**Лабораторна робота № 12-2 Формування тестів Web-Додатків**

**Мета:** Навчитися формувати сценарії тестування Web-Додатків

**Методичні рекомендації до виконання лабораторної роботи.**

1. Повторіть теоретичні відомості лекцій 20, 21.
2. Оберіть веб – додаток, вкажіть його назву та посилання, визначте необхідні сценарії тестування (тест –кейси).

Тест-кейс

| **№** | **Завдання** | **Послідовність дій** | **Очікуваний результат** |
| --- | --- | --- | --- |
| 1 |  |  |  |
| .. |  |  |  |

Зробіть не менше 3-х сценаріїв

1. 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 – 17.05.2024**

**ТЕОРЕТИЧНІ ВІДОМОСТЕЙ**

**Етапи тестування веб-проєктів:**

Підготовчий етап та вивчення документації.

В даний етап входить аналіз технічного завдання; вивчення кінцевих макетів; тест кейсів; матриці відповідності (для валідації покриття вимог щодо продукту тестами) і складання плану тестування.

**Тестування верстки.**

Візуальна частина:

* невірне відображення блоків, складових інтерфейсу, нестиковки колірної гами;
* тестування локалізованих версій (переклад сайту);
* відповідність макету (шари у PhotoShop);
* у разі зменшення/збільшення масштабів (75-150%) без візуальних недоліків;
* підсвічування полів з помилками;
* перевірка у дозволах (+ прокрутка);

Перевірити можна так: FirefoxMenu -> Інструменти -> Веб-розробник -> Адаптивний дизайн або Resolution Test Plugin у Chrome.

**Доступність і відсутність JS помилок:**

* чи натискаються клікабельні елементи (внутрішні/зовнішні посилання, посилання на електронну пошту, кнопки, іконки);
* під час наведення на клікабельні елементи курсор змінюється, в іншому випадку – ні;
* підказки на незрозумілих клікабельних елементах;
* під час відключення зображень повинні бути підписи невеликим сірим кольором (у Web Developer -> Images -> Replace Image With Alt Attributes);
* працездатність при вимкненому JS. Критичні функції повинні бути доступні без JS (в Web Developer -> Disable -> Disable JS -> All JS)

**Коректна робота, надійна верстка:**

перевірка роботи з даними (введення великої та малої кількості тексту у форму; блоки з контентом міняються місцями (Firebug (HTML -> Edit)));

перевірка роботи стилів (введення тексту з заголовками, з абзацом і без, з картинками).

**404-і запити:**

чи є 404-і помилки (Firefox -> Tools -> Validate links)

**Функціональне тестування.**

Вид тестування, у якому виявляється некоректна/неправильна робота функціоналу програми.

**Необхідними перевірками є:**

* коректність роботи головних функцій сайту;
* перехід за посиланнями;
* перевірка користувальницьких форм (валідація полів, обов’язкові/необов’язкові поля, повідомлення про помилки під час неправильного введення, додавання коментарів у блог, зворотний зв’язок);
* пошук і покупка товару, оформлення замовлення;
* звірка переданого замовником контенту з наявним на сайті;
* перевірка можливої авторизації/реєстрації;
* додавання, видалення та редагування даних користувачів, товарів і замовлень.

**Ad-hock тестування** – імпровізаційне тестування без підготовки.

Допомагає зрозуміти:

* чи зрозумілим є призначення форм;
* чи відзначені обов’язкові поля та чи всі обов’язкові поля відмічені;
* чи вбудована обов’язкова перевірка заповнених форм;
* чи відбувається перевірка правильності введення контактних даних.
* З переваг даного тестування можна виділити:
* достатньо швидке знайомство з системою;
* специфічні несправності;
* масу питань і пропозицій;
* економію часу.

**Негативне тестування**, яке зазвичай називають тестуванням шляху помилок або тестуванням на збій, це процес застосування якомога більшої кількості креативних підходів та перевірки програми на предмет наявності невірних даних. Його призначення полягає в тому, щоб перевірити, чи показуються помилки користувачеві, де вони можуть бути, або більш витончено обробляти неправильні значення. Проводиться для забезпечення стабільності додатків.

**Еквівалентні тести** – це тести, які призводять до одного і того ж результату. Група тестів є класом еквівалентності, за умови дотримання наступних умов:

* всі тести написані для виявлення однієї помилки;
* якщо один з тестів виявить помилку, то інші теж її виявлять;
* зворотне теж вірно.

Еквівалентна область – частина області вхідних або вихідних даних, для яких поведінка компонентів або систем, ґрунтуючись на специфікації, вважаються однаковими.

**Exploratory testing**, також носить назву інтуїтивного тестування, має на увазі одночасне проєктування, виконання тестів і навчання продукту.

**Usability тестування (User Experience).**

Дозволяє перевірити комфортне використання сайту для користувача, наскільки легко знайти необхідну інформацію або виконати бажані дії.

**Навігаційне тестування сайту**. Чи всі сторінки, кнопки та поля на них, зрозумілі під час використання, доступ до головної сторінки та меню з усіх інших сторінок можливий, навігація проста та інтуїтивно зрозуміла.

**Тестування контенту**. Відсутність граматичних/орфографічних помилок, контент інформативний та структурований, зображення та заголовки мають відповідні розміри і розміщені вірно.

**Зручність використання**. Чи зрозуміла структура веб-додатку, яке враження справляє і чи наявні зайві компоненти на сторінках.

**Тестування UI (User Interface).** Відповідність стандартам графічних інтерфейсів й елементів дизайну, правильність локалізованих версій, тестування з різними дозволами, на смартфонах і планшетах.

**Тестування сумісності (конфігураційне тестування).**

Тип нефункціонального тестування програмного забезпечення, що дозволяє перевірити, чи може ПЗ працювати на іншому обладнанні, операційних системах, додатках, мережевих середовищах або мобільних пристроях.

**Кросплатформенне (багатоплатформне) тестування сайту.** Окремі функції можуть мати проблеми з певними операційними системами, тому необхідно перевіряти роботу програми у різних версіях Windows, Unix, Mac, Linux, Solaris і ін.

**Кросбраузерні тестування сайту.** Також коректна робота залежить від типу браузера. Верстка повинна бути кросбраузерною, щоб забезпечити однакову візуальну частину, доступність, функціональність і дизайн у всіх браузерах. Необхідно перевіряти масштабованість, розширюваність, рамки для елементів у фокусі, відсутність JS помилок (лівий нижній кут сторінки). Перевіряти роботу необхідно у таких браузерах, як: Internet Explorer, Firefox, Chrome, Safari, Opera, Edge різних версій.

**Перегляд на мобільних пристроях.** Незважаючи на перевірку роботи веб-додатків у різних роздільних здатностях на комп’ютері, найчастіше помилки на мобільних пристроях залишаються непоміченими. Отже, наполегливо рекомендується перевіряти коректне відображення та роботу вашого веб-додатку на мобільних пристроях різних операційних пристроях, а також на планшетах.

**Тестування БД.** Необхідно перевірити правильність встановлення зв’язку з сервером, перевірити сумісність сервера з ПО, апаратними засобами, базою даних і мережею. Також слід перевірити, що відбувається під час переривання будь-якої дії, під час наступного з’єднання з сервером під час виконання операцій.

**Тестування продуктивності.**

Методика нефункціонального тестування, для вимірювання таких параметрів системи як чутливість та стабільність, за різних навантажень. Дозволяє досліджувати швидкодію сайту та можливості масштабованості додатку, наприклад, під час додавання нових користувачів. Проводиться з метою з’ясувати, яке навантаження сайт здатний витримати. Тестування продуктивності вимірює атрибути якості системи, такі як масштабованість, надійність і використання ресурсів.

**Навантажувальне тестування** – це метод тестування продуктивності, у якому реакція системи вимірюється в різних умовах навантаження. Відповідає за реакцію веб-додатка у разі збільшення робочого навантаження. Навантажувальні випробування проводяться для нормальних і пікових навантажень (одночасна купівля товару або авторизація на сайті великої кількості користувачів).

Підхід навантажувального тестування:

* Оцінити критерії прийнятності продуктивності
* Визначити критичні сценарії
* Модель робочого навантаження
* Визначити цільові рівні навантаження
* Дизайн тестів
* Виконати тести
* Проаналізуйте результати
* Завдання навантажувального тестування: час відгуку, пропускна здатність, утилізація ресурсів, максимально призначене для користувача навантаження, бізнес-метрики.

**Стрес-тестування (Stress Testing)** перевіряє систему на її стійкість і обробку помилок в умовах надзвичайно високого навантаження (оцінює як система працює в екстремальних умовах, поза обмеженнями та лімітами). Стрес-тестування проводиться, щоб переконатися, що система не буде аварійно завершувати роботу у критичних ситуаціях.

**Тестування стабільності/надійності (Stability/Reliability Testing)** – тип тестування програмного забезпечення, який перевіряє, чи може програмне забезпечення працювати безвідмовно протягом певного періоду часу у вказаному середовищі.

**Об’ємне тестування (Volume Testing)** – тип тестування програмного забезпечення, яке проводиться для аналізу продуктивності системи за рахунок збільшення обсягу даних у базі даних.

**Тестування паралелізму (Parallel Testing)** – тип тестування програмного забезпечення, який перевіряє кілька додатків або підкомпонентів однієї програми одночасно, щоб скоротити час тестування. Під час паралельного тестування тестувальник запускає дві різні версії програмного забезпечення одночасно з одним і тим самим вводом. Мета полягає в тому, щоб з’ясувати, чи ведуть себе колишня система і нова система однаково або по-різному.

**Тестування безпеки.**

Спрямоване на оптимізацію безпеки системи під час проєктування, розробки, використання та обслуговування програмних систем і їх інтеграції з критично важливими для безпеки апаратними системами у виробничому середовищі.

***Аспекти безпеки*** *програмного забезпечення:*

Функціональне програмне забезпечення не повинно створювати небезпек (наприклад: управління сучасним літаком НЕ повинне спрямовувати його в океан).

Системи моніторингу повинні працювати без збоїв (наприклад: резервний комп’ютер повинен запускатися автоматично у разі збою на основному).

***Цілі*** *в тестуванні* ***безпеки****:*

* у складних системах, де задіяні багато взаємодій, критично важливі для безпеки функції повинні бути ідентифіковані та ретельно проаналізовані;
* помилки визначені та усунені;
* кількість критичних помилок підтримується на низькому рівні, щоб уникнути непрацездатності системи;
* атрибути безпеки повинні розглядатися як частина всіх рівнів тестування ПЗ.

Додаткову інформацію з безпеки програм можна дізнатися тут: CHECK, ISACA, NIST Guideline, OSSTMM, OWASP Guide.

**Принципи безпеки:**

Конфіденційність (обмеження або надання доступу до інформації).

Цілісність (можливість відновити дані в повному обсязі у разі їх пошкодження; доступ до зміни інформації тільки певної категорії користувачів).

Доступність (ієрархія рівнів доступу та їх чітке дотримання).

*Обробка помилок та регресійне тестування.*

Після завершення розробки веб-додатку слід провести оцінку й аналіз виявлених помилок для подальшого запобігання їх повтору. А також виконати регресійне тестування.

**Регресійне тестування**

Використовує техніку тестування чорного ящика (повторне виконання тестів), на які впливають зміни коду. Ці тести повинні виконуватися якомога частіше протягом всього ЖЦПЗ у разі змін коду для виправлення дефектів або для покращення роботи веб-додатків.

**Практичні поради:**

* Перед тим, як розпочати тестування, необхідно обговорити всі важливі деталі з командою (BA, PM, розробники).
* Використовувати широкий підхід з використанням технік тест-аналізу та набору методик тест-дизайну.
* Визначити види тестування, які необхідно провести.
* Визначити цілі та ключових користувачів веб-додатків.
* Списки пристроїв, ОС та браузерів, на яких необхідно провести тестування.
* Доступ для різних ролей відвідувачів.
* Необхідність складання та передачі документації.

Хочемо нагадати, що робочий процес – це не рутина, а творчий процес, який визначає широту польоту вашої думки. Ставтеся до вашої роботи як до нового челенджу, і ви напевне почнете отримувати не тільки задоволення, але й матимете натхнення та бажання розвиватися. Завдання тестувальника дуже багатогранні: йому необхідно зрозуміти завдання веб-додатку, зрозуміти як він повинен працювати, які завдання вирішувати, яку користь приносити користувачам, а потім перевірити ще раз всі ПЗ кілька разів, щоб випустити проєкт в світ. Їм потрібно бути зібраними та сміливими, щоб випускати проєкти, якими зможе пишатися вся команда

**Тест-кейс (Test Case)**

Тест-кейс– це тестовий артефакт, суть якого полягає у виконанні певної кількості дій і/або умов, необхідних для перевірки певної функціональності програмної системи, що розробляється. Структура даного артефакту полягає у визначенні:

- що треба зробити (Action) ;

– який очікуваний результат (Expected result);

– який отримано фактичний результат (Test result).

Безпосередньо сам тестовий випадок складається з 3 частин:

1. ***PreConditions*** (передумови) – або список кроків, які призводять систему в стан, придатний для тестування, або список перевірок умов того, що система вже знаходитися в необхідному стані.

2. ***Test Case Description*** (опис тестового випадку) – список дій, за допомогою яких здійснюється основна перевірка функціоналу (після якої і звіряється фактичний результат з очікуваним).

3. ***PostConditions*** (післяумови) – список дій, які повертають систему в початковий стан.

**Тест-кейси** описують послідовні покрокові операції перевірки функціоналу програми або веб-сайту. Це мінімальні елементарні операції звірки для кожної функції або елемента додатки.