень адаптації програми. Тестування інтерфейсу користувача — це чудовий спосіб покращити будь-які елементи чи варіанти дизайну, щоб їх було легше використовувати.

***3. Зміцнює репутацію програми***

Витрачення часу на належне тестування інтерфейсу користувача та залучення таких інструментів, як програмне забезпечення для автоматизації тестування, є чудовими способами вдосконалити програму та зробити її максимально зручною для користувача.

Якщо все зроблено правильно, це робить додаток чудовим амбасадором бренду, що підвищує його загальну репутацію. Якщо програма працює без помилок і виконує все, що вона повинна робити, то користувачі оцінять це і використовуватимуть програму.

**Основні проблеми тестування інтерфейсу користувача**

Хоча тестування інтерфейсу користувача є важливою частиною розробки програми, це не обов’язково легка частина процесу.

Існує ряд проблем і проблем, пов’язаних із безкоштовним програмним забезпеченням для автоматизації тестування інтерфейсу користувача, що ускладнює його роботу.

Нижче наведено деякі з основних проблем, пов’язаних з тестуванням інтерфейсу користувача при використанні неадекватних інструментів тестування інтерфейсу користувача:

***1. Оновлення інтерфейсу***

Розробка додатків зазвичай є ітеративним процесом, який привносить нові можливості та функції протягом циклу розробки та за його межами.

Усі ці спорадичні зміни можуть ускладнити ефективне виконання тестів інтерфейсу користувача, оскільки інші залежності та взаємодії коду змінюють те, що тестується.

***2. Тестування, яке стає складнішим***

Програми та веб-сайти зараз набагато досконаліші, ніж навіть кілька років тому. З усією цією додатковою функціональністю інструменти тестування інтерфейсу користувача та програмне забезпечення для автоматизації інтерфейсу користувача мають вивчати більше елементів і процесів.

Як наслідок, багато інструментів у тестуванні інтерфейсу користувача потрібно налаштувати, щоб врахувати всі ці складні доповнення.

***3. Часові обмеження***

Оскільки додатки стають все складнішими, інструменти, які використовуються для тестування, теж ускладнюються. Сценарії тестування інтерфейсу користувача стають набагато трудомісткішими через величезний обсяг коду, який потрібно протестувати. Ця проблема ускладнюється, коли правильні інструменти тестування інтерфейсу користувача недоступні.

***4. Підтримка сценаріїв інтерфейсу користувача в актуальному стані***

Оскільки інтерфейс користувача змінюється та з’являються нові функції, тестові сценарії потрібно налаштовувати для перевірки нових процесів. Це стає складнішим із кожним новим доповненням, оскільки тестові сценарії постійно оновлюються та налаштовуються, щоб відповідати новим функціям.

**ручне тестування інтерфейсу користувача**

Ручне тестування, на відміну від автоматизації інтерфейсу користувача, передбачає використання тестувальника для ручної взаємодії та перевірки всіх функцій, знайдених у програмі чи веб-сайті.

Їхнє головне призначення — виявляти будь-які проблеми, невідповідності чи проблеми в загальній програмі. Це особливо корисний параметр для невеликих програм з обмеженими елементами, як-от ті, що є в ранніх версіях програм.

1. Переваги ручного тестування інтерфейсу користувача

Вибір ручного тестування інтерфейсу користувача має багато переваг залежно від програми та її дизайну.

**Переваги ручного тестування інтерфейсу користувача**.

• Тестування інтерфейсу користувача вручну використовує людський інтелект для виявлення помилок або проблем. Є речі, які автоматизоване тестування інтерфейсу користувача просто не може виконати, і потрібна людська взаємодія, критичне мислення та людський фактор, щоб знайти всі недоліки програми.

• Автоматизовані тести можуть займати досить багато часу, оскільки вони відтворюють кілька сценаріїв для різних функцій, які має перевірити тестувальник. Тестування інтерфейсу користувача вручну дозволяє людям-тестерам зосередитися на пошуку недоліків, а не налаштовувати емуляції.

• Люди-тестери, як правило, добре знають програму, часто витрачаючи незліченні години на звикання до інтерфейсу. Саме завдяки цьому вони розуміють, на що слід звернути увагу щодо помилок, допомагаючи їм бути в курсі поточного стану програми.

• Є проблеми, які не можуть бути позначені автоматизованим тестуванням інтерфейсу користувача, оскільки це не впливає на код. Такі речі, як час відповіді сервера, можуть затримуватися, але їх можна легко проігнорувати під час автоматичного тестування. Ручне тестування інтерфейсу користувача усуває цю проблему, оскільки користувач-людина відразу помічає ці проблеми.

• Тестування інтерфейсу користувача вручну є найточнішою емуляцією взаємодії з користувачем, оскільки ви створюєте ситуацію, яка відображає те, як кінцевий користувач буде взаємодіяти з програмою. Це створює реальний контекст для пошуку проблем, які зазвичай виявляють кінцеві користувачі, але, можливо, пропускаються під час автоматизованого тестування інтерфейсу користувача.

**Обмеження ручного тестування інтерфейсу користувача**

Існують також обмеження щодо ручного тестування інтерфейсу користувача, які слід враховувати, перш ніж приймати рішення щодо найкращого підходу до тестування для вашої програми.

Деякі з обмежень ручних тестів інтерфейсу користувача включають наступне:

• Виконання ручного тестування займає набагато більше часу, ніж автоматизоване тестування інтерфейсу користувача, особливо при використанні сучасних інструментів, таких як [гіперавтоматизація](https://www.zaptest.com/hyperautomation). Сценарії для автоматизованого тестування можуть працювати набагато швидше, ніж будь-який тип введення людиною, тому вибір ручного тестування веб-інтерфейсу додає додаткові години до розкладу.

• Оскільки це, зрештою, людський процес, ручне тестування веб-інтерфейсу користувача схильне до помилок людини. Під час тестування інтерфейсу користувача вручну можуть виникнути помилки, які ви пропустили через недостатню увагу або відволікання, що може призвести до проблем. Для порівняння, автоматизоване тестування інтерфейсу користувача виключає людський елемент із процесу, роблячи його набагато менш схильним до подібних проблем. Це особливо вірно для останніх типів автоматизованого тестування інтерфейсу користувача, наприклад [роботизованої автоматизації процесів](https://www.zaptest.com/a-complete-guide-to-robotic-process-automation-rpa) .

• Фактичний процес реєстрації будь-яких знайдених помилок займає набагато більше часу, що може ускладнити відстеження будь-яких змін у міру їх внесення. Автоматизоване тестування інтерфейсу користувача є кращим підходом, оскільки воно потребує оновлення, лише якщо реалізовано нову функцію.

• Ручне тестування інтерфейсу користувача вимагає глибокого знання програми для компетентного тестування на наявність проблем. У результаті цього тестувальникам-людям потрібен певний рівень знань, перш ніж вони зможуть ефективно тестувати. [Автоматизоване тестування](https://www.zaptest.com/a-complete-guide-to-software-testing-automation) не вимагає такого рівня знань.

3. Тестування запису та відтворення

Тестування запису й повтору — це форма безкодового тестування інтерфейсу користувача, яка дозволяє запускати тести без глибоких знань програмування. Він використовує функції для запису ручних дій, які виконуються в додатку, перш ніж зберегти його як тестовий шаблон.

Це дозволяє запускати тест інтерфейсу користувача знову і знову без участі людини.

**Тестування вручну проти запису та відтворення та проти автоматизації**

Вибираючи між цими трьома типами тестування інтерфейсу користувача, важливо враховувати обсяг і масштаб програми та доступні ресурси.

Тестування інтерфейсу користувача вручну найлегше налаштувати та використовувати, але воно має багато вимог, як-от добре знання програми тестувальником. Також важко продовжувати ручне тестування інтерфейсу користувача, якщо ви постійно оновлюєте програму.

Інструменти автоматизації тестування інтерфейсу користувача є чудовим варіантом, якщо ви збираєтеся регулярно оновлювати програму, і з часом це окупиться.

Щоб подолати розрив між двома типами тестування інтерфейсу користувача, використовується запис і відтворення. Він пропонує базовий рівень автоматизації інтерфейсу користувача, але все одно вимагає участі людини, щоб його ввімкнути.

**Що ви перевіряєте під час тестування інтерфейсу користувача?**

Хоча фактичні процеси, які потрібно перевірити, відрізняються від програми до програми, існує ряд загальних речей інтерфейсу користувача, які потрібно перевірити, наприклад:

***1. Помилки в типах даних***

Цей тест інтерфейсу користувача гарантує, що правильний тип даних працює у відповідних полях. Наприклад, текст для імен без можливості використання цифр. Якщо тестер інтерфейсу користувача може вводити числові значення під поле імені, це означає, що щось не так.

***2. Проблеми з шириною поля***

Це використовується для обмеження кількості символів для певних полів, наприклад поштових індексів. Якщо програма не обмежує кількість символів у цих полях, це може призвести до недійсних введень від кінцевого користувача.

***3. Кнопки***

Ці тести інтерфейсу користувача перевіряють правильність роботи кнопок, тому, наприклад, кнопка наступної сторінки спрямовує кінцевого користувача на наступну сторінку. Існує багато різних типів кнопок із різним призначенням, тому важливо, щоб вони виконували свою роботу, щоб створити функціональну програму.

***4.Прокрутка таблиці***

Якщо в програмі є будь-які таблиці з даними, прокручування таблиці гарантує, що ви можете прокручувати дані, залишаючи видимими заголовки.

Якщо це не працює, це заплутає дані для кінцевого користувача.

***5. Журнали помилок***

У разі збою або помилки програми важливо перевірити журнали помилок, щоб переконатися, що вони надають точні результати для звітів про помилки.

Без точних звітів про помилки та журналів помилок неможливо визначити, що є причиною проблеми або як її виправити.

**Виконання тесту графічного інтерфейсу користувача (GUI)**

Щоб дати вам гарне уявлення про те, як провести тестування інтерфейсу користувача або графічного інтерфейсу користувача, ми створимо для вас приклад.

Припустимо, ми будемо тестувати сторінку форми в додатку для реєстрації облікового запису. На цій сторінці є кілька елементів інтерфейсу користувача, позначених як TC-X (де TC означає тестовий випадок, а X — номер елемента).

**Нижче наведено список доступних TC для перевірки:**

**TC-1: логотип бренду у верхній частині екрана**

• Це слід перевірити, щоб переконатися, що він відображає правильну позицію, тип шрифту та підпис сторінки.

**TC-2: Зареєструйте свій обліковий запис**

• Це повинно перевірити, чи вірний верхній колонтитул сторінки.

• Слід також перевірити, чи відображається правильний шрифт.

**TC-3: Поле імені**

• Це повинно перевірити правильне вирівнювання та положення текстового поля.

• Він також повинен протестувати мітки полів і перевірити, чи приймає дійсні та відхиляє неприпустимі записи.

**TC-4: Поле прізвища**

• Це повинно перевірити правильне вирівнювання та положення текстового поля.

• Він також повинен протестувати мітки полів і перевірити, чи приймає дійсні та відхиляє неприпустимі записи.

**TC-5: поле імені користувача**

• Це повинно перевірити, яке повідомлення про помилку відображається під час введення заборонених символів.

• Слід також перевірити, чи повідомлення про помилку є дійсним і точним.

**TC-6: Поле пароля**

• Це повинно перевірити мітки полів, щоб переконатися, що вони приймають дійсні символи та відхиляють недійсні.

• Слід також перевірити вирівнювання та положення текстового поля.

**TC-7: кнопка наступної сторінки**

• Це має перевірити, чи надсилання форми працює належним чином.

• Слід також перевірити розташування кнопок і переконатися, що їх читає користувач.

**План тестування інтерфейсу користувача**

План тестування інтерфейсу користувача – це документ, який є частиною процесу тестування програм.

План тестування інтерфейсу користувача розбиває ключову інформацію про програму та будь-які дії тестування, пов’язані з нею.

Створення плану тестування зазвичай є одним із перших кроків під час тестування програм, оскільки він закладає основу для методологій тестування та очікуваних результатів.

Це корисний документ, який дає тим, хто не входить до команди тестування, краще зрозуміти, що відбувається в процесі.

Плани тестування інтерфейсу користувача пропонують чудові вказівки та інструкції для тестувальників інтерфейсу користувача, тож правильний вибір дійсно допомагає під час тестування та перевірки програм.

Перегляньте наведені нижче кроки, щоб дізнатися, як написати план тестування інтерфейсу користувача.

1. Включіть ключову інформацію про тестування інтерфейсу користувача

План тестування інтерфейсу користувача включає всю ключову інформацію, необхідну для проведення тестування програми. Деякі з цієї інформації включають наступне:

• Фахівці, необхідні для тестування, їхні ролі та навички.

• Загальний час, необхідний для тестування програми.

• Техніки тестування, які застосовуються до тесту.

• Будь-які ресурси, необхідні для тестування, наприклад спеціальне обладнання, документація чи інструменти.

• Розбивка цільових тестових середовищ, таких як мобільні пристрої, конкретна операційна система або браузери.

• Загальні цілі процесу тестування.

2. Випробування на дим

Далі ви можете використовувати димове тестування, щоб допомогти створити план тестування інтерфейсу користувача. Димове тестування є корисним способом виявлення основних проблем і помилок у програмі, але воно не надто глибоко шукає проблеми.

Це техніка, яка найкраще підходить для тестування інтерфейсу користувача на верхньому рівні програми, тому вона може досить легко виявити кричущі проблеми.

3. Перевірка на працездатності

Щоб глибше досліджувати програму та знаходити менш очевидні помилки та недоліки, тестування працездатності є чудовою технікою для тестування інтерфейсу користувача.

Перевірка працездатності перевіряє будь-яке нове або змінене кодування, щоб переконатися, що воно відповідає вимогам програми.

Воно відрізняється від димового тестування тим, що воно є набагато повнішим за допомогою тестування інтерфейсу користувача, що дозволяє глибше вивчати функціональні можливості програми.

Після того, як програма пройшла перевірку диму, перевірка працездатності додає додатковий рівень перевірки.

**Сценарії тестування інтерфейсу користувача**

Щоб переконатися, що програма працює належним чином у багатьох областях і взаємодіях, важливо виконати різні сценарії тестування інтерфейсу користувача.

Нижче наведено розбивку сценаріїв тестування інтерфейсу користувача з прикладом.

1. Що таке сценарії тестування інтерфейсу користувача?

Тестовий сценарій інтерфейсу користувача — це спосіб створення документації для кількох випадків використання в програмі.

Тестовий сценарій інтерфейсу користувача використовується для опису конкретних дій, які може виконувати користувач під час використання програми.

У деяких випадках він також описує сценарій, з яким може зіткнутися користувач під час використання програми.

Сценарії тестування інтерфейсу користувача корисні, оскільки вони перевіряють, чи функціональні можливості програми працюють належним чином. Для створення корисних сценаріїв потрібне глибоке розуміння програми та вхід від клієнтів і розробників.

2. Приклад сценаріїв тестування інтерфейсу користувача

Як приклад розглянемо сценарій тестування для сторінки входу програми. Сценарій тестування інтерфейсу користувача для цього шукатиме відповідь на такі запитання:

• Чи можуть користувачі входити на платформу, використовуючи правильні облікові дані?

• Який результат використання неправильних облікових даних для входу?

• Що станеться, якщо ви використовуєте дійсне ім’я користувача, але невірний пароль?

• Що станеться, якщо залишити поля порожніми та спробувати увійти?

• Якщо є кнопка «Забули пароль», що станеться, якщо її натиснути?

• Чи всі посилання на сторінці працюють належним чином?

Відповіді на ці запитання допомагають тестувальникам інтерфейсу користувача визначити будь-які області програми, які не працюють належним чином.

Він також перевіряє, чи всі доступні дії забезпечують очікуваний результат, наприклад вхід із використанням правильних облікових даних.

**Тестові випадки інтерфейсу користувача**

Щоб розглянути окремі аспекти сценарію тестування інтерфейсу користувача, використовуються тестові випадки, щоб розбити окремі функції частин функціональності програми.

Нижче наведено підсумок того, що таке тестові випадки інтерфейсу користувача з прикладами.

1. Що таке тестові приклади інтерфейсу користувача?

Тестовий приклад інтерфейсу користувача – це серія дій, які виконуються для перевірки певної функції чи частини функціональності програми.

Тестові випадки інтерфейсу користувача розбивають етапи тестування, дані, передумови та післяумови для конкретних сценаріїв, а також перевіряють вимоги.

Тестовий приклад інтерфейсу користувача, як правило, включає дуже конкретні змінні, щоб забезпечити можливість поглибленого тестування на окремому рівні. Потім тестувальники інтерфейсу користувача порівнюють фактичні результати з очікуваними, щоб переконатися, що програма функціонує відповідно до вимог.

2. Приклади інтерфейсу та графічного інтерфейсу

Щоб допомогти вам краще зрозуміти тестові випадки інтерфейсу користувача та графічного інтерфейсу користувача, перегляньте наведені нижче приклади тестових випадків для сценарію тестування, який розглядає функціональність екрана входу:

• Перевірте поведінку системи під час введення дійсних облікових даних.

• Перевірте поведінку системи, коли використовується недійсна електронна адреса, але дійсний пароль.

• Перевірте поведінку системи, коли використовується дійсна електронна адреса, але невірний пароль.

• Перевірте поведінку системи при використанні недійсної електронної пошти та пароля.

• Перевірте поведінку системи, якщо поля залишені незаповненими.

• Перевірте посилання «забули пароль», щоб перевірити, чи воно працює належним чином.

• Перевірте поведінку системи, якщо позначено кнопку «Залишати мене в системі».

• Перевірте поведінку системи при введенні недійсного номера телефону.

Отже, усі ці приклади є індивідуальними тестовими випадками інтерфейсу користувача.

На відміну від сценарію тестування, який охоплює весь процес, тестові випадки розглядають окремі функції. Іншими словами, кожен наведений вище приклад є тестовим випадком інтерфейсу користувача, а весь список класифікується як сценарій тестування.

**Тестові сценарії інтерфейсу користувача**

Щоб отримати ще більш детальну розбивку тестування додатків, створено тестові сценарії інтерфейсу користувача, щоб надати тестувальникам більше інформації про тестові випадки та сценарії.

Нижче наведено короткий виклад того, що таке сценарії тестування інтерфейсу користувача та як їх писати.

1. Що таке сценарії тестування інтерфейсу користувача?

Тестові сценарії інтерфейсу користувача — це детальні описи тестів, які виконуються в програмі, зазвичай рядок за рядком.

Вони дуже специфічні за своєю природою з великою кількістю деталей щодо використаних тестів, даних і очікуваної функціональності програми.

Будь-які результати тестових прикладів також включені в тестові сценарії, щоб збільшити кількість інформації.

2. Як писати сценарії тестування інтерфейсу користувача

Тестові сценарії інтерфейсу користувача прості, оскільки вони просто деталізують тестові випадки.

Якщо ви включите в них наведену нижче інформацію, ви зможете отримати багато корисного від своїх тестових сценаріїв інтерфейсу користувача:

• Ідентифікатор тестового сценарію: це унікальний ідентифікатор для тестового сценарію.

• Назва: назва тестового сценарію.

• Ідентифікатор тесту: Це ідентифікатор тесту, для якого ви створюєте сценарій.

• Вимоги: це специфікації застосування обладнання, необхідного для виконання тестів.

• Процедура: це кроки, зроблені для просування вперед у тестуванні.

• Результат: це результат і остаточний результат тестування.

• Статус: це вказівка на успішність тестового сценарію – він пройшов чи не пройшов?

• Код помилки: якщо виникла проблема, код помилки докладно пояснює, в чому полягала проблема.

**Контрольний список для тестів інтерфейсу користувача**

Тепер, коли ви готові розпочати тестування інтерфейсу користувача, використовуйте контрольний список нижче, щоб створити власні тести:

1. Перевірте основні функції

Функціональне тестування — чудовий спосіб знайти такі речі, як візуальні помилки чи збої на платформі.

На цьому етапі обов’язково вкажіть біометричні дані, будь-які повідомлення та інформацію про пам’ять програми.

2. Перевірте сумісність між платформами

Щоб уникнути таких проблем, як фрагментація пристрою, яка блокує певним користувачам доступ до програми, корисно виконувати перевірки сумісності між платформами.

Це включає перевірку програми на різних роздільних здатностях екрана.

Доцільно перевірити сумісність нативних і гібридних програм на мобільних пристроях, таких як Android та iOS.

3. Перевірте сумісність із різними розмірами екрана

Є багато різних розмірів екрана, які кінцеві користувачі можуть спробувати використовувати з програмою, тому важливо перевірити інтерфейс користувача для них.

Тестування чутливості інтерфейсу користувача найкраще впровадити на найновіших пристроях, щоб вирішити потенційні проблеми. Крім того, пам’ятайте про тестування як в альбомному, так і в портретному режимах.

4. Перевірте продуктивність і масштабованість

Якщо програма має масштабованість, вона здатна забезпечити чудову продуктивність на різних платформах.

Тестуйте різні рівні навантаження, трафік та інші сценарії кінцевого користувача, щоб оцінити продуктивність і масштабованість програми.

Це можна зробити за допомогою паралельного тестування, яке використовує автоматизоване тестування інтерфейсу користувача, як [автоматизація роботизованих процесів](https://www.zaptest.com/rpa) у кількох середовищах.

5. Перевірте доступність програми

Тестування доступності гарантує, що певні функції, спрямовані на допомогу кінцевим користувачам, працюють належним чином. Тут можна перевірити такі речі, як розмір шрифту, режим читання з екрана та параметри масштабування.

6. Перевірте кольори та текст

Програми мають відображати кольори певним чином, тому важливо перевірити це, перевіривши схеми кольорів.

Це стосується таких речей, як колір гіперпосилання чи інші типи шрифтів. Також корисно перевірити текст на наявність проблем з орфографією, розміром шрифту та вирівнюванням.

7. Оцініть швидкість навігації

Переконайтеся, що інтерфейс програми працює гладко, без збоїв. Такі речі, як екран завантаження для заголовків, є хорошим місцем для пошуку затримок.

**Контрольні запитання**

1. Що таке тестування інтерфейсу користувача?
2. В чому суть UI тестування?
3. Наведіть список необхідних перевірок тестування GUI.
4. Перелічіть інструментальні засоби GUI тестування.
5. **Які основні проблеми тестування інтерфейсу користувача?**