ТЕМА 11. ОСОБЛИВОСТІ ТЕСТУВАННЯ WEB-ДОДАТКІВ

1. Визначення та основні складові веб-додатку.

Веб-додатки – сфера, яка динамічно розвивається. Не усі підходи та методи, які застосовуються для тестування класичних додатків можуть бути використанні при тестуванні веб-додатків.

Веб-додаток - це клієнт-серверний додаток, в якому клієнтом є браузер, а сервером - веб-сервер, що вже по суті є відмінними програмами, які необхідно тестувати як окремо, так і разом.

На рисунку 1 наведена структура веб-додатку.

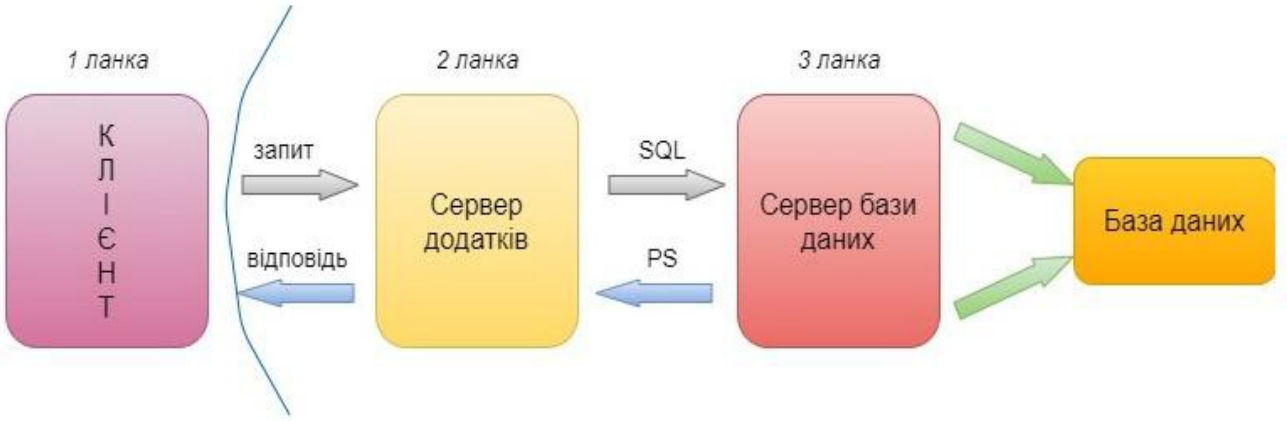

Рисунок 1 – Структура веб-додатку

Веб-додаток представлений наступними складовими («сторонами»):

1. ***Клієнт.*** Як правило клієнт – це браузер, але зустрічаються і виключення (в тих випадках, коли один веб-сервер виконує запит до іншого, роль клієнта виконує перший сервер). В класичній ситуації (коли роль клієнта виконує браузер) для того, щоб користувач побачив графічний інтерфейс додатку у вікні браузера, останній повинен обробити отриману відповідь веб-сервера, в якій буде міститись інформація, реалізована із застосуванням HTML, CSS, JS (найбільш застосовувані технології). Саме ці технології «дають зрозуміти» браузеру, як саме необхідно «відтворити» все, що він отримав у відповіді.

2. ***Сервер.*** Веб-сервер – це сервер, який приймає HTTP-запити від клієнтів та

віддає їм HTTP-відповіді. Щоб уникнути можливої плутанини, відмітимо, що веб-сервером називають як програмне забезпечення, що виконує функції веб- сервера, так і безпосередньо комп'ютер, на якому це програмне забезпечення працює. Найбільш розповсюдженими видами програмного забезпечення веб- серверів є Apache, IIS та NGINX. На веб-сервері функціонує тестований додаток, який може бути реалізований із застосуванням самих різноманітних мов програмування: PHP, Python, Ruby, Java, Perl та інші.

3. ***База даних.*** В класичній теорії мова іде про дві «сторони» веб-додатку, однак, якщо уважно подивитись на весь процес роботи додатку, ми можемо відповісти, що в алгоритмі роботи веб невидимо, але доволі активно приймає участь ще одна «сторона» - база даних. Фактично вона не є частиною веб- сервера, але більшість додатків просто не можуть виконувати всі покладені на них функції без неї, так як саме в базі даних зберігається вся динамічна інформація додатку (звітні, користувацькі дані та інші).

База даних – доволі широке поняття, котре використовується не лише в сфері інформаційних технологій. В даному контексті – це інформаційна модель, котра дозволяє впорядковано зберігати дані про об'єкт чи групу об'єктів, які володіють набором властивостей, котрі можна категоризувати. База даних функціонує під керівництвом так званих систем керування базами даних (СУБД). Найбільш популярними СУБД є MySQL, MS SQL Server, PostgreSQL, Oracle (все веб-серверні).

Також існують вбудовані та файл-серверні СУБД. Для загального розвитку відмічу лише одну найбільш популярні вбудовану СУБД – SQLite, котра використовуються в деяких браузерах, Android API, Skype та інших відомих додатках. Взаємодія з перерахованими СУБД основана на спеціальній мові структурованих запитів – SQL.

Майже всі сучасні програми орієнтовані на роботу з мережею. Збереження даних веб-додатків здійснюється зазвичай на сервері, обмін інформацією відбувається по мережі. Коли ми бачимо помилку в мережевому середовищі, то досить часто важко впевнено сказати, де саме вона відбувалась, і тому режим роботи чи повідомлення про помилку, яке ми отримуємо, може бути результатом помилок, які виникли в різних частинах мережі.

Маючи багато спільного із тестуванням класичного додатку, тестування веб-орієнтованих додатків має свої особливості, пов'язані в першу чергу із середовищем

функціонування. Маючи компонентні, структурні та технологічні особливості, веб-додаткам притаманні особливості режимів роботи, інсталяції, запуску, зупинки та видалення, а також формування інтерфейсів. Працюючи завжди з мережею та великою кількістю користувачів, веб-додатки мають на увазі під собою різноманітні права доступу для різних користувачів.

Логіка веб-додатку розподілена між сервером та клієнтом, зберігання даних здійснюється на сервері, обмін інформацією відбувається по мережі.

Однією із переваг підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформними сервісами.

Те, що веб-розробки будуть продовжувати збільшувати темпи свого розвитку, підтверджується і набираючим силу «мейнстрімом»: все

«переміщується» в хмару. Хмарні технології стають реальністю сучасного Інтернету: навіть колись звичні нам десктопні Word та Excel сьогодні представлені у вигляді веб-альтернатив від Microsoft. Виходячи із сказаного, можна стверджувати, що необхідність в деяких інженерах по забезпеченню якості, які спеціалізуються на веб-продуктах, буде лише збільшуватись.

2. Особливості тестування веб-додатків

1. Технологічні відмінності. Класичний додаток працює із використанням однієї чи сімейства споріднених технологій.

2. Структурні відмінності. Класичний додаток «монолітний». Складається з одного чи невеликої кількості модулів. Не використовує сервери бази даних, веб-сервери та інше. Веб-додаток – «багатокомпонентний». Складається з великої кількості модулів. Обов'язково використовує сервери бази даних, веб-сервери та сервери додатків.

3. ***Відмінності режимів роботи.*** Класичний додаток працює в *режимі реального часу,* тобто відомо про дії користувача одразу ж, як тільки вони були виконані. Веб-додаток працює в режимі «запит-відповідь», тобто відомо про деякий набір дій тільки після запиту на сервер (див.рис.2).

4. ***Відмінності формування інтерфейсу.*** Класичний додаток використовує для формування інтерфейсу користувача відносно усталені та стандартизовані технології. Веб-додатки використовують для формування користувацького інтерфейсу технології, які стрімко розвиваються, більшість яких конкурують між собою.

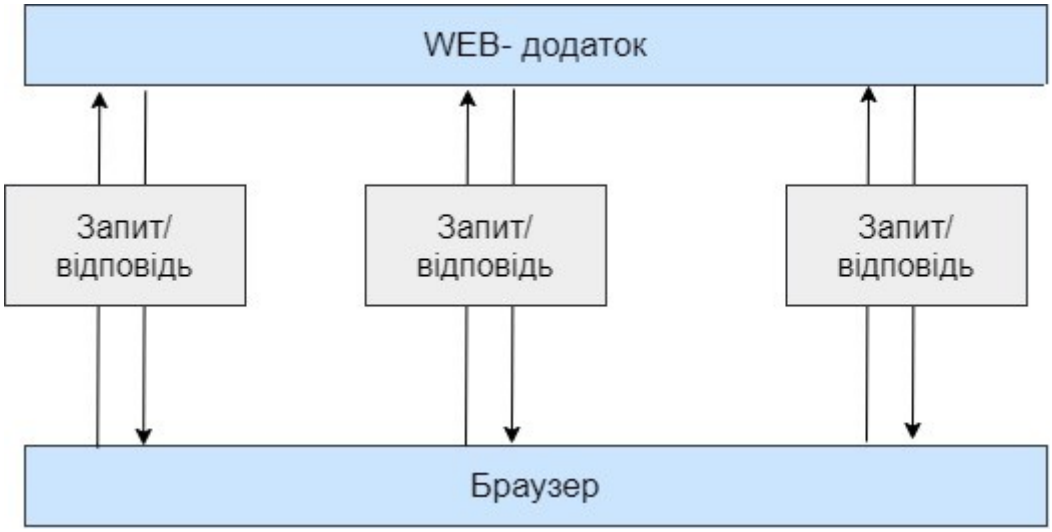

Рисунок 2 – Режим роботи веб-додатку

5. Відмінності роботи з мережею. Класичний додаток практично не використовує мережеві канали передачі даних. Веб-додаток активно використовує мережеві канали передачі даних.

6. Відмінності запуску та зупинки. Класичний додаток запускається та зупиняється рідко. Веб-додаток запускається та зупиняється по факту поступлення кожного запиту, тобто дуже часто.

7. Різниця в кількості користувачів. Класичний додаток: кількість користувачів, які одночасно використовують додаток, контрольована, обмежена та легко прогнозована. Веб-додаток: кількість користувачів, які одночасно використовують додаток, важкопрогнозована і може стрибкоподібно змінюватись в широких діапазонах.

8. Особливості збоїв та відмов. Класичний додаток: вихід із ладу тих чи інших компонент одразу стає очевидним. Веб-додаток: вихід з ладу деяких компонент дає непередбачуваний вплив на працездатність додатку вцілому.

9. Відмінності в інсталяції. Класичний додаток – процес інсталяції стандартизований та максимально орієнтований на широку аудиторію користувачів. Не потребує специфічних знань. Додавання компонент додатку виконується стандартним способом із використанням одного і того ж інсталятора. Веб-додаток – процес інсталяції часто недоступний кінцевому користувачеві. Інсталяція потребує специфічних знань. Процес зміни компоненти додатку не передбачає або потребує кваліфікації користувачів, інсталятор відсутній.

10. ***Відмінності в деінсталяції.*** Класичний додаток: процес деінсталяції стандартизований та виконується автоматично чи напівавтоматично. Веб-додаток: процес деінсталяції потребує специфічних знань для втручання адміністратора і часто поєднаний із зміною коду середовища функціонування додатку, бази даних, налаштування системної операційної системи.

11. Особливості середовища функціонування. Класичний додаток: середовище функціонування стандартизоване та не сильно впливає на функціонування додатку. Веб-додаток: середовище функціонування дуже різноманітне і може проявляти серйозний вплив на дієздатність як серверної, так і клієнтської частини.

3. Особливості архітектури: «під прицілом» клієнт

Ми зазначали клієнта як першу складову архітектури. В класичній ситуації клієнт представлений браузером, а тому питання тестування кросбраузерності досить актуальний. Ми також розглянемо тестування форм, які заповнюються та тексту, як основного джерела інформації, що отримується від клієнта.

Кросбраузерність: різноманіття клієнтів

Що ми розуміємо під тестуванням на кросбраузерність? Це перевірка на правильність (відповідність вимогам і стандартам) відображення та функціонування веб-додатку в різних браузерах та на різних операційних системах. В сучасному світі стандартизація приймає глобальні масштаби, а тому більшість популярних браузерів однаково обробляють код. При цьому необхідність в кросбраузерному тестуванні не зникає, так як далеко не всі проблеми вирішуються стандартизацією.

Перед початком робіт рекомендовано взнавати у відповідальних людей необхідний для тестованого програмного забезпечення набір браузерів. Можна самим зібрати статистику по цільовій аудиторії вашого продукту та обмежити кросбраузерне тестування набором найбільш популярних в цій аудиторії браузерів. В умовах регресійного тестування, коли обмежені часові та людські ресурси, також можна застосовувати практику розподілу браузерів: закріпляючи за кожним тестувальником певний браузер, ви зможете «покрити» більший відсоток кросбраузерних дефектів. Цей метод має і свої недоліки: він найбільш результатиний у великій команді тестувальників, але стає практично незастосовуваним при наявності одного тестувальника.

Що необхідно перевіряти при кросбраузерному тестуванні:

- функціональні можливості продукту, які реалізуються на стороні клієнта;
- правильність відображення елементів графіки;
- шрифти та розміри текстових символів;
- доступність та функціональність різноманітних форм, включаючи їх інтерактивність.

Тестування потребують всі основні (серед користувацької аудиторії) браузери, але особливу увагу необхідно приділити Internet Explorer, якщо він входить в їх кількість. Саме в ньому дуже часто виникають проблеми, котрі відсутні в інших браузерах: так як в Internet Explorer додатково рекомендується звертати особливу увагу на масштабованість, фокус полів та роботу JS.

Окремо варто не забувати про різного роду валідаторів верстки. Навіть якщо у вас достатньо знань, щоб оцінити відповідність верстки стандартам, можна використати для цього автоматичні засоби і, проаналізувавши результат, вказати розробникам на самі серйозні «огріхи». Не варто забувати, що інколи валідатори звертають увагу на самі «дрібні дрібниці», котрі ніхто і ніколи виправляти не буде. Якщо ви і заносите баг-репорти на подібні зауваження, то зручніше буде зібрати їх в окремий документ та прикріпити до репорта. До такого роду «дрібниць» можна віднести всі можливі рекомендації, котрі не мають свого впливу на відображення та функціонування контенту.

Веб-форми на стороні клієнта

Однією із важливих складових інтернет-додатків є форми заповнення, взаємодію із якими користувач здійснює за допомогою все того ж клієнта. Однак, дані форми дуже часто слугують джерелом дефектів, котрі, розмістившись у «продакшині», можуть принести великі фінансові і репутаційні збитки компанії.

Як ж не пропустити дефекти у формах на продакшен? Розглянемо декілька простих кроків:

1. Ретельно перевіряємо обов’язковість заповнення полів та наявність відповідного маркування у них.
2. Заповнивши та надіславши форму, впевнюємось у тому, що з даними відбувається саме те, що заплановано. Якщо дані повинні бути внесені в базу даних, перевіряємо коректність внесення.

3. Використовуємо чіт-листи для тестування форм.

Чіт-лист (Cheat-sheet) – список перевірок, що повторюються. Чіт-листи складаються із ціллю їх наступного багаторазового використання. У зв'язку з цим такі списки складаються по відношенню до розповсюджених та часто зустрічних складових програмного забезпечення, з яким належить працювати неодноразово не лише в поточному проекті, але і в наступних.

Кожний окремий спеціаліст розширює перелік перевірок у своїх чіт- листах із досвідом. На сьогодні багато автоматизованих систем управління тест-кейсами також містять і функціонал підтримки списків чіт-листів, що в значній мірі спрощує їх створення, наступні підтримки та практичне застосування.

4. Перевіряємо чи виводяться зрозумілі користувачу інформаційні повідомлення про необхідність заповнення пустих полів після спроби надіслати форму.

5. Звертаємо увагу на реалізацію екранування символів в полях форм, які є потенційним джерелом вразливості для додатку та користувачів. Екранування повинно здійснюватись на рівні не лише клієнта, але і сервера, вимкнути який в клієнті досить просто (наприклад за допомогою спеціальних плагінів, котрі знімають всі можливі обмеження в декілька кліків).

6. Впевнюємось, що після заповнення форми, користувачу приходить підтверджувальний лист з вказуванням відповідного відправника, а саме тіло листа відповідає вимогам.

7. Використовуємо допоміжні спеціальні інструменти для тестування форм, наприклад Web Developer Toolbar.

Текст, як основне джерело інформації при роботі через клієнт

Основною цінністю мережі Інтернет є те, що вона являється практично безмежним джерелом інформації. Частина цієї інформації представлена у вигляді текстів, з якими, знову ж таки, користувач взаємодіє за допомогою клієнту. Більшість веб-ресурсів в тому чи іншому об'ємі потребують перевірки текстів на предмет відсутності граматичних та друкарських помилок.

Звісно, значимість цього тестування не така велика порівняно з функціональним напрямком, але нехтувати ним не потрібно. На практиці бувають серйозні друкарські помилки: на продакшині одного з великих інтернет-гіпермаркетів була допущена

помилка, яка змінила назву міста на лайливе слово, яке було видно не лише мешканцям цього міста. Так, часу на вичитування всього тексту буває замало, тоді в таких випадках на допомогу приходять «SpellCheker» (програми для перевірки орфографії, онлайн або у вигляді плагінів для браузерів).

4. Особливості архітектури: «під прицілом» сервер

Тестування частини веб-додатку, розміщеної на веб-сервері, можна провести оминувши графічний(клієнтський) інтерфейс, однак це потребує від спеціаліста визначеного рівня знань та навиків технічного характеру, а також застосування додаткових інструментів. Розглянемо веб-сервер з точки зору тестування навантаження та інсталяції.

Інсталяція на веб-сервер

Отже, перед тим як почати тестувати, ми повинні встановити веб-додаток на веб-сервер. Відповідно тут є схожість з тестуванням десктоп додатків, але існує і відмінність в нюансах, котрі необхідно врахувати та протестувати, особливо якщо це стосується програмного забезпечення, розповсюджуваного для локальної інсталяції на веб-сервери користувачів.

В чому нюанси:

1. Більша частина веб-додатків потребує для інсталяції специфічних знань в адмініструванні операційних систем. Спробуйте встановити додаток на декількох веб-серверах. В оптимальному випадку це будуть самі популярні технології серед ваших користувачів, для встановлення списку яких необхідно попереднє дослідження.

2. Інсталюючи веб-додаток, звертайте увагу на те, чи дійсно додаток встановлюється у вказану вами директорію, базу даних, використовує обраний вами префікс та дотримується інших конфігураційних моментів.

3. Переконайтесь, що додаток можна встановити як localhost, так і віддалено.

4. Якщо процес інсталяції є ітерактивний, і кожний вибір користувача на певному кроці впливає на наступні дії, то необхідно буде пройти всі гілки, так як саме інсталяція є першим кроком користувача при роботі з вашим додатком.

5. Не забувайте про негативні тести: перевірка в умовах нестачі ресурсів (місця на дискові, оперативної пам'яті) чи привілегій, переривання процесу встановлення під

час активної його фази (наприклад, надсилаючи сервер на перезаванатаження).

Тестування навантаження

Тестування навантаження імітує роботу з додатком певної кількості користувачів. Цей вид тестування здійснюється за допомогою спеціальних інструментів (наприклад jMeter), головна ціль яких – визначити профілі навантажень та штучно створити для них навантаження, які визначають граничні можливості додатку (чи сервера) в умовах роботи з ним того чи іншого користувача.

Отримана інформація підлягає детальному аналізу з наступним виявленням «вузьких місць» та граничних програмних чи апаратних можливостей, котрі в подальшому використовуються для забезпечення стабільності веб-сервера та самого додатку, що на ньому працює.

Наведемо приклад з практики функціонування великого комерційного продукту, котрий довший час працював з різноманітними типами договорів. Одного разу в реліз випустили черговий особливо очікуваний тип договору, і на наступний день система повністю перестала працювати, а служба підтримки була завалена величезною кількістю звернень. Всьому виною став прорахунок в тестуванні: команда перевіряла одночасну роботу з десятками тисяч договорів, але ніхто не зміг передбачити, що на практиці мова піде про сотні тисяч, а іноді і про мільйони договорів. Тестування навантаження дозволяє виявити потенційні проблеми такого характеру ще на етапі тестування.

5. Особливості архітектури: «під прицілом» база даних

Ще однією раніше розглянутою складовою веб-додатків є база даних, в якій додаток зберігає всю необхідну інформацію. Для того, щоб база даних служила гідним сховищем інформації для вашого додатку, при тестуванні необхідно звертати увагу на такі основні моменти:

1. Інформація, внесена через інтерфейс, повинна бути збережена в базі даних в незмінному (первісному) вигляді.
2. Збережена в базі даних інформація повинна відображатися в будь-якій частині програми однаково (якщо іншого не вимагає бізнес-логіка програми).
3. Назви таблиць і структура бази даних повинні відповідати проектній документації.

4. Потрібно стежити за тим, щоб запити не оброблялись надто довго, а кількість з'єднань було достатнім. Моніторинг стану бази даних - один із важливих моментів тестування.

5. Варто враховувати, що видалення запису з бази даних не завжди супроводжується повним видаленням сутності. Іноді використовується так зване «псевдовидалення», і потрібно перевірити, чи правильно воно виконується.

6. Порожню базу даних на тестовому стенді рекомендується або заповнювати згенерованими випадковими даними, або знімати дамп з продакшена і після обфускації (обфускація – це процес, в результаті якого код програми чи інших даних набуває вигляду, важкого для аналізу) даних

«заливати» їх в тестовану базу даних. Іноді кваліфікація тестувальника (або інша причина) не дозволяє виконати цей процес самостійно: в такому випадку рекомендується звернутися за допомогою до фахівців інфраструктури або розробникам.

6. Запити

Всі складові веб-додатів повинні взаємодіяти між собою і відбувається це за допомогою HTTP. Без HTTP наша багатостороння система не функціонувала б взагалі, так як HTTP – це протокол передачі даних, який займає одне із основних місць в нашій архітектурі.

Взаємодія здійснюється через повідомлення: на надісланий запит від клієнта повинна прийти відповідь сервера. Класичний запит/відповідь складається з 3 складових:

- стартова стрічка;
- заголовок;
- текст повідомлення.

При роботі з відповідями спеціаліст по тестуванню в першу чергу повинен звертати увагу на методи та коди станів, котрі присутні в стартовій стрічці.

Найпопулярнішими методами є GET, HEAD і POST:

1. Метод GET. Використовується для запиту вмісту, розміщеного на сервері (наприклад, GET/path/resource?param1=value1¶m2 = value2 HTTP/1.1).

2. Метод HEAD. За своєю суттю не відрізняється від вищезгаданого методу, однак відповідь сервера на такий запит позбавлений «тіла», а практичне застосування

орієнтоване на полегшене використання з метою отримання мінімальної інформації про сервер/продукт або його статус.

3. Метод POST. Даний метод використовується для передачі даних на сервер, однак його основа «ховається» в тіло, що відрізняє його від GET. Під час публікації цієї статті, наприклад весь текст поміщений в тіло POST-запиту; після обробки його сервером на сайт буде додано статтю.

Існують і інші методи: PUT, DELETE, CONNECT, TRACE, PATCH тощо. Те, що складові веб-додатку взаємодіють між собою за допомогою HTTP,

є гарною новиною для фахівця з тестування, так як це дозволяє не просто відстежувати спілкування, вникаючи в логіку роботи і звіряючи її з технічними вимогами, але і впливати безпосередньо на додаток, прикладаючи свою руку до відправлених запитів

Класичними додатками, які можна використовувати для генерації запитів, є Fiddler або Postman. Використовуючи Fiddler, можна з легкістю відстежувати всі запити від клієнта і відповіді, переглядати їх деталі, а також вносити свої зміни і відправляти модифіковані запити на сервер, оцінюючи поведінку системи в такому випадку.

На що звертати увагу при запитах?

1. Чи правильний метод використовується для того чи іншого запиту? Якщо ви відправляєте повідомлення, а на сервер йде тільки GET-запит, то щось тут явно не так.

2. Здавалося б, клік по одній і тій же функціональній клавіші генерує один і той же запит. Але є прецеденти, коли клік по кнопці приводив до ситуації, в якій JavaScript переписував запит кнопки, розміщеної поруч, таким чином змінюючи її призначення.

3. Вникайте у запити, які надсилаються. У них досить легко розібратися, особливо якщо це GET - вони логічно зрозумілі навіть пересічному користувачеві. Аналіз запитів - це можливість виявити прихований дефект набагато швидше, ніж здійснюючи його пошук в інтерфейсі.

4. Моніторте трафік на предмет запитів на інші (не ваші) сервера. Приклад з життя: фронтенд сайту робив фріланс-розробник, по завершенню роботи якого взялася за тестування фірма. Тестувальник мав звичку моніторити весь трафік тестованої програми: це дозволило виявити, що вищезгаданий розробник безсоромно сховав у «фронт» сайту запити, які працювали на благо його особистого інтернет-магазину.

5. Моніторте трафік, уважно стежте за кодами станів. Трішки докладніше зупинимося на цьому пункті.

Коди станів очима адепта якості

Кожна відповідь на будь-який запит несе в собі масу корисної для розробників і фахівців з тестування програмного забезпечення інформації. Одним з джерел такої інформації може стати код стану: за цим кодом клієнт дізнається про результати його запиту і визначає, які дії йому робити далі. Набір кодів стану є стандартом, він описаний у відповідних документах RFC. Кожен раз, коли ви створюєте баг-репорт про непрацюючі посилання, інші елементи веб-сторінки або системні помилки, обов'язково вказуйте відповіді сервера: вони заощадять час розробнику при визначенні причин дефекту.

Всі коди можна поділити на групи (соті, двохсоті, трьохсот, чотирьохсоті і п'ятисоті) кожна група-«сотня» несе свій тип інформації (див.табл.1).

Таблиця 1. – Коди станів

Код	Тип інформації коду	Призначення
1xx	Інформаційні	Інформують про передачу повідомлення. Самі повідомлення від сервера містять тільки стартову стрічку відповіді і, якщо необхідно, декілька специфічних для відповіді полів заголовку.
2xx	Повідомляючі про успішні операції	Інформування про успішну обробку запиту клієнта. Самим класичним та поширеним є «200 OK»
3xx	Перенаправляючі	Повідомляє, що для успішного виконання операції необхідно зробити інший запит. З даного типу п’ять кодів: 301,302,303,305 та 307 відносяться безпосередньо до перенаправлення. Адреса, за якою клієнту варто проводити запит, сервер вказує в заголовку Location

4xx	Помилка на стороні клієнта (запит)	Вказує на помилки на стороні клієнта. При використані всіх методів, крім HEAD,сервер повертає пояснення причини
5xx	Помилка на стороні сервера (відповідь)	Говорить про помилки виконання операції за виною веб-сервера

На практиці, використовуючи при тестуванні спеціальні додатки (той самий Fiddler), ви без зусиль зможете впорядкувати свої запити і відповіді за кодом стану і відібрати, наприклад, всі 400-і і 500-і з подальшим їх аналізом.

Таким чином дуже швидко «відловлюються» дефекти з «відваленими» стилями, скриптами, файлами, функціями додатку тощо.

7. Відмінність веб-додатку від десктопного

Багатокористувацька сутність веб-додатків.

Широта аудиторії додатків накладає свій відбиток на специфіку роботи.

1. Один додаток одночасно може використовуватися величезною кількістю людей. Ми вже розглядали питання навантажувального тестування, але також слід звернути увагу на те, що в число користувачів можуть входити представники різних культур, мов і релігій. Нам необхідно пам'ятати про це, особливо якщо мова йде про тестування міжнародного додатка.

2. Кожен користувач може мати свої рівні доступу. В ідеальному варіанті тестувальник створює для себе матрицю рівнів доступу і тестує кожен доступ окремо (див.рис. 3).

3. Користувачі з одним рівнем доступу можуть звертатися до одних і тих же сутностей, що призводить до конкурентного доступу. Тестується це досить просто. Для прикладу розглянемо систему, що має справу з договорами, які можна створювати, публікувати, редагувати, анулювати. Алгоритм роботи такий: під декількома вікнами в режимі інкогніто авторизуємося в додатку під користувачами з різними рівнями доступу; далі обрану для тесту сутність відкриваємо на редагування, а під другим обліковим записом цю ж сутність пробуємо перевести в статус «Анульовано» - на цьому етапі

повинен спрацювати контроль на конкурентний доступ. Операція анулювання блокується, а користувачу видається повідомлення про те, що сутність редагується іншим користувачем (поведінка і пріоритет дій визначається відповідно до вимог і особливостей продукту, але логіка не змінюється).

Функція/Роль	Роль 1	Роль 2	Роль 3	Роль 4	Роль 5	Роль 6
Облікові записи						
Авторизація	X	X	X	X	X	X
Вихід з системи	X	X	X	X	X	X
Налаштування прав доступу						X
Фільтр						
Застосування глобального фільтру до контенту	X	X	X	X	X	X
"Горячі" клавіші						
Використання горячих клавіш	X	X	X	X	X	X
Пошук						
Пошук за об'єктами та сторінками, які доступні для перегляду поточному користувачу	X	X	X	X	X	X
Сценарії						
Створення/редагування/ видалення сценарію		X		X	X	X
Прив'язка сценарію до інших сутностей системи		X		X	X	X
Перегляд сценарію	X	X	X	X	X	X
Пошук за сценаріями	X	X	X	X	X	X
Перегляд списку сценаріїв	X	X	X	X	X	X
Сортування сценаріїв	X	X	X	X	X	X

Рисунок 3 – Приклад матриці рівнів доступу

4. Широта аудиторії говорить про те, що за монітором може перебувати людина, що має злий намір щодо вашого програмного забезпечення.

Мережеві пристрасті: веб-додаток в різних умовах передачі даних

Веб-додатки активно використовують мережу, і це є джерелом можливих проблем. Такою, наприклад, є використання додатку в умовах низької швидкості передачі даних (в браузер Google Chrome, наприклад, вбудована функція Throttling, яка дозволяє сильно занижувати швидкість передачі даних), в умовах втрати пакетів або при відключенні мережі під час активної фази роботи програми (спосіб імітації: спочатку робимо швидкість передачі даних за допомогою Throttling мінімальною, а потім перериваємо мережеве з'єднання під час обробки запиту).

У будь-якому з описаних вище випадків програма має працювати коректно. При «падінні» запиту (time out) або іншої проблеми ми повинні, перезавантаживши сторінку, знову отримати повністю працюючий веб-додаток без будь-якого натяку на щойно пережиту «шкоду». Чи для всіх функцій програми необхідні подібні тести? Ні в якому разі! В майбутньому можете орієнтуватися на свій досвід, а на перших етапах в цих питаннях краще проконсультуватися з розробниками.

Тестування безпеки веб-додатків

Тестування безпеки - окремий напрямок тестування, яке вимагає від фахівця фундаментальних знань технічного характеру і хорошої профільної кваліфікації. Зазначимо ряд загальних моментів, які можуть допомогти будь- якому тестувальнику знаходити класичні уразливості, не допускаючи їх вихід на продакшен. Питання безпеки додатків регламентуються OWASP Guide, CHECK, ISACA, NIST Guideline, OSSTMM.

Існує ряд принципів безпеки, до яких відносяться конфіденційність, цілісність і доступність:

1) Конфіденційність - обмеження доступу до тієї чи іншої інформації для певної категорії користувачів (або навпаки надання доступу лише обмеженій категорії).

2) Цілісність включає в себе:

а) можливість відновити дані в повному обсязі при їх пошкодженні; б) доступ на зміну інформації тільки певної категорії користувачів.

3) Доступність - ієрархія рівнів доступу і чітке їх дотримання.

Перерахуємо класичні уразливості сучасних веб-додатків:

1. XSS - генерація на сторінці продукту скриптів, які становлять небезпеку для користувачів продукту;

2. XSRF - вразливість, при якій користувач переходить з довіреної сторінки на шкідливу, де крадуться цінні для користувача дані;

3. Code injection (PHP, SQL) - ін'єкція частини виконавчого коду, яка робить можливим отримати несанкціонований доступ до програмного коду або бази даних і вносити в них зміни;

4. authorization bypass - це вид уразливості, при якому можна отримати несанкціонований доступ до облікового запису або документів іншого користувача;

5. переповнення буфера - явище, якого можна досягти у шкідливих цілях, по

своїй суті являє використання місця для запису даних далеко за межами виділеного буфера пам'яті.

При тестуванні рекомендовано використовувати чіт-листи вразливостей XSS Filter Evasion Cheat Sheet і MySQL SQL Injection Cheat Sheet.

8. Практичні поради

З чого починати тестувати?

Закцентувати вашу увагу на необхідності узгодити всі ключові аспекти з відповідальними особами (аналітиками, проектними менеджерами, розробниками) ще до початку тестування. Необхідно заздалегідь з'ясувати наступні моменти:

- мета тестування (з їхньої точки зору);
- види тестування, які необхідно провести;
- специфіку програми та її цільову аудиторію;
- перелік пристроїв, на яких необхідно проводити тестування; список операційних систем і браузерів для тестування;
- роздільну здатність екранів на яких необхідно перевірити додаток;
- чи передбачені вимоги до різного роду форм, чи є стайл-гайд, актуальне технічне завдання;
- необхідність надання конкретної документації за результатами тестування (звіт, чек-листи, тест-кейси тощо).

Отримавши відповіді на ці питання, про які часто забувають новачки, ви заощадите свій час. Далі переходите до безпосередньої підготовки оточення і формування стратегії тестування.

Ніколи думати, потрібно тестувати!

Будь-яке тестування вимагає змістовного підходу із застосуванням технік тест-аналізу і тест-дизайну. В іншому випадку ви ризикуєте назавжди залишитися «monkey-тестером», цінність праці якого буде мізерна. В цілому, ключові положення тест-аналізу і тест-дизайну застосовні як до тестування десктоп-додатків, так і до веб, але з суттєвим застереженням: ви повинні враховувати всі згадані в статті нюанси. Окремо закцентувати увагу на тому, що без стійкого розуміння методик і способів застосування тест-аналізу і тест- дизайну тестувати якісно програмне забезпечення практично неможливо.

Розглянемо класичний набір методик тест-дизайну, які можна застосовувати при тестуванні веб-додатків:

- розбиття на класи еквівалентності;
- попарне тестування;
- тестування переходів станів;
- аналіз граничних значень;
- тестування за допомогою таблиць рішень;
- методика тест-турів Віттакер і безліч інших.

Остання методика тестування відноситься до дослідницького тестування і буде розглянута у окремому тематичному розділі далі.

Подай ключ «на 13».

В даний час існує величезна безліч різноманітних інструментів, які спрощують життя всім учасникам розробки нового програмного забезпечення. Отже, не варто забувати про те, що крім розвитку особистих якостей, технічних знань і навичок, ми повинні вміти добре користуватися допоміжними інструментами, кожен раз випробовуючи все нові і нові.

Зазначимо лише деякі з них:

- **Ресайзер** - оперативна зміна усіляких налаштувань розмірів відображення, перегляд метрик, перемикання орієнтацій відображення.
- **Opera Mobile Classic Emulator** - класичний однойменний емулятор.
- **Mobile Phone Emulator** - класичний емулятор.
- **Fiddler** – програмне забезпечення для відстеження всього вашого трафіку. За допомогою цієї програми можна відправляти помилкові запити на сервер з потрібними вам параметрами.
- **Xenu Link Evaluator** (альтернатива - Black Widow) - «чекер» веб-додатку на предмет наявності в ньому «битих» посилань. Також можна використовувати його для формування карти додатку.
- **Skipfish** - активний сканер вразливостей веб-додатків.
- **OpenVAS** - безкоштовний сканер вразливостей.
- **Nikto** - веб-сканер, який перевіряє веб-сервери на найчастіші помилки, що виникають зазвичай через людський фактор.