



Лабораторні роботи з навчального курсу «Software testing for universities»

**Провідна українська компанія
з тестування програмного
забезпечення QATestLab
Ukrainian HI-Tech Initiative**

2019
Київ

ЗМІСТ

Лабораторна робота №1. Введення в тестування	3
Лабораторна робота №2. Види тестування	11
Лабораторна робота №3. Веб-тестування та чеклісти	15
Лабораторна робота №4. Тестування зручності використання	26
Лабораторна робота №5. Кросбраузерне тестування.....	32
Лабораторна робота №6. Тестування веб-проектів.....	38
Лабораторна робота №7. Функціональне тестування	44
Лабораторна робота №8. Технічне тестування.....	49
Лабораторна робота №9. Види тестування, пов'язані зі змінами.....	55
Лабораторна робота №10. Тест-дизайн та тест-кейси	59
Лабораторна робота №11. Техніки тест-дизайну	66
Лабораторна робота №12. Тест-плани.....	70
Лабораторна робота №13. Звіти про тестування	73
Лабораторна робота №14. Мобільне тестування.....	79
Лабораторна робота №15. Мобільне тестування веб-проектів	83
Лабораторна робота №16. Інструменти тестування мобільних додатків ...	87
Лабораторна робота №17. Тестування ігор.....	94
Лабораторна робота №18. Ролі в процесі тестування ПЗ. Комунікації у сфері тестування.....	98

ЛАБОРАТОРНА РОБОТА №1

ВВЕДЕННЯ В ТЕСТУВАННЯ

Мета лабораторної роботи: отримання базових теоретичних та практичних навичок у розпізнаванні ступенів критичності дефектів (*bug*), розділенні та об'єднанні дефектів за змістом, описі основних атрибутів дефектів та роботі із системою відслідковування помилок Mantis Bug Tracker.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з основними вимогами до оформлення звітів про помилки в лекції «Введення в тестування».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Введення в тестування».
3. Виконати завдання до лабораторної роботи «Введення в тестування» в Особистому кабінеті.
4. Встановити браузері останніх версій: Mozilla Firefox, Opera, Google Chrome, Edge або інші.
5. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Введення в тестування».
4. Натиснути кнопку «Почати проходження» (рис. 1.1).

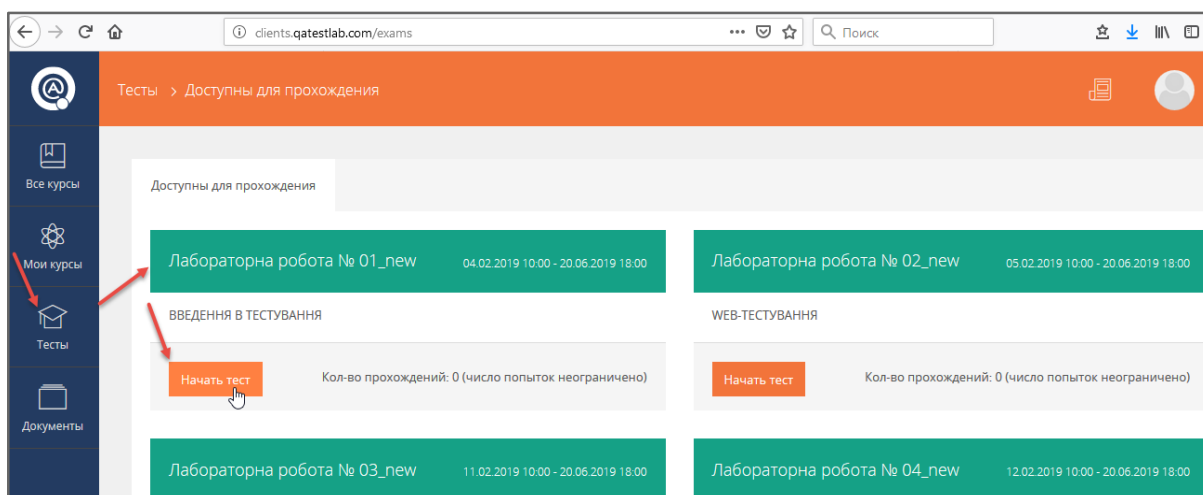


Рис. 1.1.

Зміст тесту:**Завдання 1.**

Проаналізувати список дефектів (0100016, 0100018, 0100022, 0100026, 0100030, 0100032, 0100033, 0100077) у системі відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> в проєкті «Train to deal with tickets» та відсортувати їх за ступенем критичності (спочатку найбільш критичні), шляхом зміни послідовності дефектів у відповіді тесту.

Завдання 2.

В баг-трекері Mantis <http://mantis.qatestlab.net> в проєкті «Train to deal with tickets» проаналізувати та визначити баг-репорти, які необхідно розділити на кілька окремих різних за змістом звітів. Вказати у відповіді до тестового завдання ID-дефектів та коротко описати зміст багів, на які необхідно розділити звіти.

Приклад відповіді:

Необхідно розділити баги:

13542 – на два баги (перший – про некоректну валідацію на формі реєстрації, другий – про баг в інтерфейсі на сторінці реєстрації);

16214 – на два баги (перший – про баг верстки, другий – про функціональний баг непрацюючого посилання).

Завдання 3.

В баг-трекері Mantis <http://mantis.qatestlab.net> в проєкті «Train to deal with tickets» проаналізувати дефекти: 0100017, 0100018, 0100039, 0100073,

0100074, 0100075, 0100076, 0100081 та об'єднати однотипні баг-репорти. Вказати у відповіді до тестового завдання ID-дефектів, які необхідно об'єднати в один звіт.

Приклад відповіді:

Необхідно об'єднати баги:

13450, 15263, 12354 – в один звіт

12854, 14632 – в один звіт

Завдання 4.

Знайти дефекти, переглянувши відео «Coffee» в модулі «Документи: Video» в Особистому кабінеті – <http://clients.qatestlab.com>, та оформити баг-репорти в системі відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net>. У відповіді до завдання вказати посилання на звіти про дефекти.

Порядок виконання завдання:

1. Відкрити сайт <http://mantis.qatestlab.net/>.
2. Авторизуватися в системі.
3. Створити звіт про помилку, натиснувши на посилання «Report Issue» (рис. 1.2).
4. Описати необхідні атрибути дефекту.
5. Натиснути кнопку «Submit Report».

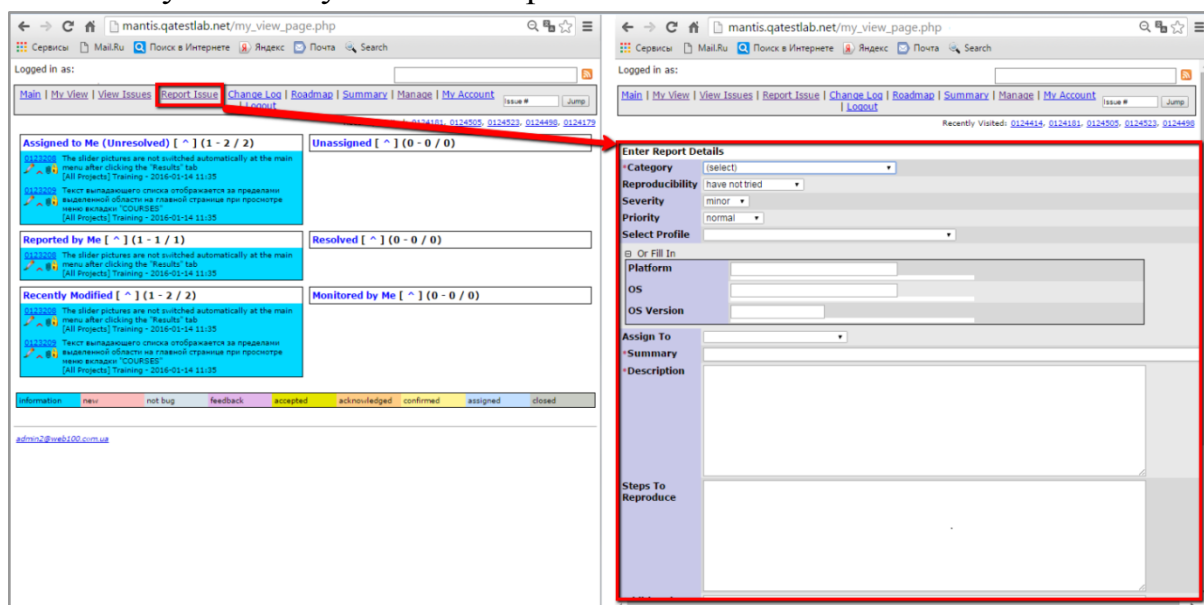


Рис.1.2. Приклад створення звіту в системі відслідковування помилок Mantis Bug Tracker

Теоретичний матеріал

При описі знайденого дефекту в системах відслідковування помилок (*bug tracking system*) вказують серйозність, у зв'язку з цим необхідно чітко розуміти, що являє собою серйозність.

Серйозність (Severity) – атрибут, що характеризує вплив дефекту на працездатність програми.

Серйозність відноситься безпосередньо до самого дефекту та показує рівень, якість взаємодії користувача й системи або програми. Серйозність дефекту визначає ймовірність збою в роботі системи та наслідки цієї помилки, якщо така буде виявлена. Також необхідно відзначити, що серйозність дефекту часто не змінюють, оскільки це постійний параметр, який змінюється тільки у зв'язку з появою нової інформації про дефект, наприклад, доповнень у сценарії користувача або нових можливих обхідних рішень.

Розрізняють такі види серйозності (Severity):

- **блокуюча (blocker)** – помилка, що призводить до додатку до неробочого стану, у результаті якого подальша робота із системою, що знаходиться на тестуванні, або її ключовими функціями стає неможлива. Рішення проблеми необхідно для подальшого функціонування системи;
- **критична (critical)** – критична помилка, неправильно працююча ключова бізнес логіка, діра в системі безпеки, проблема, яка призвела до тимчасового падіння сервера або призводить до неробочого стану деяку частину системи, без можливості вирішення проблеми, використовуючи інші вхідні точки;
- **значна (major)** – значна помилка, частина основної бізнес-логіки працює некоректно. Помилка не критична або є можливість для роботи з тестовою функцією, використовуючи інші вхідні точки;
- **незначна (minor)** – незначна помилка, не порушує бізнес-логіку частини програми, що знаходиться на тестуванні, очевидна проблема інтерфейсу користувача, наприклад, функціональна проблема;
- **тривіальна (trivial)** – помилка, яка не стосується бізнес-логіки додатка, погано відтворювана проблема, малопомітна через інтерфейс

користувача, проблема сторонніх бібліотек або сервісів, проблема, яка не впливає на загальну якість продукту;

- **текстова (text)** – помилка в тексті, друкарська помилка, пропущена кома, інша граматична помилка;
- **потрібне налаштування (tweak)** – поліпшення, яке може бути в результаті більш точного налаштування системи;
- **побажання, фіча (feature)** – побажання до системи або продукту, що вимагає доопрацювання.

Однотипними дефектами є помилки в одному тексті на одній сторінці. Або однакове зміщення елемента інтерфейсу у всіх браузерях. Або однаково не реагуючі на натискання пункти в головному меню. Розуміння, чи однотипні два дефекти чи ні, приходить із досвідом та знаннями структури проекту.

Переваги об'єднання однотипних дефектів в один звіт:

1. Процес перегляду звітів, внесених до системи відслідковування помилок (*bug tracking system*), відбувається швидше. Немає необхідності зупинятися окремо на схожих дефектах.

2. Процес виправлення дефектів проходить швидше. Якщо для виправлення декількох дефектів в інтерфейсі необхідно оновити один і той же файл, зникає необхідність відкривати-редагувати та залишати коментарі до одного й того ж файлу декілька разів.

3. Процес тестування відбувається швидше у зв'язку з тим, що розробники виправляють однотипні дефекти разом, тестувальник також може їх перевірити всі разом. Тим самим зменшується час на перегляд описаних дефектів, особливо коли для оновлення усуненого дефекту необхідно оновити інший сервер.

Альтернативною можливістю об'єднання дефектів, є додавання посилань. Коли при перегляді одного звіту видно всі дефекти, що відносяться до нього.

Об'єднання різних за змістом дефектів в одному звіті може призвести до того, що один із дефектів у звіті буде вирішено не виправляти, у результаті чого, звіт буде закритий із резолюцією «Не буде виправлений», а тестувальнику знадобиться оформити окремий звіт на помилку, яка вимагає виправлення.

Контрольні запитання

1. Визначити основні властивості та відмінності встановлених браузерів: Mozilla Firefox, Opera, Google Chrome, Edge.
Обґрунтувати розповсюдженість використання встановлених браузерів.
2. Для чого призначені та які існують системи відслідковування помилок?
3. Що повинен містити якісний опис помилки?
4. Що відносять до атрибутів дефекту?
5. Пояснити за яким принципом були відсортовані дефекти в проекті «Train to deal with tickets» в Mantis Bug Tracker. Обґрунтувати відповідь.
6. Обґрунтувати розділення та об'єднання дефектів у завданні 2-3.
7. Для чого здійснюють об'єднання однотипних дефектів в один звіт? Навести приклади таких дефектів та обґрунтувати відповідь.

Список літератури:

1. ISTQB Сертифицированный тестировщик. Программа обучения Базового уровня [Электронный ресурс]. – Режим доступа: <http://ru.scribd.com/doc/56795254/Istqb-Ctfl-Syllabus-2011-Ru>.
2. Техника написания хороших кратких Summary баг-репортов [Электронный ресурс]. – Режим доступа: <http://svyatoslav.biz/education/bug-reports-summary/>.
3. Забавные баги [Электронный ресурс]. – Режим доступа: <http://habrahabr.ru/company/ua-hosting/blog/244479/>.
4. Быстрое тестирование. Роберт Калбертсон. Стр. 120-131, 167-169 [Электронный ресурс]. – Режим доступа: <http://lib.mdpu.org.ua/e-book/vstup/L/Kalbertson.pdf>.
5. Portnov computer school. Bug Tracking Databases [Электронный ресурс]. – Режим доступа: <http://www.youtube.com/watch?v=IlwFD7gISOw>.
6. Тестирование программного обеспечения (Software Testing по SWEBOK) [Электронный ресурс]. – Режим доступа: http://swbokwiki.org/Chapter_4:_Software_Testing.
7. Bug Advocacy: Effective Bug Investigation and Reporting [Электронный ресурс]. – Режим доступа: <http://www.testingeducation.org/BBST/bugadvocacy/>.
8. How to Set Defect Priority and Severity (with Defect Triage Process) [Электронный ресурс]. – Режим доступа: <http://www.softwaretestinghelp.com/how-to-set-defect-priority-and-severity-with-defect-triage-process/>.
9. Список книг (по тестированию и не только) [Электронный ресурс]. – Режим доступа: http://okiseleva.blogspot.ru/2014/02/blog-post_6.html.
10. Основные атрибуты баг-репорта [Электронный ресурс]. – Режим доступа: <http://training.qatestlab.com/front-page/blog/technical-articles/attributes-bug-report/>.
11. Как правильно составить описание бага в баг-трекере. принцип «Что? Где? Когда?» [Электронный ресурс]. – Режим доступа: <http://training.qatestlab.com/front-page/blog/technical-articles/description-bug-tracker/>.
12. Как правильно необходимо описывать шаги в баг-репорте [Электронный ресурс]. – Режим доступа: <http://training.qatestlab.com/front-page/blog/technical-articles/steps-bug-report/>.

13. С чего начинать описание шагов воспроизведения дефекта (бага)?
[Электронный ресурс]. – Режим доступа:
<http://training.qatestlab.com/front-page/blog/technical-articles/steps-bug-reproduction>.
14. Правила описания результатов в баг-репорте [Электронный ресурс]. – Режим доступа:
<http://training.qatestlab.com/front-page/blog/technical-articles/results-bug-report/>.
15. Правила описания багов на английском языке. Использование пассива [Электронный ресурс]. – Режим доступа:
<http://training.qatestlab.com/front-page/blog/technical-articles/passive-voice-bugs>.

ЛАБОРАТОРНА РОБОТА №2

ВИДИ ТЕСТУВАННЯ

Мета лабораторної роботи: ознайомлення з видами тестування.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Види тестування».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Види тестування».
3. Виконати завдання до лабораторної роботи «Види тестування» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Види тестування».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Проаналізувати список дефектів в проекті «Kinds of tickets» та відсортувати їх за категоріями (залежно від виду тестування).

1. Проаналізувати знайдені дефекти в проекті «Kinds of tickets» у системі відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> та відсортувати їх за категоріями (залежно від виду тестування).
2. У відповіді до завдання вказати категорії та ID звітів про дефекти.

Теоретичний матеріал

Існує кілька ознак, за якими прийнято класифікувати види тестування, а саме:

- *за об'єктом тестування:*
 - функціональне тестування;
 - тестування продуктивності;

- навантажувальне тестування;
 - стрес-тестування;
 - тестування стабільності;
 - конфігураційне тестування;
 - юзабіліті-тестування;
 - тестування інтерфейсу користувача;
 - тестування безпеки;
 - тестування локалізації;
 - тестування сумісності;
- **за знанням системи:**
- тестування чорного ящика;
 - тестування білого ящика;
 - тестування сірого ящика;
- **за ступенем автоматизації:**
- ручне тестування;
 - автоматизоване тестування;
 - напівавтоматизоване тестування;
- **за ступенем ізольованості компонентів:**
- модульне тестування;
 - інтеграційне тестування;
 - системне тестування;
- **за часом проведення тестування:**
- альфа-тестування:
 - димове тестування (*smoke testing*);
 - тестування нової функції (*new feature testing*);
 - підтверджуюче тестування;
 - регресійне тестування;
 - приймальне тестування;
 - бета-тестування;
- **за ознакою позитивності сценаріїв**
- позитивне тестування;
 - негативне тестування;
- **за ступенем підготовленості до тестування:**
- тестування за документацією (*формальне тестування*);
 - інтуїтивне тестування (*ad hoc testing*).

На рисунку 2.1 у вигляді схеми представлені види тестування.

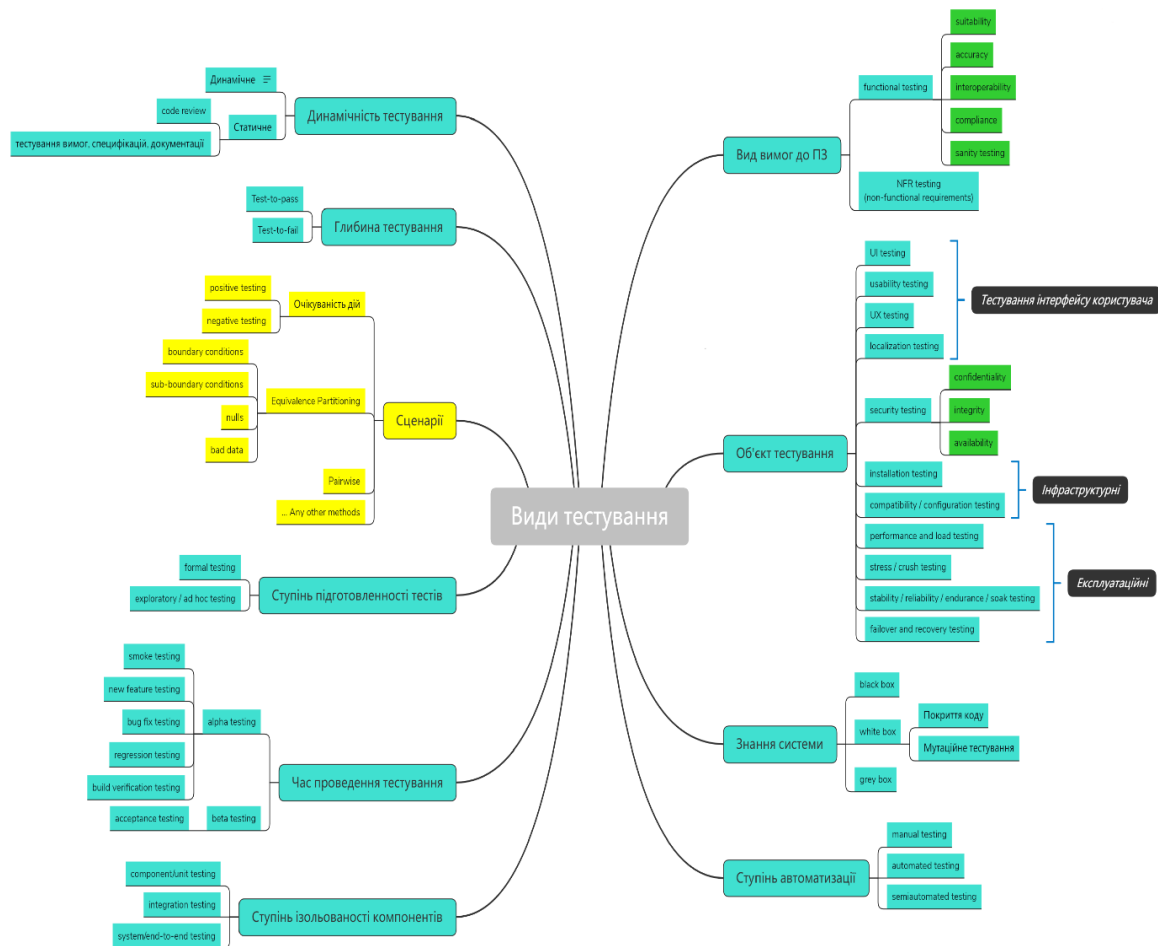


Рис. 2.1. Види тестування

Контрольні запитання

1. Пояснити на підставі чого були віднесені дефекти до тієї, чи іншої категорії в проєкті «Kinds of tickets» в <http://mantis.qatestlab.net>. Обґрунтувати відповідь.
2. Для чого виставляється категорія дефекту?
3. Як впливає категорія дефекту на черговість виправлення дефекту?
4. Вказати переваги та недоліки використання тестових сценаріїв у тестуванні ПЗ на прикладі.

Список літератури:

1. Какие тесты вам нужны? Часть 2. Матрица видов тестирования [Электронный ресурс]. – Режим доступа: <http://habrahabr.ru/post/257159/>.
2. Виды тестирования [Электронный ресурс]. – Режим доступа: <http://xmindshare.s3.amazonaws.com/preview/5HiG-QziDBqK-00619.png>.
3. Регрессионное тестирование [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Регрессионное_тестирование.
4. Гленфорд Майерс, Том Баджетт, Кори Сандлер. Искусство тестирования программ, 3-е издание = The Art of Software Testing, 3rd Edition. — М.: «Диалектика», 2012. — 272 с.
5. Лайза Криспин, Джанет Грегори. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams. – М.: «Вильямс», 2010. – 464 с.
6. Канер Кем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. – Киев: ДияСофт, 2001. – 544 с.
7. Калбертсон Роберт, Браун Крис, Кобб Гэри. Быстрое тестирование. – М.: «Вильямс», 2002. – 374 с.
8. Сеницын С. В., Налютин Н. Ю. Верификация программного обеспечения. – М.: БИНОМ, 2008. – 368 с.
9. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. – СПб.: Питер, 2004. – 320 с.
10. Основы разработки тестовых сценариев [Электронный ресурс]. – Режим доступа: <https://delta-course.org/docs/Delta2014S-T2-L6.pdf>

ЛАБОРАТОРНА РОБОТА №3

ВЕБ-ТЕСТУВАННЯ ТА ЧЕКЛІСТИ

Мета лабораторної роботи: отримання базових теоретичних та практичних навичок в розпізнаванні етапів тестування та створенні контрольного списку (*checklist*).

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Веб-тестування та чеклісти».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Веб-тестування та чеклісти».
3. Виконати завдання до лабораторної роботи «Веб-тестування та чеклісти» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Веб-тестування та чеклісти».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1. Доповнити та пройти контрольний список тестування (*checklist*) верстки сайту <http://prestashop.qatestlab.com.ua/en/> – вкладка «Верстка».

Порядок виконання завдання:

1. Завантажити приклад чекліста в Особистому кабінеті у завданні до лабораторної роботи «Веб-тестування та чеклісти» (рис.3.1).

2. Вказати прізвище та ім'я власника у назві контрольного списку.
3. Додати мінімум 5-7 пунктів до існуючих пунктів контрольного списку на вкладці «Верстка», не повторюючись з вже доданими, та відмітити їх будь-яким кольором.
4. Провести тестування верстки сайту <http://prestashop.qatestlab.com.ua/en/>, перевіряючи всі пункти контрольного списку на вкладці «Верстка» в одному з браузерів (Firefox, Google Chrome, Edge, Opera або інший) в останній або передостанній версії.
5. Вказати назву та версію браузера. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на звіт у системі відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> або додати тему багу за принципом «Що? Де? Коли?».
6. Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» додати до відповіді.

	Сайт http://opencart.qatestlab.net	Google Chrome 74.0.3729.131	Firefox 66.0.4	Edge 44.17763.1.0	Примітки
1					
2	Наявність елементів сторінок				
3	Перевірити коректне відображення сторінок згідно файлів дизайну	Skipped	Skipped	Skipped	
4	Перевірити наявність логотипу	Failed	Failed	Passed	http://mantis.qatestlab.net/view.php?id=
5	Перевірити наявність головного меню	Passed			
6	Перевірити наявність підвалу сайту				
7	Відображення елементів				
8	Перевірити коректне відображення шрифтів тексту				
9	Перевірити коректне відображення кнопок, блоків меню і т.д.				
10	Перевірити відображення кольорової гами усіх елементів				
11	Перевірити коректне розміщення банерів				
12	Активні елементи				
13	Перевірити наявність підказок для активних елементів				
14	Перевірити реагування активних елементів на наведення				
15	Зміст сторінок				
16	Перевірити орфографію/граматику контенту сайту				

Рис. 3.1. Приклад контрольного списку верстки

Завдання 2.

Занести у систему відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> 5-7 багів, знайдених під час проведення тестування верстки сайту <http://prestashop.qatestlab.com.ua/en/>.

Теоретичний матеріал

Важливою особливістю веб-тестування (*web-testing*) є часові рамки. Якщо при тестуванні програмного забезпечення (*software*) у тестувальника є місяці (а інколи і роки), то у веб-тестувальника є лише дні та тижні (в кращому випадку) на тестування запропонованого сайту. Отже, якщо при тестуванні програм дозволяється й потрібно формувати детальні плани тестування, описувати тестові випадки (*test cases*) на основі отриманої від розробника документації, то при тестуванні веб-сторінок це може значно продовжити терміни публікації матеріалів в мережі Інтернет. Таким чином, на веб-тестування може виділятися до 50% загального бюджету та часових ресурсів.

Веб-тестування (web-testing) – тестування програмного забезпечення, сфокусоване на веб-додатках (*web-applications*), яке здійснюється з застосуванням систем, що використовують веб-технології (*web-based system*), та вирішують такі питання як: безпека веб-додатків, базова функціональність сайту, доступність користувачам та ін.

Веб-додатки (web-applications) – клієнт серверний додаток, в якому клієнтом виступає браузер, а сервером – веб-сервер.

Веб-сайт (website) – система з декількох сторінок, що має одну адресу в мережі Інтернет.

Верстка (layout) – етап дизайну сторінки сайту, що представляє собою просторове розміщення на ній текстових елементів і графічних зображень відповідно до концепції оформлення ресурсу.

Веб-дизайн (web-design) – галузь веб-розробки, а також різновид дизайну, до завдань якої входить проектування веб-інтерфейсів (*Web UI*) для сайтів або веб-додатків.

Розрізняють наступні етапи тестування веб-проектів (*web-project stages of testing*):

1. Вивчення документації (*documentation analysis*).
2. Тестування верстки (*layout testing*).
3. Функціональне тестування (*functional testing*).
4. Тестування практичності/перевірка на простоту використання (*usability testing*).
5. Тестування безпеки (*security testing*).
6. Тестування продуктивності сайту (*performance testing*).

Вимоги до розмітки (верстки):

- чи немає помітних оку невідповідностей: поламани блоки, не стикування кольору, некоректне відображення тексту навколо зображень;
- якщо дизайн згідно ТЗ розрахований на певну ширину, то, незалежно від браузера, не повинна з'являтися горизонтальна прокрутка;
- під час зменшення розміру вікна менше мінімального згідно ТЗ не повинно нічого ламатися, фони не повинні «розпливатися»;
- сайт повинен виглядати однаково у всіх стандартних розширеннях екрану (1024x600, 1024x768, 1152x864, 1280x800, 1280x1024, 1440x900, 1680x1050, 1920x1080): не повинно нічого ламатися, не повинні різко обриватися фони при великих розширеннях;
- при відключених зображеннях, написи (особливо логотип та головне меню сайту) повинні залишатися читабельними, на всіх інформаційних картинках повинні бути підписані акуратним, невеликим, сірим шрифтом (відключення картинок: Chrome → Settings → Show advanced → Site settings → Images → вибрати опцію «Do not show any images»; Firefox → download Web developer plugin → після установки вибрати Images → Disable Images → Disable All Images);
- працездатність при вимкненому JavaScript. Увесь критично важливий функціонал сайту повинен бути доступний без JS (відключення JavaScript: Chrome → Settings → Show advanced → Site settings → Do not allow any site to run JavaScript; Firefox → download Web developer plugin → після установки вибрати Images → Disable JavaScript → Disable All JavaScript).

Перевірка однотипності/симетричності в інтерфейсі:

- однотипність/симетричність використання відступів;
- ширина колонок та контентного поля;
- позиціонування елементів верхнього та нижнього колонтитулів на всіх сторінках;
- позиціонування банерів, правого та лівого блоку;
- розмір текстів;
- відступи між абзацами;

- відстані між рядками;
- елементи управління (кнопки, чекбокси, випадаючі списки);
- розмір/колір/тип шрифту;
- наявність стилів оформлення заголовків першого, другого та більше рівнів;
- коректне відображення тексту навколо зображень;
- напрямки тіней у всіх елементів управління;
- назви однакових елементів у різних місцях;
- чи мають елементи, що призначені для натискання, вказівник «pointer»; елементи, які можна перетягувати – вказівник «move» або «crosshair»; неактивні/недоступні елементи – вказівник «default» (рис.3.2).









Вид	Значення
	default
	crosshair
	help
	move
	pointer
	progress
	text
	wait

Рис. 3.2. Можлива форма вказівника при наведенні на відповідний елемент

Приклади неоднотипних дефектів в інтерфейсі.

Заголовки тексту не вирівняні відносно один одного (рис. 3.3):

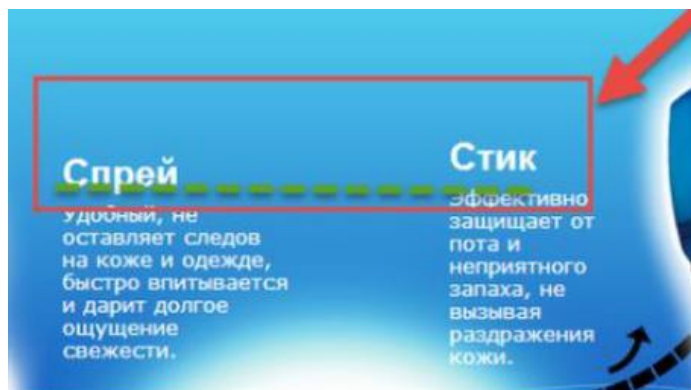


Рис. 3.3. Приклад багу із розміщенням заголовків тексту

Відстані між підписами полів та міткою обов'язкового поля не збігаються. Довжина та розташування поля «*Quantity*» не збігається з полем «*Starch*» (рис.3.4):

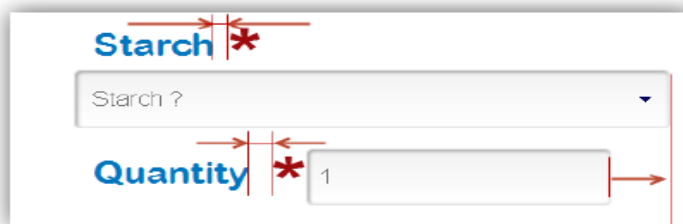


Рис. 3.4. Приклад невідповідності розміщення полів та міткою

Приклади дефектів зі списком, що випадає (рис. 3.5 - 3.6).

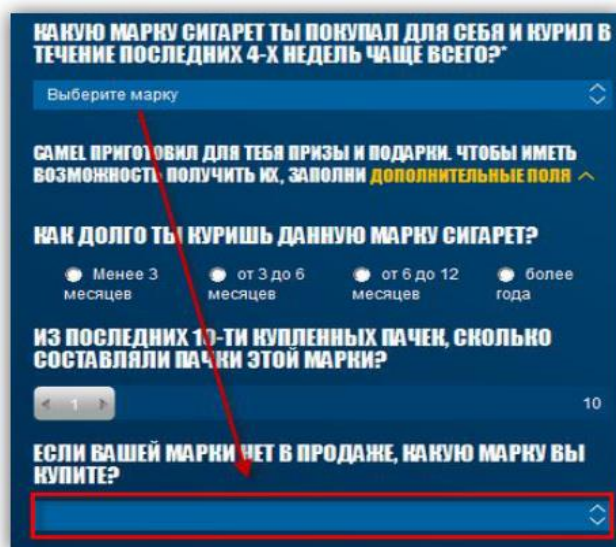


Рис. 3.5. Не для всіх випадаючих списків є значення за замовчуванням

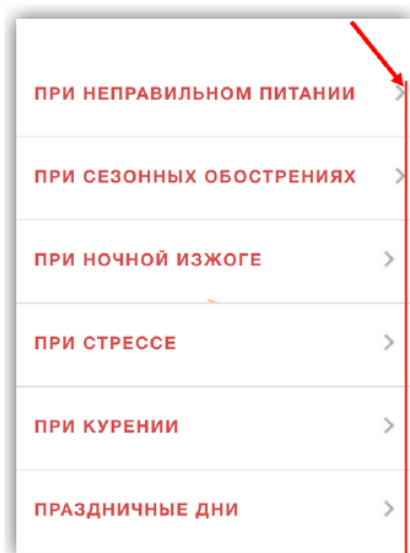


Рис. 3.6. Пример нецентрированных иконок выпадающего меню

Примеры дефектов на форме (рис. 3.7 - 3.8).

Подпись не выровнян относительно поля:

A registration form with labels and input fields: "Your Name:", "Your Email:", "Friend's Name:", "Friend's Email:", and "Personal Note:". The "Personal Note:" label is underlined and has a red arrow pointing to it, indicating a misalignment with the text area.

Рис. 3.7. Пример невыровненных подписей относительно полей ввода

A phone number input field with the label "Enter your phone number:". The input field contains the text "(+7)" followed by a placeholder "e number without country". A red arrow points to the text, indicating a truncation issue.

Рис. 3.8. Текст по умолчанию обрывается в поле ввода номера телефона

Чекліст – це документ, який описує що має бути протестовано. При цьому чекліст може бути абсолютно різного рівня деталізації. На скільки детальним буде чекліст залежить від вимог до звітності, рівня знання продукту співробітниками й складності продукту.

Чекліст – один з фундаментальних інструментів тестування, який дозволяє не забувати про важливі тести, фіксувати результати роботи та відстежувати статистику про статус програмного продукту.

Головний принцип чеклістів полягає в тому, що кожен тестувальник по-своєму проходить їх, розширюючи тестовий набір своєю експертизою.

Навіщо потрібен чекліст?

- ✓ не забути необхідні тести;
- ✓ для поділу завдань за рівнем кваліфікації;
- ✓ для збереження звітності та результатів тестування.

Що має бути в чеклісті?

- ✓ список перевірок (з необхідним ступенем деталізації);
- ✓ оточення перевірки:
 - збірка, на якій проводилося тестування;
 - тестове оточення (якщо є);
 - інформація про тестувальників;
- ✓ результат перевірки.

Пункти можуть містити як лінійну структуру, так і деревоподібну, з розділами/підрозділами або без них. Вони повинні бути максимально короткі та в той же час зрозумілі тестувальнику, який ще не знайомий з додатком. Всі пункти повинні бути оформлені однією мовою.

Як правило, кожен чекліст має кілька стовпців. Кожен стовпець призначений для тестування на окремій платформі. Слід завжди вказувати назву пристрою, назви браузерів та їх версії.

Тестувати додаток може декілька людей одночасно. У цьому випадку кожен тестувальник бере по одній-дві платформи та тестує тільки на них. При цьому слід навпроти кожної платформи вказати тестувальника, який виконує зазначений обсяг робіт.

При проходженні чеклістів тестувальник зазначає статус навпроти кожного тестованого пункту. Можливі такі варіанти статусів:

- «Passed» – перевірка пройдена успішно, багів не знайдено;
- «Failed» – знайдений один або більше багів;
- «Blocked» – неможливо перевірити, тому що один з багів блокує поточну перевірку;
- «In Progress» – поточний пункт, над яким працює тестувальник;
- «Not run» – ще не перевірено;

▪ «Skipped» – не буде перевірятися з будь-якої причини. Наприклад, поточний функціонал ще не реалізований.

Для більшої наочності, як правило, кожен зі статусів має свій колір.

Всі заведені по чеклісту баг-репорти повинні бути додані в примітки до комірок зі статусом «Failed». Цілком припустимо додавати примітки до комірок з іншими статусами, якщо це необхідно. Для комірок зі статусом «Blocked» примітки з посиланнями на баг-репорти також обов'язкові. Однак, як правило, примітка в комірці зі статусом «Blocked» посилається на раніше заведений баг-репорт, який в одному з попередніх пунктів вже відзначений як «Failed». Іншими словами: пункт, одного разу зазначений як «Failed», може бути блокером для декількох або всіх наступних пунктів чекліста (рис. 3.9).

Passed	Passed	Passed	Passed
Failed +	http://bt-w.qatestlab.com/view.php?id=...		
Failed	Failed	Passed	Passed
Blocked	Blocked	Passed	Passed
Blocked	Blocked	Skipped	Skipped

Рис. 3.9. Статуси перевірок у чеклісті

Основні моменти, які варто враховувати при роботі з чеклістами:

- ✓ після завершенні проходження чекліста не повинно залишитися комірок зі статусом «Not run»;
- ✓ всі комірки зі статусом «Failed» і «Blocked» обов'язково повинні мати примітки з посиланнями на баг-репорти;
- ✓ статус «Passed» встановлюється тільки для пунктів, які перевірені і не містять помилок.

Правила складання чеклістів:

1. Один пункт – одна операція. Пункти чекліста – це однозначні атомарні і повні операції. Наприклад, додавання товару до корзини сайту та оплата замовлення – це два різні завдання. У списку перевірок подібні операції оформлюються окремими пунктами: додавання товару в корзину, відправлення оплати за замовлення.

2. Починати пункти з іменника. Мета чекліста – врахувати всі дії для найбільш повного покриття тестами ПЗ, тому складаючи пункти слід дотримуватися уніфікованої форми. Для зрозумілого і однозначного уявлення пункти краще починати з іменника – «Перевірка», «Додавання»,

«Відправлення» або дієслова неозначеної форми – «Перевірити», «Додати», «Відправити».

3. Скласти чекліст за рівнями деталізації. Для зручності проходження чекліста, найкраще складати тести в послідовності використання функціоналу. Наприклад, розділ «Реєстрація та Особистий профіль»: реєстрація на сайті, редагування профілю; розділ «Форма зворотного зв'язку»: валідація полів, відправка листа, доставка листа.

Контрольні запитання

1. Що відносять до дефектів розмітки (верстки)? Навести приклади та обґрунтувати відповідь.
2. За яким принципом були додані нові пункти до чекліста? Обґрунтувати важливість нових пунктів при тестуванні сайту <http://prestashop.qatestlab.com.ua/en/>.
3. Для чого в примітці до пункту чекліста вказується посилання на звіти у системі відслідковування помилок? Обґрунтувати необхідність посилання на звіт в чеклісті.
4. Обґрунтувати необхідності створення пунктів та підпунктів в чеклісті.
5. Які існують переваги використання чеклістів порівняно з іншою подібною документацією для проведення тестування? Навести приклади та обґрунтувати відповідь.

Список літератури:

1. Web testing [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Web_testing.
2. Правильное тестирование веб-сайта, или как обеспечить себе спокойный сон [Електронний ресурс]. – Режим доступу: <http://fresh-design.com.ua/blog/technology/website-testing>.
3. Software Testing Help. “Web Testing: Complete guide on testing web applications” [Електронний ресурс]. – Режим доступу: <http://www.softwaretestinghelp.com/web-application-testing/>.
4. Web-testing [Електронний ресурс]. – Режим доступу: <http://www.edb.utexas.edu/minliu/multimedia/PDFfolder/WebTestingPadolina.pdf>.
5. Software Testing Help. “Entries Tagged 'Cookie Testing. Website Cookie Testing, Test cases for testing web application cookies?’” [Електронний ресурс]. – Режим доступу: <http://www.softwaretestinghelp.com/category/cookie-testing/>.
6. Обзор решений для тестирования сайтов разные виды тестов для веб-приложений и применяемое ПО [Електронний ресурс]. – Режим доступу: <http://hostinfo.ru/articles/442>.
7. Принцип золотого сечения в дизайне сайтов [Електронний ресурс]. – Режим доступу: <http://design-mania.ru/web-design/zolotoe-sechenie/>.
8. Золотое сечение для веб-дизайна и типографики – особенности использования и некоторые советы [Електронний ресурс]. – Режим доступу: <https://wayup.in/blog/golden-ration-in-web-design-and-typography-usage-features-and-tips>.
9. Что такое чеклісты и как с ними работать [Електронний ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/work-with-checklist/>.

ЛАБОРАТОРНА РОБОТА №4

ТЕСТУВАННЯ ЗРУЧНОСТІ ВИКОРИСТАННЯ

Мета лабораторної роботи: отримання базових теоретичних та практичних навичок в визначенні, застосуванні тестування зручності використання та складанні простого контрольного списку.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Тестування зручності використання».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Тестування зручності використання».
3. Виконати завдання до лабораторної роботи «Тестування зручності використання» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Тестування зручності використання».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1. Доповнити та пройти чекліст (*checklist*) з тестування зручності використання сайту <http://prestashop.qatestlab.com.ua/en/> – вкладка «Usability».

Порядок виконання завдання:

1. Завантажити приклад чекліста в Особистому кабінеті у завданні до лабораторної роботи «Тестування зручності використання» (рис. 4.1).
2. Вказати прізвище та ім'я власника у назві контрольного списку.

3. Додати мінімум 5-7 пунктів до існуючих пунктів контрольного списку на вкладці «Usability», не повторюючись з вже доданими, та відмітити їх будь-яким кольором.

4. Провести юзабіліті тестування сайту

<http://prestashop.qatestlab.com.ua/en/>, перевіривши всі пункти контрольного списку на вкладці «Usability» в одному з браузерів (Firefox, Google Chrome, Edge, Opera або інший) в останній або передостанній версії.

5. Вказати назву та версію браузера. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на звіт у системі відслідковування помилок Mantis Bug Tracker

<http://mantis.qatestlab.net> або додати тему багу за принципом «Що? Де? Коли?».

6. Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» додати до відповіді.

1	Сайт opencart.qatestlab.net	Критерії проходження / Підпункти	Google Chrome 74.0.3729.131	Firefox 66.0.4	Edge 44.17763.1.0	Примітки/Посилання на баг-репорт
2	Швидкість завантаження	Сайт швидко завантажується (1-2 секунди)	Failed	Failed	Passed	http://mantis.qatestlab.net/view.php?id=
3	Підстроювання під географію цільової аудиторії	Сайт оптимізовано для подання в тому регіоні, для якого призначений: інформація викладена на відповідній мові (або є вибір мови), дані представлені в звичайній для відвідувача метричній системі	Passed			
4	Зручність перемикачів мови	Якщо на сайті є вибір мови, то його перемикач просто перезавантажує сторінку, на якій зараз знаходиться користувач, а не перекидає його на головну сторінку сайту	not run	not run	not run	
5	Наявність інтуїтивно зрозумілих іконок	Скрізь, де це доречно і можливо, текстові пункти доповнені іконками	not run	not run	not run	
6	Швидкий доступ до кнопок CTA (Call To Action)	Можливість купити товар / замовити послугу / замовити зворотний дзвінок є на кожній сторінці сайту	not run	not run	not run	
7	Однозначність інтерфейсу	Структура сайту (шапка, підвал, головне меню і т.д.) - однакова на всіх сторінках, винятком можуть бути сторінки кошика або оформлення замовлення	not run	not run	not run	
8	Однозначність і зрозумілість	Потрапивши на будь-яку сторінку сайту, користувач одразу ж розуміє, що це за сайт, якої тематики: завдяки слогану, логотипу, зображенням в шапці, заголовку сторінки і т.д. Всі стандартні елементи відвідувач знаходить на звичних місцях:	not run	not run	not run	
9	Передбачуване місцезнаходження ключових елементів	- логотип компанії - зліва вгорі, - контакти - справа вгорі, - рядок пошуку - угорі ліворуч або вгорі по центру	not run	not run	not run	
10	Зручність взаємодії з лого сайту	Логотип клікабельний та веде на головну сторінку	not run	not run	not run	
11	Карта сайту	На сайті є посилання на карту сайту	not run	not run	not run	

Рис. 4.1. Приклад контрольного списку з тестування зручності використання сайту

Завдання 2.

Занести у систему відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> 5-7 багів, знайдених під час проведення тестування зручності використання сайту <http://prestashop.qatestlab.com.ua>.

Теоретичний матеріал

Тестування зручності використання (usability testing) – метод тестування, спрямований на встановлення ступеня зручності використання, здатності до навчання користувача, зрозумілості та привабливості для користувачів розроблювального продукту в контексті заданих умов.

Тестування зручності користування дає оцінку рівня зручності використання програми **за наступними пунктами:**

- **продуктивність, ефективність (efficiency)** – скільки часу та кроків знадобиться користувачеві для завершення основних завдань програми, наприклад, розміщення новини, реєстрації, покупка та ін.? (*менше – краще*);
- **правильність (accuracy)** – скільки помилок зробив користувач під час роботи з додатком? (*менше – краще*);
- **активізація в пам'яті (recall)** – як багато користувач пам'ятає про роботу програми після припинення роботи з нею на тривалий період часу? (*повторне виконання операцій після перерви має проходити швидше ніж у нового користувача*);
- **емоційна реакція (emotional response)** – як користувач себе почуває після завершення завдання - розгублений, в стані стресу? Чи порекомендує користувач систему своїм друзям? (*позитивна реакція – краще*).

Важливо звертати увагу на:

- ✓ простоту використання сайту або інтерфейсу;
- ✓ ефективність використання;
- ✓ запам'ятовуваність;
- ✓ помилки, їх кількість та серйозність;
- ✓ задоволення користувача (*суб'єктивне*).

Правильне уявлення про тестування зручності використання (*usability testing*) та застосування в роботі приходить з досвідом та знаннями структури проекту, а також вимог до проекту. Також слід звернути увагу на те, що саме підлягає тестуванню:

- веб-додаток;
- мобільний додаток;
- десктоп-додаток.

У кожного з цих видів додатків можуть бути різні цілі, використовувані ресурси (наприклад пам'ять, енергія та ін.), оточення, ступінь досвіду користувача та багато іншого.

Розглянемо приклади:

1. Веб-додаток (інтернет-магазин).
2. Спеціалізований веб-додаток для формування бухгалтерських звітів.

Інтернет-магазин:

Мета – користувачі повинні багаторазово користуватися сайтом, весь цикл повинен бути простий, містити якомога менше кроків, інтерфейс повинен бути інтуїтивно зрозумілий для будь-якого користувача.

Критично важливо перевірити простоту використання (основного функціоналу) та зрозумілість використання інтерфейсу без додаткового вивчення яких-небудь документів або правил використання.

Веб-додаток для формування бухгалтерських звітів:

Мета – користувач повинен мати можливість скласти звіти з усіма можливими даними, що зберігаються в БД в різній послідовності за певний період часу.

Не так важлива простота використання як функціональні можливості та правильність даних, тобто скоріше всього в складній системі працюватимуть користувачі, які заздалегідь навчені всім можливим алгоритмам роботи та знають де й що натискати.

Контрольний список (checklist) – один з фундаментальних інструментів тестування: це документ, який описує, що має бути протестовано та дозволяє не забувати про важливі тести, фіксувати результати роботи та відстежувати статистику про статус програмного продукту.

Навіщо потрібен контрольний список (checklist)?

- ✓ не забути необхідні тести;
- ✓ для поділу завдань за рівнем кваліфікації;
- ✓ для збереження звітності та результатів тестування.

Що має бути в контрольному списку (checklist)?

- ✓ список перевірок (з необхідним ступенем деталізації);
- ✓ оточення перевірки:
 - збірка, на якій проводилося тестування;
 - тестове оточення (якщо є);
 - інформація про тестувальників;
- ✓ результат перевірки.

Контрольні запитання

1. Що відносять до дефектів зручності використання? Навести приклади та обґрунтувати відповідь.
2. За яким принципом були додані нові пункти до чекліста? Обґрунтувати важливість нових пунктів при тестуванні сайту <http://prestashop.qatestlab.com.ua/en/>.
3. Якого принципу необхідно дотримуватися при поліпшенні зручності користування?
4. Які основні пункти та стовпці повинен містити контрольний список з тестування зручності використання (*usability testing*)?
5. Обґрунтувати важливість тестування зручності використання.

Список літератури:

1. Что такое юзабилити? [Электронный ресурс]. – Режим доступа: <http://seoprofy.ua/blog/wiki/chto-takoe-usability>.
2. Тестирование удобства пользования или Usability Testing [Электронный ресурс]. – Режим доступа: <http://www.protesting.ru/testing/types/usability.html>.
3. Что такое чеклист? [Электронный ресурс]. – Режим доступа: <http://sitechco.ru/2011/08/chto-takoe-chek-list/>.
4. Юзабилити-тестирование [Электронный ресурс]. – Режим доступа: <http://usabilitylab.ru/services/usability-testing>.
5. Thinking Aloud: The #1 Usability Tool [Электронный ресурс]. – Режим доступа: <http://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/>.
6. Гаврилюк О. Как сделать сайт удобным. Юзабилити по методу Стива Круга [Электронный ресурс]. – Режим доступа: <http://www.aweb.com.ua/seo-blog/steve-krug-usability-testing/>.
7. Обзор методов юзабилити-тестирования [Электронный ресурс]. – Режим доступа: <https://artw.ru/blog/archives/3537/>.
8. Элементарные основы юзабилити [Электронный ресурс]. – Режим доступа: <https://www.promo-webcom.by/analytics/usability/1429-elementarnyye-osnovyi-yuzabiliti/>.
9. Сергеев С. Ф. Методы тестирования и оптимизации интерфейсов информационных систем [Электронный ресурс]. – Режим доступа: <http://window.edu.ru/resource/441/80441/files/itmo1363.pdf>.
10. Юзабилити-тестирование [Электронный ресурс]. – Режим доступа: <https://www.ashmanov.com/education/articles/yuzabiliti-testirovanie/>.
11. Методы юзабилити-тестирования сайтов. Делаем сайт удобным [Электронный ресурс]. – Режим доступа: <https://adverbs.ru/blog/post-metody-yuzabilititestirovaniya-saytov-dat-bolshe-tsennosti-v-edinitsy-/>.

ЛАБОРАТОРНА РОБОТА №5

КРОСБРАУЗЕРНЕ ТЕСТУВАННЯ

Мета лабораторної роботи: отримання базових теоретичних та практичних навичок в кросбраузерному тестуванні.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Кросбраузерне тестування».
2. Ознайомитися з матеріалом лекції «Кросбраузерне тестування».
3. Виконати завдання до лабораторної роботи «Кросбраузерне тестування» в Особистому кабінеті.
4. Здійснити тестування з застосуванням спеціальних програм для тестування кросбраузерності.
5. Здійснити тестування веб-додатків, реалізованих за допомогою Flash Player.
6. Зробити висновки до лабораторної роботи та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Кросбраузерне тестування».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1.

Виконати кросбраузерне тестування веб-сайту <http://prestashop.qatestlab.com.ua/en/>, використовуючи контрольний список (checklist).

Порядок виконання завдання:

1. Провести кросбраузерне тестування сайту <http://prestashop.qatestlab.com.ua/>, використовуючи чекліст, створений та доповнений під час виконання лабораторних робіт «Веб-тестування та чеклісти», «Тестування зручності використання».

2. Перевірити всі пункти контрольного списку на вкладках «Верстка» та «Usability» мінімум в трьох браузерях в останніх або передостанніх версіях (Firefox, Google Chrome, Edge, Opera або інші).
3. Вказати назви та версії браузерів.
4. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на звіти у системі відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> або додати тему багу за принципом «Що? Де? Коли?».
5. Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» додати до відповіді.

Завдання 2.

Занести у систему відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> 5-7 багів, знайдених під час проведення кросбраузерного тестування сайту <http://prestashop.qatestlab.com.ua/en/>.

Теоретичний матеріал

Кросбраузерність – властивість сайту відображатися та працювати у всіх браузерах ідентично. Під ідентичністю розуміється відсутність розвалів верстки та здатність відображати матеріал з однаковим ступенем читабельності. Поняття «кросбраузерність» дуже часто плутають з попиксельною відповідністю, що насправді є різними поняттями. Так як веб-технології весь час розвиваються, прийнятну кросбраузерність можна забезпечити тільки для останніх версій різних браузерів. На практиці зазвичай обмежуються тільки найпопулярнішими браузерами, що суттєво може скоротити час на розробку сайту.

Правила кросбраузерності:

- під час застосування будь-яких змін в дизайні сайту (чи то новий інформер, рекламний блок, новий шаблон або ділянка шаблонів) – необхідно звертати увагу, який має вигляд дизайн в інших браузерах;
- потрібно тестувати сайти самостійно за допомогою встановлених на комп'ютері браузерів, інколи використовуючи спеціальні онлайн сервіси;
- необхідно здійснювати тестування кросбраузерності на всіх доступних пристроях (комп'ютер, ноутбук, планшет та ін.);
- завжди повинен бути встановлений на комп'ютері комплект найбільш популярних браузерів, які не вичерпують ресурси жорсткого диска, але якщо буде потрібно перевірити кросбраузерність сайту – вебмастер запускає їх та проводить необхідний аналіз;

▫ різні версії одного й того ж браузера будуть відображати сайт теж по-різному і це потрібно враховувати, але намагатися використовувати найостанніші версії.

Основні проблеми кросбраузерності сайту лежать на витоках його створення – під час створення шаблону. Існує багато готових шаблонів, з яких можна вибрати той єдиний, який буде однаково (або з незначними змінами) відображатися у всіх браузерах, а потім доопрацювати його під свій проект, але з постійним орієнтиром на різні браузери.

На рис. 5.1 представлені найпопулярніші браузери.



Рис. 5.1. Рейтинг інтернет браузерів на березень 2019 згідно <http://gs.statcounter.com>

Інструменти тестування кросбраузерності верстки:

1. Browsershots, ймовірно, володіє найширшим набором браузерів серед безкоштовних інструментів тестування, включає в себе браузери, що працюють в Linux, Windows та BSD. Серед них є такі, про які взагалі ніколи не чули (наприклад, Galeon, Iceape, Kazehakase або Epirhany). Browsershots дозволяє тестувати як в останніх версіях кожного браузера, так і в застарілих версіях. Хоча Browsershots дозволяє тестувати у величезній кількості браузерів, варто пам'ятати, чим більший набір браузерів для тестування, тим довше доведеться чекати результату. Так що варто зупинитися на основних браузерах.

2. Browser Sandbox – інструмент, який буде корисним тільки користувачам Windows. Він підтримує такі браузери, як Firefox, Chrome, ChromiumCanary, Firefox Mobile, Safari, Opera та FirefoxNightly. В безкоштовному варіанті існує можливість тестування тільки в останній версії певного браузера. Для тестування в старих версіях необхідно користуватися платною версією.

3. Microsoft Edge – це справжня платформа для тестування сайтів в ІЕ. В ньому можна отримати скріншот сайту в багатьох браузерах, в тому числі і мобільних, але на основі платформи Microsoft Edge можна розгорнути віртуальну машину тільки для тестування в ІЕ7 і новіших.

4. CrossBrowserTesting – сервіс, який використовує реальні пристрої для тестування сайту. Сервіс платний, але надає можливість тестувати більш як в 900 браузерах і приблизно в 40 операційних системах. Ще одна його особливість полягає в режимі live testing, в якому можна тестувати сайт в реальному оточенні, перевіряючи дієздатність AJAX, HTML-форм, JavaScript, Flash тощо.

5. Sauce Labs (безкоштовна та комерційна версії) – надає доступ до безлічі браузерів в різних ОС та встановлює з'єднання браузера з налагодженою віртуальною машиною. Також записується відео всієї сесії тестування. Sauce надає 200 хвилин безкоштовного тестування на місяць і дозволяє створювати тести автоматизованого тестування в браузерах (використовується Selenium).

6. Browsera (безкоштовна та комерційна версії) – забезпечує автоматизацію тестування сумісності. Він автоматично визначає відмінності в відображенні сторінок браузерами, тим самим спрощуючи процес тестування. Також визначаються помилки JavaScript, а в комерційній версії дозволяє тестувати сторінки за передплатою та сторінки, що вимагають авторизації. Також можна протестувати динамічні сторінки. Безкоштовна версія включає в себе достатньо обмежене число браузерів та низький дозвіл. Існують різні комерційні версії, які підтримують більшу кількість браузерів, забезпечують високий дозвіл та дозволяють тестувати «закриті» сторінки.

7. BrowserStack – ще один з найвідоміших сервісів для кросбраузерного тестування. Він також надає реальні пристрої для тестування і підтримує більш як 700 браузерів. Існує можливість локального тестування та швидкого отримання скріншотів на різних розширеннях екранів від 800-600 до 2048-1536. Сервіс платний, але для деяких open source проектів пропонуються безкоштовні послуги.

Контрольні запитання

1. Обґрунтувати необхідність та важливість кросбраузерного тестування.
2. За якими чинниками визначають, в яких браузерах проводити тестування?
3. Перерахувати існуючі розширення у різних браузерах для тестування веб-інтерфейсів, вказати доцільність та ефективність застосування.
4. Для чого здійснюється та як впливає на роботу очистка історії (включаючи cookies, cache та ін.) у браузері? Навести приклад роботи браузера до очистки та після.
5. Назвати програму, яка була використана для тестування кросбраузерного тестування у лабораторній роботі, обґрунтувати вибір даної програми.

Список літератури:

1. Инструменты разработчика Firefox [Электронный ресурс]. – Режим доступа: <https://developer.mozilla.org/ru/docs/Tools>.
2. Консоль в браузере Chrome [Электронный ресурс]. – Режим доступа: <https://qaevolution.ru/konsol-v-brauzere-chrome/>.
3. Инструменты разработчика [Электронный ресурс]. – Режим доступа: <https://www.bestfree.ru/article/webdesign/chrome-devtools.php>.
4. Прокачиваем навыки отладки с помощью инструментов разработчика Chrome [Электронный ресурс]. – Режим доступа: <https://css-live.ru/faq/prokachivaem-navyki-otladki-s-pomoshhyu-instrumentov-razrabotchika-chrome-chast-1.html>.
5. Safari: как включить меню «Разработка» [Электронный ресурс]. – Режим доступа: <http://macovod.com.ua/safari-how-to-enable-develop-menu/>.
6. Why Opera isn't planning on going Open Source [Электронный ресурс]. - Режим доступа: <https://operawatch.wordpress.com/2006/10/30/why-opera-isnt-planning-on-going-open-source/>.
7. Кросс-браузерное тестирование: зачем и кому нужно его проводить [Электронный ресурс]. - Режим доступа: <https://training.qatestlab.com/blog/technical-articles/cross-browser-testing/>.
8. Что такое Ad-hoc тестирование? [Электронный ресурс]. - Режим доступа: <https://training.qatestlab.com/blog/technical-articles/what-is-ad-hoc-testing/>.

ЛАБОРАТОРНА РОБОТА №6

ТЕСТУВАННЯ ВЕБ-ПРОЕКТІВ

Мета лабораторної роботи: ознайомлення з основними етапами тестування веб-додатків та функціональними можливостями Інструментів розробника.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Тестування веб-проектів».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Тестування веб-проектів».
3. Виконати завдання до лабораторної роботи «Тестування веб-проектів» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Тестування веб-проектів».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1.

Прикріпити 5 скріншотів з помилкою в консолі Web Developer (F12) при перегляді різних сайтів. У відповіді додати посилання на сторінки сайтів.

Приклад відповіді (рис. 6.1):

Сайт <http://prestashop.qatestlab.com.ua>

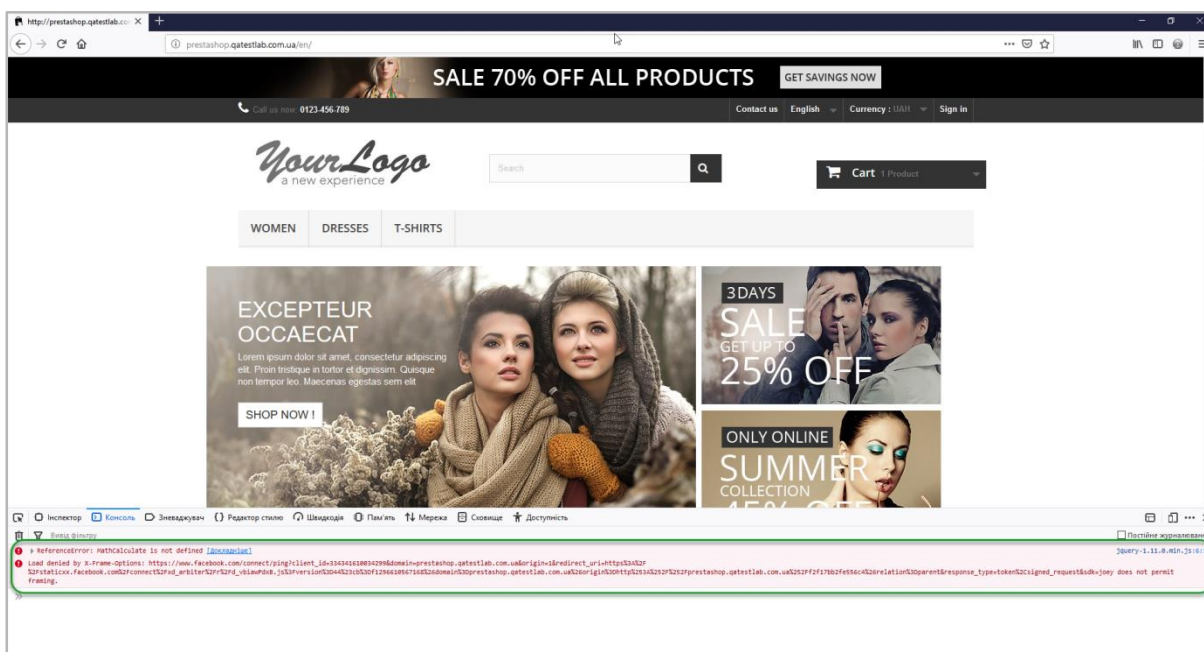


Рис. 6.1. Скріншот з помилкою в консолі Web Developer

Завдання 2.

Дослідити 3 зображення на будь-яких веб-сайтах, визначити розмір цих зображень за допомогою Інструментів розробника. У відповіді додати посилання на сторінки сайтів, вказати розмір зображень, а також прикріпити скріншоти.

Приклад відповіді (рис. 6.2):

Сторінка сайту: <http://prestashop.qatestlab.com.ua/en/tshirts/1-faded-short-sleeve-tshirts.html>, розмір зображення «Faded Short Sleeve T-shirts» – 458 x 458 px.

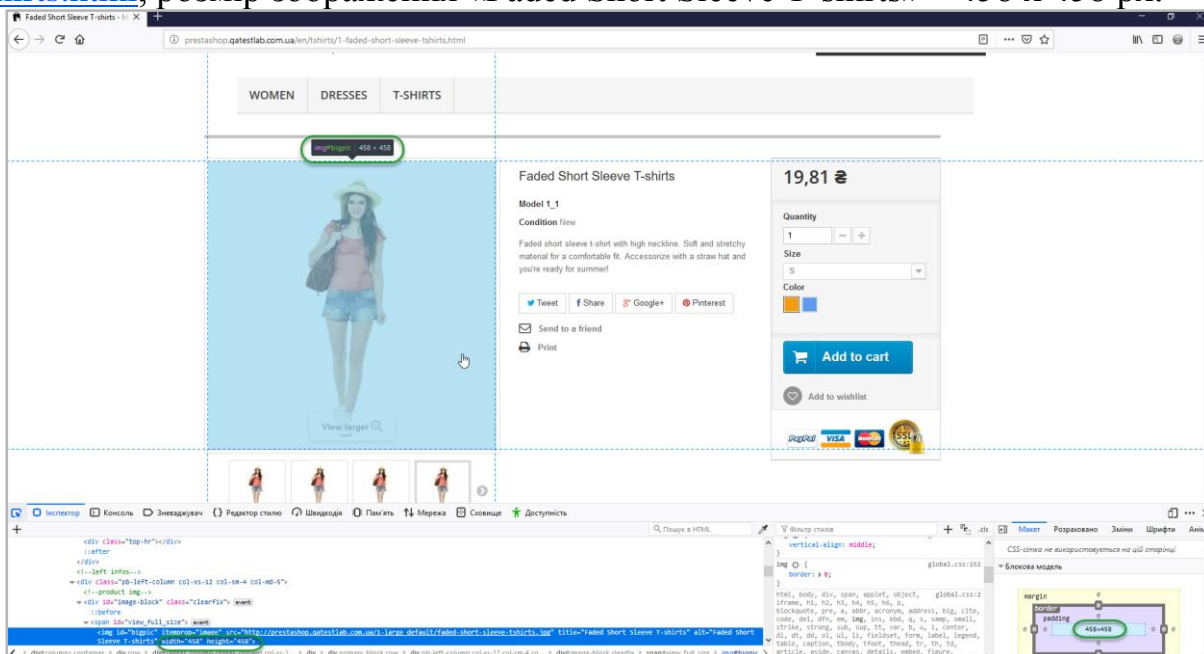


Рис. 6.2. Визначення розміру зображення

Завдання 3.

Змінити властивості шрифту на сторінці будь-якого сайту (колір та розмір), ширину поля. У відповіді додати посилання на сторінки сайтів та прикріпити відповідні скріншоти.

Приклад відповіді (рис. 6.3):

Сторінка сайту: <http://prestashop.gatestlab.com.ua/en/tshirts/1-faded-short-sleeve-tshirts.html>

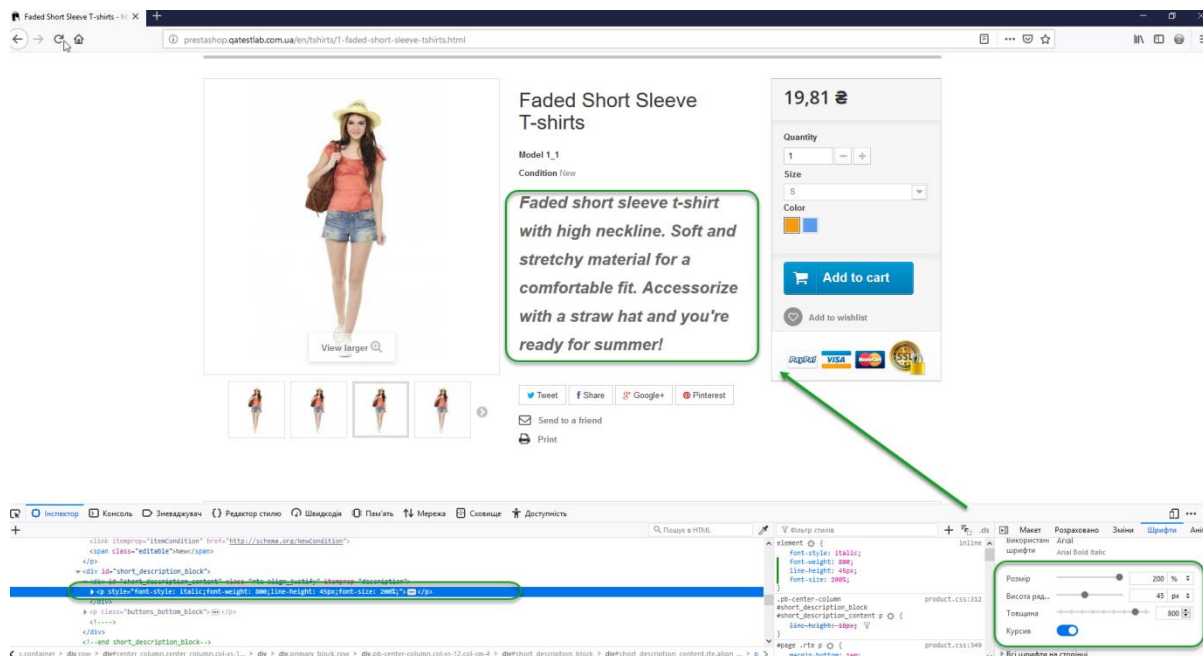


Рис. 6.3. Зміна параметрів сторінки сайту

Теоретичний матеріал

Будь-який веб-додаток складається з клієнтської та серверної частини, де клієнтом виступає браузер, для відображення даних при взаємодії з сервером. Перевага використання веб-додатків в тому, що всі стандартні функції в браузері виконуються незалежно від операційної системи. Проблеми виникають від різної реалізації HTML, CSS, DOM та інших специфікацій в продукті, що призводить до проблем при розробці та підтримці веб-додатків.

Етапи тестування веб-додатків:

1. Підготовчі роботи (вивчення отриманої документації, аналіз функціоналу, технічного завдання, кінцевих макетів сайту, складання тест-плану для подальшого тестування).
2. Тестування верстки веб-сайтів: перевірка розташування елементів, відповідність їх позицій наданим макетам, оптимізація зображень і графіки, перевірка валідності коду та ін.

3. Функціональне тестування – найбільш тривалий етап перевірки ресурсу. Суть цього процесу полягає у перевірці всього описаного функціоналу:

- перевірка роботи всіх обов'язкових функцій сайту;
- інтеграція даних (Data Integration);
- перевірка полів даних (Data field checks);
- перевірка числових полів (Numeric field checks);
- перевірка буквено-цифрових полів (alphanumeric field checks);
- перевірка працездатності форм на сайті (наприклад, зворотній зв'язок, додавання коментаря в блог та ін.);
- перевірка роботи пошуку (включаючи релевантність результатів);
- перевірка гіперпосилань, пошук неробочих посилань;
- перевірка підгрузки файлів на сервер;
- перегляд на відповідність вмісту сторінок сайту вихідного контенту, наданого замовником;
- тестування взаємодії веб-додатків з сервером.

4. Usability тестування: проводиться для оцінки зручності продукту у використанні.

5. Тестування безпеки: перевірка, чи немає у користувачів доступу до службових/закритих сторінок; перевірка захисту всіх критично важливих сторінок (наприклад, розділу адміністрування сайту) від зовнішнього впливу.

6. Тестування продуктивності сайту: проводиться з метою визначення швидкодії сайту або його частини під певним навантаженням. Включає в себе такі види тестування:

- тестування навантаження;
- тестування швидкодії.

7. Обробка результатів. Звіт про виявлені в процесі тестування помилки передається учасникам проекту, після чого керівник проекту визначає відповідального за виправлення кожної з помилок. Далі визначається графік виправлення помилок, після чого проводиться повторне тестування з метою контролю якості виправлення помилок, а також відсутності нових. Дана процедура повторюється, поки сайт не буде відповідати специфікаціям технічного завдання.

Інструменти розробника – це потужний інструмент для налагодження коду веб-сайтів, який за замовчуванням встановлений в браузерах Firefox, Google Chrome та в інших браузерах. Крім веб-розробки, даний інструментарій, а також накопичені знання з HTML, CSS будуть дуже корисні тестувальникам при тестуванні веб-сайтів і створенні

баг-репортів. Грамотне оформлення баг-репортів, використання правильної термінології (назви елементів, стильових властивостей, помилок, які відображаються і т.д.), а також знаходження причини дефекту, допоможе розробникам швидко виправити дефект.

Основні функції Інструментів розробника:

- перегляд HTML, зміна стилів і верстки в режимі реального часу;
- перегляд CSS-метрик;
- аналіз використання та продуктивності мережі з великою точністю;
- налагодження та профілювання JavaScript;
- зручна консоль, яка надає детальну і корисну інформацію про помилки в JavaScript, CSS та XML;
- набір функцій логування сценаріїв JavaScript;
- інструменти для перегляду DOM сайту і багато-багато іншого.

Для того, щоб відкрити Dev Tools можна:

- використати гарячі клавіші: ***Ctrl + Shift + I*** (Windows/Linux) або ***Command + Option + I*** (Mac);
- вибрати в браузері ***Firefox*** пункт меню ***Веб-розробка => Перемкнути інструменти***; в браузері ***Google Chrome*** вибрати пункт меню ***Додаткові інструменти => Інструменти розробника***;
- натиснути клавішу ***F12***.

Контрольні запитання

1. Яким чином можна оцінити проблемні частини сайту за допомогою Dev Tools? Відповідь обґрунтувати та навести приклади.
2. За допомогою якого програмного інтерфейсу можливо змінити «JavaScript» сценарії у реальному часі на сайті?
3. Як визначити за допомогою Dev Tools, коли почалося та закінчилося завантаження файлів на сайті? Відповідь обґрунтувати та навести приклади.
4. В якій послідовності завантажується «JavaScript»? Відповідь обґрунтувати та навести приклади.
5. Перерахувати, які помилки на сайті можна знаходити за допомогою інструментів розробника, навести приклади.

Список літератури:

1. Особенности тестирования веб-приложений [Электронный ресурс]. – Режим доступа: <https://quality-lab.ru/key-principles-of-web-testing/>.
2. Web Application Testing Complete Guide (How to Test a Website) [Электронный ресурс]. – Режим доступа: <http://www.softwaretestinghelp.com/web-application-testing/>.
3. Особенности тестирования web-приложений [Электронный ресурс]. – Режим доступа: <https://qaevolution.ru/osobennosti-testirovaniya-web-prilozhenij/>.
4. Website Cookie Testing & Test Cases for Testing Web Application Cookies [Электронный ресурс]. – Режим доступа: <https://www.softwaretestinghelp.com/website-cookie-testing-test-cases/twaretestinghelp.com/category/cookie-testing>.
5. Позитивное и негативное тестирование [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/positive-negative-testing/>.
6. Обзор видов тестирования [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/review-the-types-of-testing/>.

ЛАБОРАТОРНА РОБОТА №7

ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ

Мета лабораторної роботи: ознайомлення з функціональним тестуванням та отримання базових теоретичних та практичних навичок в створенні тестових випадків (*test case*) для функціонального тестування.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Функціональне тестування».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Функціональне тестування».
3. Виконати завдання до лабораторної роботи «Функціональне тестування» в Особистому кабінеті.
4. Зробити висновки по лабораторній роботі та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Функціональне тестування».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1.

Виконати функціональне тестування веб-сайту

<http://prestashop.qatestlab.com.ua/en/>, використовуючи контрольний список (checklist).

Порядок виконання завдання:

1. Завантажити приклад чекліста в Особистому кабінеті у завданні до лабораторної роботи «Функціональне тестування» (рис. 7.1).
2. Вказати прізвище та ім'я власника у назві контрольного списку.
3. Додати мінімум 5-7 пунктів до існуючих пунктів контрольного списку на вкладці «Функціонал», не повторюючись з вже доданими, та відмітити їх будь-яким кольором.

4. Провести функціональне тестування сайту

<http://prestashop.qatestlab.com.ua/en/>, перевіривши всі пункти контрольного списку на вкладці «Функціонал» мінімум в 3х браузерах (Firefox, Google Chrome, Edge, Opera або інший) в останніх або передостанніх версіях.

5. Вказати назву та версію браузера. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на звіти у системі відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> або додати тему багу за принципом «Що? Де? Коли?».

6. Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» додати до відповіді.

Завдання 2.

Занести у систему відслідковування помилок Mantis Bug Tracker

<http://mantis.qatestlab.net> 5-7 багів, знайдених під час проведення функціонального тестування сайту <http://prestashop.qatestlab.com.ua/en/>.

1	Сайт http://opencart.qatestlab.net	Google Chrome 74.0.3729.131	Firefox 66.0.4	Edge 44.17763.1.0	Примітки
2	Реєстрація та Особистий профіль				
3	Перевірити реєстрацію на сайті	Passed	Passed	Passed	
4	Перевірити редагування профілю	Failed	Failed	Failed	http://bt-w.qatestlab.com/view.php?id=...
5	Форма зворотнього зв'язку				
6	Перевірити валідацію полів				
7	Перевірити відправку листа / повідомлення				
8	Перевірити доставку листа / повідомлення				
9	Пошук				
10	Перевірити роботу пошуку за назвами				
11	Перевірити перехід за посиланнями				
12	Перевірити роботу пошуку по різноманітним параметрам				
13	Коментарі				
14	Перевірити додавання коментаря				
15	Перевірити коректне відображення коментаря				
16	Шаринг + фото				
17	Перевірити функцію шаринга фото у всіх запропонованих соцмережах, відправка ел. поштою і т.п.				
18	Перевірити функцію перегортання фото				
19	Перевірити функцію збільшення фото (якщо є)				
20	Перевірити функцію додавання, відображення, зміни / видалення з обраного (якщо є)				
21	Перевірити коректне відображення списку фотографій і самих фото				
22	Сторінка товару				
23	Перевірити чи містить заголовок сторінки назву товару				
24	Перевірити коректне відображення інформації про товар				
25	Перевірити чи можливе додавання товару в корзину				
26	Перевірити чи виводиться візуальне підтвердження після додавання товару в корзину				

Рис. 7.1. Приклад контрольного списку для функціонального тестування

Теоретичний матеріал

Функціональне тестування (functional testing) – вид тестування, при якому виявляється некоректна/неправильна робота функціоналу програми або процес перевірки відповідності поведінки системи першочергово заявленим функціональним вимогам.

Також, можна сказати, що **функціональне тестування** – це тестування програмного забезпечення з метою перевірки реалізованості функціональних вимог, тобто здатності програмного забезпечення в певних умовах вирішувати завдання, потрібні користувачам. Функціональні вимоги визначають, що саме робить програмне забезпечення, які завдання вирішує.

При функціональному тестуванні перевіряється коректність роботи системи, яка включає перевірку кожної з функцій програми, адекватність збережених і вихідних даних, методи їх обробки, обробка даних, що вводяться, методи зберігання даних, методи імпорту та експорту даних і т.д. залежно від специфіки додатку. При перевірці проектів проводиться документація функціональних вимог, що спрощує перевірку.

При функціональному тестуванні проводиться перевірка наступних модулів сайту:

1. Реєстрація (registration, logging in);
2. Авторизація (authorization);
3. Новинний модуль (news);
4. Пошук по сайту (search);
5. Зворотній зв'язок (feedback);
6. Банери (banners);
7. Фотоальбоми (photo album);
8. Форум (forum);
9. Інтернет-магазин (online shop, e-shop);
10. Список, що випадає (drop-down list);
11. Коментарі, поширення в соціальних мережах (sharing);
12. Відео (video);
13. Модуль розсилки (mailout module);
14. Форми (forms).

Під час роботи з даними важливу роль відіграє валідація даних (*data validation*). Перш ніж використовувати отримані від користувача дані, необхідно переконатися, що вони введені правильно і показують коректні значення.

Валідація (validation) – це процес перевірки даних на відповідність певним, заздалегідь відомим правилам (форматам, вимогам).

Під час тестування необхідно перевіряти узгодженість валідаторів вхідних даних з логікою обробки цих даних додатком. Для тестування затвердження даних, треба розуміти, як вони повинні працювати, що можна вважати правильним, а що ні.

«Невалідні» дані, що не задовольняють певним обмеженням, можуть викликати збій у роботі програми.

Валідація даних здійснюється наступним чином:

1. Посимвольна перевірка.
2. Перевірка окремих значень.
3. Сукупність вхідних значень.
4. Перевірка стану системи після обробки даних.

Контрольні запитання

1. Що відносять до дефектів функціонального тестування? Навести приклади та обґрунтувати відповідь.
2. За яким принципом були додані нові пункти до чекліста? Обґрунтувати важливість нових пунктів при тестуванні сайту <http://prestashop.qatestlab.com.ua/en/>.
3. Які основні пункти повинен містити контрольний список (*checklist*) з тестування функціоналу сайту?
4. Надати пояснення твердженню, що функціональне тестування є трудомістким процесом та може займати до 80% всього бюджету проекту з тестування.
5. Тестування яких модулів сайту <http://prestashop.qatestlab.com.ua/en/> здійснювалось в першу чергу при функціональному тестуванні, навести приклади та обґрунтувати відповідь.

Список літератури:

1. Тестирование программного обеспечения – основные понятия и определения [Электронный ресурс]. – Режим доступа: <http://www.znannya.org/?view=software-testing-testing>.
2. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапе. - М.: Дело, 2007. - 312 с.
3. Система отслеживания ошибок [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Система_отслеживания_ошибок.
4. Техника написания хороших коротких Summary баг-репортов [Электронный ресурс]. – Режим доступа: <http://svyatoslav.biz/education/bug-reports-summary/>.
5. Баг-репорты [Электронный ресурс]. – Режим доступа: <http://bugscatcher.net/archives/3307>.
6. История развития тестирования программного обеспечения. [Электронный ресурс]. – Режим доступа: <https://social.msdn.microsoft.com/Forums/ru-RU/531410fb-fc1e-4f1d-bb9c-0804970bb949/-?forum=fordsktopru>.
7. Профессия тестировщик [Электронный ресурс]. – Режим доступа: <http://enjoy-job.ru/professions/testirovschik/>.
8. Что такое функциональный баг и как его найти [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/what-is-a-functional-bug-and-how-to-find-it/>.
9. Как тестировать формы [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/test-contact-form/>.

ЛАБОРАТОРНА РОБОТА №8

ТЕХНІЧНЕ ТЕСТУВАННЯ

Мета лабораторної роботи: знайомство з основними програмами для технічного тестування CMS системи: *Joomla*, *Drupal*. Отримання базових теоретичних та практичних навичок в створенні тестових сайтів, використовуючи *Joomla* або *Drupal* системи.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Технічне тестування».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Технічне тестування».
3. Виконати завдання до лабораторної роботи «Технічне тестування» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Технічне тестування».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Створити тестовий сайт на Joomla або Drupal.

Тема сайту – довільна. Сайт повинен містити від 5 до 10 сторінок, кожна сторінка – це один розділ (наприклад: Головна, Новини, Фотогалерея і т.д.), кожен розділ може містити різний контент, картинки, текст, а також різні функціональні модулі.

Порядок виконання завдання:

Створення сайту на Joomla:

1. Встановити веб-сервер, наприклад: *Denwer*.

З порядком установки можна ознайомитися за посиланнями:

<http://bit.ly/2JurJbu>

<http://bit.ly/2JurJbu>

2. Завантажити **Joomla** з сайту <http://www.joomla.org/>
- 3 порядком установки можна ознайомитися за посиланням:
<http://bit.ly/2VBmrfN>
3. Ознайомитися з адміністративною частиною **Joomla** за посиланням:
<http://bit.ly/2Wf30xJ>
4. Створити власний сайт.

Створення сайту на Drupal:

1. Встановити веб-сервер, наприклад: **Denwer**.
- 3 порядком установки можна ознайомитися за посиланням:
<http://bit.ly/2Wg9Ra4>
2. Завантажити **Drupal** з сайту <https://www.drupal.org/download>.
- 3 порядком установки можна ознайомитися за посиланням:
<http://bit.ly/2w8HpZ5>
3. Ознайомитися з адміністративною частиною **Drupal** за посиланням:
<http://bit.ly/2WmLBmZ>
4. Створити власний сайт.

Приклад відповіді:

Фронт (сайт): посилання на сайт <https://...>

Адміністративна частина: посилання на адміністративну частину сайту <https://...>

Логін: *admin*

Пароль: *admin*

Додати короткий опис створеного сайту для попереднього ознайомлення. Описати який шаблон використовувався, яка кількість сторінок, розділів та інше.

Теоретичний матеріал

Технічне тестування – група видів тестування, що проводиться на завершальному етапі готовності програмного продукту (сайту) й спрямована на перевірку технічних характеристик, таких як:

- здатність сервера витримувати навантаження;
- швидкість завантаження сторінок;
- трасовість (такі показники як вік, PR);
- цілісність посилань;
- індексація сторінок, наявність в каталогах пошукових машин;
- мета-теги, HTML-теги;
- відповідність стандартам W3C і багато іншого.

Основні програми для технічного тестування:

1. ***Інструменти розробника (Developer Tools)*** – це потужний інструмент для налагодження коду веб-сайтів, який за замовчуванням встановлений в браузерах Firefox, Google Chrome та в інших браузерах. Крім веб-розробки, даний інструментарій, а також накопичені знання з HTML, CSS будуть дуже корисні тестувальникам при тестуванні веб-сайтів і створенні баг-репортів. Визначити колір, погратися зі шрифтами, виміряти піксельну позицію елементу на сторінці або протестувати адаптивність сайту для різних пристроїв – тепер все це можна зробити не встановлюючи окремих додатків.

До основних можливостей Інструментів розробника відносять:

Після виконання деяких операцій на веб-сайті, завжди корисно

- ***зручний перегляд HTML-коду сторінки:*** функція Inspect дозволяє точно визначити місцезнаходження тега того чи іншого елемента, переглянути всі «прив'язані» до нього властивості та стилі;
- ***редагування HTML та CSS безпосередньо в браузері:*** можна змінювати атрибути тегів та значення властивостей для того, щоб спостерігати зміни. Це зручно для тих випадків, коли потрібно шляхом експериментів знайти найбільш прийнятний варіант оформлення створеної сторінки;
- ***налагодження JavaScript;***
- ***відстеження процесу завантаження сторінки.*** Існує можливість урізати швидкість з'єднання, щоб подивитися як буде поводити себе контент сторінки при повільному інтернеті або коли взагалі не буде інтернету;
- ***перегляд HTTP-заголовків звичайних та AJAX-запитів;***
- ***редагування або видалення cookie.*** Вкладка Sources;
- ***зміна даних геолокації*** – це досить зручно при тестуванні різних сервісів з урахуванням карт і маршрутів. У цьому режимі можна й провести тестування юзабіліті мобільної версії сайту. А підтримка емуляції відразу пари десятків пристроїв на адаптивність, зробить перевірку ще більш надійною.

2. ***Xenu Link Sleuth*** – це один з найбільш корисних інструментів в пошуковій оптимізації.

До основних завдань, які виконує програма відносять:

- ***шукати биті (непрацюючі) посилання на заданому ресурсі:*** своєчасне виявлення несправних посилань дозволяє не тільки поліпшити

показники сайту в пошукових системах, але й вочевидь підвищити інтерес та лояльність користувачів сайту;

- **складати карту сайту:** для динамічних сайтів скласти карту не складає проблеми, однак, для статичних HTML ресурсів створювати карту сайту вручну вельми довго й трудомістко. Хепи вирішує цю задачу за кілька хвилин в залежності від розміру сайту та швидкості інтернет-з'єднання;

- **шукати сторінки з великим часом віддачі:** вимірювання швидкості завантаження веб-сторінок – дуже важливий процес тестування. Але заміряти вручну кожен сторінку багатосторінкового сайту занадто затратно по ресурсам. За допомогою Хепи можна побачити картину відгуку всіх сторінок в цілому та визначити проблемні місця;

- **знаходження неунікальних заголовків:** кожен заголовок на сторінці повинен бути унікальним, тоді жоден з них не буде знаходитися в додаткових результатах пошуку та фільтруватися, як дубльований контент.

- **знаходження сторінок з великим рівнем вкладеності:** всі сторінки на сайті по можливості повинні знаходитися не далі, ніж у двох-трьох рівнях від головної. Чим далі знаходиться сторінка, тим складніше до неї добратися як користувачам, так і пошуковим системам. Якщо знайшлися подібні сторінки, які являються достатньо важливими, але знаходяться далі, ніж в 3-х рівнях від головної, варто прийняти які-небудь заходи для поліпшення навігації;

- **виявляти, які зі сторінок мають найбільшу та найменшу кількість внутрішніх посилань:** перевірити внутрішню перелінковку в числовому вигляді. Які з сторінок заслужили більше уваги, а які менше (виходячи з внутрішніх посилань);

- **знаходження картинок з відсутнім атрибутом «alt»:** атрибут «alt» є важливим при оптимізації сайту або окремих сторінок під певні запити. Перевірити, можливо відсутня важлива інформація чи опис зображень на вашому сайті.

Системи управління контентом (CMS – Content Management System)

CMS – інформаційна система або комп'ютерна програма, яка використовується для забезпечення та організації спільного процесу створення, редагування та управління контентом.

Основні функції CMS:

- надання інструментів для створення наповнення (даних), організації спільної роботи над наповненням;

- управління даними: зберігання, контроль версій, дотримання режиму доступу, управління потоком документів та ін.;
- публікація наповнення;
- представлення інформації у вигляді, зручному для навігації, пошуку.

Joomla! – система управління даними (CMS), написана на мовах PHP і JavaScript, що використовується в якості сховища бази даних СУБД MySQL або іншої індустріально-стандартної реляційної СУБД. Є вільним програмним забезпеченням, що поширюється під ліцензією GNU GPL.

CMS Joomla! включає в себе мінімальний набір інструментів при початковій установці, який доповнюється в міру необхідності. Це знижує заповнення адміністративної панелі непотрібними елементами, а також знижує навантаження на сервер і економить місце на хостингу.

Drupal – система управління вмістом, що також використовується як каркас для веб-додатків (CMF). Система написана на мові PHP та використовує як сховище даних реляційну базу даних (підтримуються MySQL, PostgreSQL та інші). Drupal є вільним програмним забезпеченням, захищеним ліцензією GPL, й розвивається зусиллями ентузіастів зі всього світу.

Контрольні запитання

1. Вказати, які існують особливості при установці **Joomla** та **Drupal**.
2. Обґрунтувати необхідність в локальних веб-серверах. Навести приклади використання на практиці.
3. Визначити основні властивості та відмінності найбільш розповсюджених систем управління контентом. Відповідь обґрунтувати та навести приклади використання на практиці.
4. За допомогою якої програми можна визначити «биті посилання» на сайти? Навести приклад процесу визначення «битих посилань».
5. Перерахувати особливості технічного тестування.

Список літератури:

1. Консоль в браузере Chrome [Электронный ресурс]. – Режим доступа: <https://qaevolution.ru/konsol-v-brauzere-chrome/>.
2. Проводим аудит внутренней структуры сайта программой Xenu Link Sleuth [Электронный ресурс]. – Режим доступа: <https://devaka.ru/articles/xenu-link-sleuth>.
3. SEO: XENU - бесплатный аудит сайта и мертвых ссылок [Электронный ресурс]. – Режим доступа: <https://elims.org.ua/blog/xenu-audit-sajta-i-mertvyx-ssylok/>.

ЛАБОРАТОРНА РОБОТА №9

ВИДИ ТЕСТУВАННЯ, ПОВ'ЯЗАНІ ЗІ ЗМІНАМИ

Мета лабораторної роботи: ознайомлення з поняттями регресійного тестування (Regression Testing), димового тестування (Smoke Testing), санітарного тестування (Sanity Testing), тестування збірки (Build Verification Test), отримання базових теоретичних та практичних навичок в проведенні тестування, пов'язаного зі змінами.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Види тестування, пов'язані зі змінами».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Види тестування, пов'язані зі змінами».
3. Виконати завдання до лабораторної роботи «Види тестування, пов'язані зі змінами» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Види тестування, пов'язані зі змінами».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

1. Зробити вибірку дефектів у проєкті «Bug regression_new» в баг-трекері <http://mantis.qatestlab.net/> за такими параметрами:
 - **Reporter:** Any
 - **Date:** 13.09.2016 - 26.09.2016
 - **Severity:** major
 - **Resolution:** fixed
2. Зробити регресійне тестування дефектів, які були знайдені після вибірки, на сайті <http://prestashop.qatestlab.com.ua/en/>

3. У відповіді до завдання вказати посилання на звіти про дефекти, розділивши дефекти на дві категорії: виправлені та не виправлені.

Теоретичний матеріал

Для підтвердження того факту, що баг/дефект був дійсно виправлений, після внесення необхідних змін, програмне забезпечення повинно бути протестоване ще раз. Нижче перераховані види тестування, які необхідно провести для підтвердження працездатності програми або правильності здійсненого виправлення дефекту:

- Регресійне тестування (Regression Testing).
- Димове тестування (Smoke Testing).
- Санітарне тестування або перевірка узгодженості/справності (Sanity Testing).
- Тестування збірки (Build Verification Test).

Регресійне тестування (Regression Testing) – у більшості випадків, розглядають, як тестування, яке має на меті переконатися, що нова зміна коду в певному місці програми не вплинула негативно на інші доти працюючі частини програми. Регресійними можуть бути як функціональні, так і нефункціональні тести. Як правило, для регресійного тестування використовуються тест-кейси, написані на ранніх стадіях розробки і тестування. Це дає гарантію того, що зміни в новій версії програми не пошкодили вже існуючу функціональність. Рекомендується робити автоматизацію регресійних тестів, для прискорення подальшого процесу тестування і виявлення дефектів на ранніх стадіях розробки програмного забезпечення.

Залежно від контексту використання терміну регресійного тестування, може мати різний зміст. Сем Канер, наприклад, описав три основних типи регресійного тестування:

- **Регресія багів (Bug regression)** – спроба довести, що виправлена помилка насправді не виправлена.
- **Регресія старих багів (Old bugs regression)** – спроба довести, що нещодавня зміна коду або даних зламала виправлення старих помилок, тобто старі баги стали знову відтворюватися.
- **Регресія побічного ефекту (Side effect regression)** – спроба довести, що нещодавня зміна коду або даних зламала інші частини продукту.

Димове тестування (Smoke testing). Назва цього виду тестування походить від швидкого способу перевірки інженерами електроприладів:

при введенні в експлуатацію нового обладнання вважалося, що тестування пройшло вдало, якщо з установки не пішов дим.

В області тестування програмного забезпечення, димове тестування застосовується для поверхневої перевірки всіх модулів програми на предмет працездатності і наявності швидкого знаходження критичних і блокуючих дефектів. Мета такого тестування перевірити, що після чергової збірки програмного продукту немає явних грубих помилок.

Smoke тести повинні бути швидкими та легкими для того, щоб їх можна було запускати часто.

Санітарне тестування або перевірка узгодженості/справності (Sanity Testing) – це вузьконаправлене тестування, що є достатнім для доказу того, що конкретна функція працює згідно заявленим в специфікації вимогам. Використовується для визначення працездатності лише певної частини програми після внесення деяких незначних змін. Зазвичай виконується вручну.

На відміну від димового (*Smoke testing*), санітарне тестування (*Sanity testing*) направлено вглиб функції, що перевіряється, в той час як димове направлено в ширину, для покриття тестами якомога більшої кількості функціоналу в найкоротші терміни.

Тестування збірки (Build Verification Test) – тестування спрямоване на визначення відповідності, випущеної версії, критеріям якості для початку тестування. За своїми цілями є аналогом димового тестування, спрямованого на прийняття нової версії в подальше тестування або експлуатацію. Вглиб воно може проникати далі, залежно від вимог до якості випущеної версії.

Контрольні запитання

1. Обґрунтувати важливість та необхідність регресійного тестування, навести приклади.
2. Вказати можливу причину не виправлення знайдених дефектів, надати обґрунтовану відповідь.
3. В чому полягає різниця між регресією багів та регресією старих багів? Відповідь обґрунтувати та навести приклади.
4. Яка документація використовується для проведення регресійного тестування?
5. В чому полягає відмінність санітарного тестування від димового?

Список літератури:

1. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапы. - М.: Дело, 2007. - 312 с.
2. Тестирование программного обеспечения: понятие и определение [Электронный ресурс]. – Режим доступа: <http://www.znannya.org/?view=software-testing-testing>.
3. Профессия менеджер [Электронный ресурс]. – Режим доступа: <http://enjoy-job.ru/professions/testirovschik/>.
4. Web testing [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/Web_testing.
5. Тестирование программного обеспечения [Электронный ресурс]. – Режим доступа: <http://www.protesting.ru>.
6. Краткие основы тестирования ПО // Коробейник А.Н. - М., 2012.

ЛАБОРАТОРНА РОБОТА №10

ТЕСТ-ДИЗАЙН ТА ТЕСТ-КЕЙСИ

Мета лабораторної роботи: ознайомлення з техніками тест-дизайну, отримання базових теоретичних та практичних навичок в групуванні тест-кейсів.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Тест-дизайн, тест-кейси».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Тест-дизайн, тест-кейси».
3. Виконати завдання до лабораторної роботи «Тест-дизайн, тест-кейси» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Тест-дизайн, тест-кейси».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1. Згрупувати тест-кейси за призначенням:

- 1) позитивні тестові випадки:
 - а) перевірка функціоналу;
 - б) перевірка дизайну/UI;
- 2) негативні тестові випадки.

Порядок виконання завдання:

1. Завантажити файл з прикладами тест-кейсів в Особистому кабінеті у завданні до лабораторної роботи «Тест-дизайн, тест-кейси».
2. Згрупувати тест-кейси за призначенням та вказати ID-номери тест-кейсів за групами у відповіді.

Завдання 2. У системі TestLink <http://tl-univer.qa-testlab.net> створити тест-кейси для позитивного та негативного тестування на сайті <http://prestashop.qatestlab.com.ua/en/> для наступних форм:

- форма реєстрації користувача;
- форма авторизації нового користувача;
- форма відновлення паролю до онлайн-акаунту.

У відповіді до завдання вказати посилання на створені тест-кейси.

Теоретичний матеріал

Тест-дизайн (test design) – це етап процесу тестування програмного забезпечення (ПЗ), на якому проектуються і створюються тестові випадки (тест-кейси), відповідно з певними раніше визначеними критеріями якості і цілями тестування.

Тестовий випадок або тест-кейс (test case) – це сукупність кроків, конкретних умов і параметрів, необхідних для перевірки реалізації функції, що тестується, або її частини.

Тестовий набір (test suite):

- набір тестів, що реалізують бізнес-завдання, виконуване системою, що тестується;
- тестовий набір включає тестові сценарії, тестові дані або правила їх генерації.

До стандартних атрибутів тест-кейса відносять:

- **ID (Test Case ID)** – унікальний ідентифікатор необхідний для зручної організації зберігання і навігації у групі тестів (Test Suite), переважно генерується автоматично;
- **назва** – основна тема чи ідея тест-кейса, короткий опис його суті, який дозволяє зрозуміти призначення тест-кейсу. Може повторювати тему. Поле обов'язкове для заповнення;
- **тема (Summary)** – поле опису основної теми, яка лаконічно і точно описує його суть. Для уніфікованого підходу до викладу теми використовують конструкцію, яка відповідає на питання «Що перевіряємо? Де перевіряємо? З якими даними?». Поле обов'язково для заповнення;

- **попередні умови (Preconditions)** – список всіх необхідних підготовчих дій (налаштування програми, середовища тестування) для виконання даного тест-кейсу;
- **кроки для відтворення (Step actions)** – описують послідовність дій для відтворення тестового випадку, які повинні привести до очікуваного результату. Повинні бути короткими і зрозумілими;
- **очікувані результати (Expected results)** – визначають правильну реакцію програми на виконання даних кроків. Повинні бути зрозумілими, однозначними, простими;
- **історія редагування** – лаконічний журнал змін, де відображено ким, як і коли був змінений тест-кейс;
- **прикріплені файли (Attached files)** – може бути доданий відповідний файл для тестування або файл з тестовими акаунтами.

Тестові випадки поділяються за очікуваним результатом на **позитивні** та **негативні**.

Позитивний тест-кейс використовує тільки валідні дані і перевіряє, що додаток правильно виконав функцію, що викликається.

Негативний тест-кейс оперує як валідними, так і невалідними даними (мінімум 1 некоректний параметр) і ставить за мету перевірку виняткових ситуацій (спрацьовування валідаторів).

Тестові випадки зручно об'єднувати **за призначенням**:

1. позитивні кейси:
 - перевірка функціоналу;
 - перевірка дизайну/UI;
 - перевірка безпеки.
2. негативні кейси.

При складанні тест-кейсів необхідно дотримуватися наступних принципів:

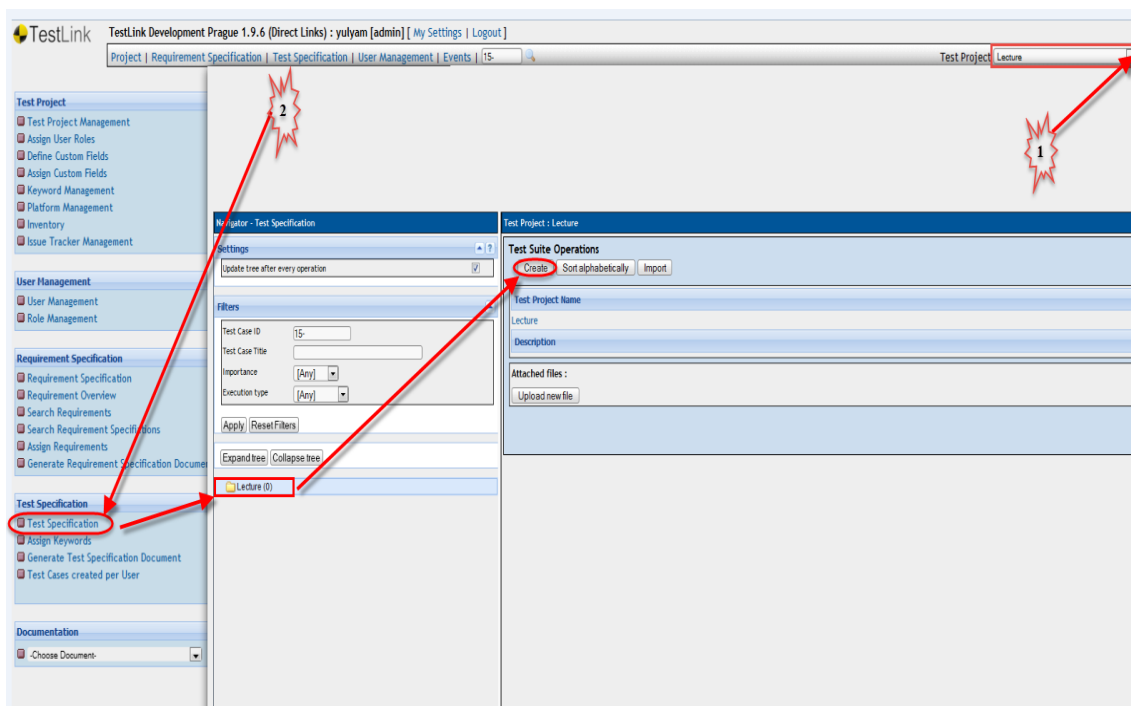
- тему тест-кейса необхідно описувати за принципом «Що перевіряється? Де перевіряється? З яким типом даних (валідні, невалідні)?»;
- наявності опису тест-кейса;
- попередніх умов, де має бути вказано, яка форма, сторінка і т. д. відкрита;
- на кожен крок повинен бути очікуваний результат;

- в кроках має бути зазначений тип даних, що вводяться (валідні/ невалідні), у разі невалідних – повинні бути наведені приклади таких даних (наприклад, для поля введення електронної пошти невалідними будуть неприпустимі спеціальні символи, без @, без домена, та ін.);
- тест-кейс повинен бути завершеним і цілісним (наприклад, функція відновлення пароля повинна закінчуватися на успішній зміні пароля, а не на відправці листа з посиланням для зміни пароля).

TestLink – система управління тестами з веб-інтерфейсом, яка дозволяє імпортувати вимоги, генерувати на їх основі тестові сценарії або створювати тестові сценарії з нуля, генерувати тестові специфікації, звіти про тестування, призначати виконання тестів на фахівців з контролю якості, відстежувати їх активність в реальному часі, пов'язувати не пройдені тести з баг-трекінговою системою і збирати різного роду статистику.

Для того, щоб створити тестовий набір (test suite) необхідно:

- вибрати тестовий проект в випадаючому списку «*Test Project*»;
- натиснути на пункт «*Test Specification*» в меню «*Test Specification*»;
- натиснути на папку з ім'ям проекту в діалоговому вікні «*Navigator - Test Specification*»;
- натиснути кнопку «*Create*» в діалоговому вікні «*Test Project: <Name Project>*»;
- присвоїти ім'я тестового набору в полі «*Test Suite Name*»;
- натиснути кнопку «*Create Test Suite*».

Рис.1. Створення тестового набору (*test suite*)

Для створення тестового випадку (*test case*) необхідно:

- натиснути на папку з ім'ям тестового набору в діалоговому вікні «*Navigator - Test Specification*»;
- натиснути кнопку «*Create*» в пункті «*Test Case Operations*»;
- в формі, що з'явилася, заповнити всі атрибути тест-кейсу;
- натиснути кнопку «*Create*».

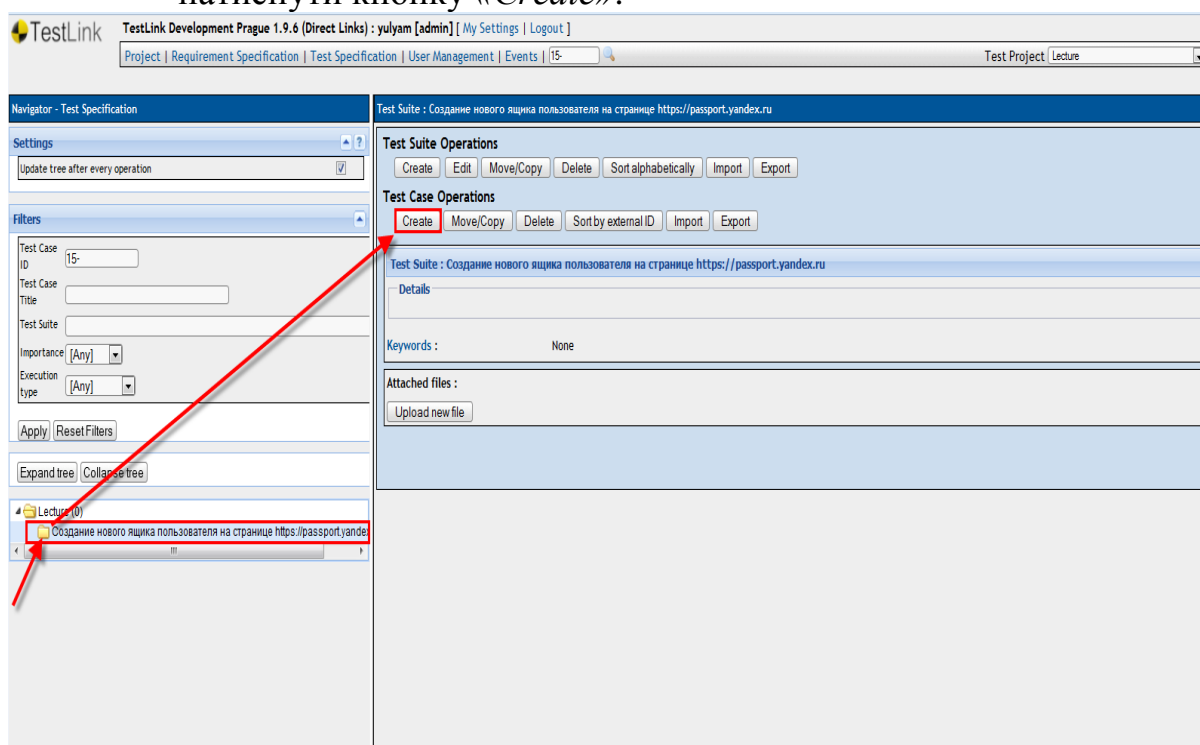


Рис.2. Створення тест-кейсу

Після створення тест-кейсу створюють кроки відтворення:

- натиснути на тестовий випадок в діалоговому вікні «Navigator - Test Specification»;
- натиснути кнопку «Create step»;
- заповнити поля «Step actions» і «Expected Results»;
- натиснути на кнопку «Save».

Редагування тест-кейсу (test case):

- натиснути на тестовий випадок в діалоговому вікні «Navigator - Test Specification»;
- натиснути кнопку «Edit»;
- внести необхідні зміни;
- натиснути на кнопку «Save».

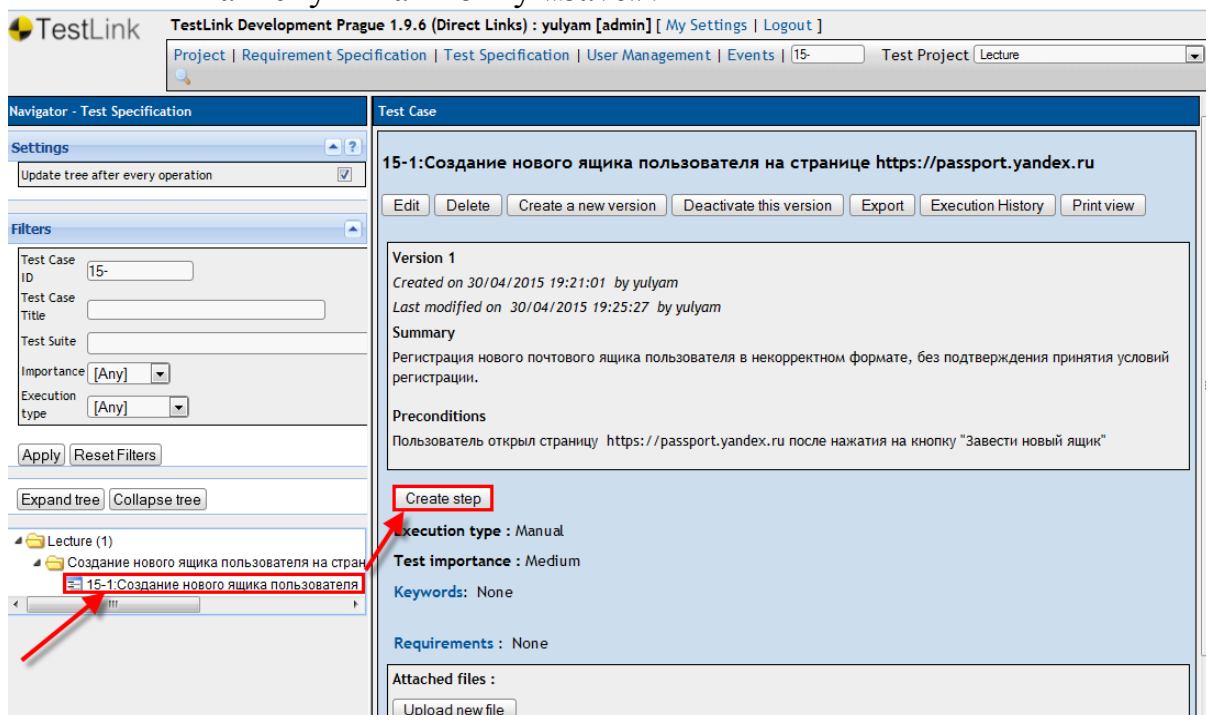


Рис.3. Створення кроків відтворення тестового випадку (test case)

Контрольні запитання

1. Дайте визначення поняттю тест-кейс та вкажіть його основні атрибути.
2. Вказати причини того, чому спочатку проводиться позитивне тестування, а потім негативне.
3. В чому полягає важливість використання тестових випадків у тестуванні ПЗ?
4. Вказати переваги та недоліки використання тестових випадків у тестуванні ПЗ на прикладі.
5. В чому полягає різниця між тест-кейсом та звітом про дефект?

Список літератури:

1. Тестирование Дот Ком, или Пособие по жестокому обращению с багами в интернет-стартапе. - М.: Дело, 2007. - 312 с.
2. Система отслеживания ошибок [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Система_отслеживания_ошибок.
3. Баг-репорты [Электронный ресурс]. – Режим доступа: <http://bugscatcher.net/archives/3307>.
4. Web testing [Электронный ресурс]. – Режим доступа: http://en.wikipedia.org/wiki/Web_testing.
5. Особенности тестирования веб-приложений [Электронный ресурс]. – Режим доступа: <https://quality-lab.ru/key-principles-of-web-testing/>
6. Обзор решений для тестирования сайтов разные виды тестов для веб-приложений и используемые ПЗ [Электронный ресурс]. – Режим доступа: <http://hostinfo.ru/articles/442>.
7. Пишем максимально эффективный тест-кейс [Электронный ресурс]. – Режим доступа: <http://habrahabr.ru/post/246463/>.
8. Мысли об основах тест дизайна [Электронный ресурс]. – Режим доступа: http://33testers.blogspot.com/2013/07/blog-post_20.html.
9. Тест Дизайн (Test Design) [Электронный ресурс]. – Режим доступа: <http://protesting.ru/testing/testdesign.html>.
10. Как правильно писать предварительные условия в тест-кейсах [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/preconditions-test-cases>.
11. Что такое целостность тест-кейса или Как правильно описать тест-кейс, чтобы проверить функционал [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/test-case-description>.
12. Правила оформления темы тест-кейса [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/test-case-topic>.
13. Парадокс пестицида и поддержка эффективности тест-кейсов [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/pesticide-paradox-support-effectiveness-test-cases>

ЛАБОРАТОРНА РОБОТА №11

ТЕХНІКИ ТЕСТ-ДИЗАЙНУ

Мета лабораторної роботи: ознайомлення з основними техніками тест-дизайну отримання базових теоретичних та практичних навичок в створенні тестів, використовуючи різні техніки тест-дизайну.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Техніки тест-дизайну».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Техніки тест-дизайну».
3. Виконати завдання до лабораторної роботи «Техніки тест-дизайну» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні запитання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Техніки тест-дизайну».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1. При оформленні дисконтної картки, інтернет-магазин надає знижки клієнтам, в залежності від накопиченої суми покупок:

- 1) більше 3000 грн – 3%;
- 2) більше 10000 грн – 5%;
- 3) більше 15000 грн – 10%.

Які класи еквівалентності можна виділити для перевірки правильності нарахування знижки?

Завдання 2. Під час реєстрації на сайті поле «Прізвище» є обов'язковим для заповнення. Дане поле допускається заповнювати лише кирилицею або латиницею, великими та малими буквами, а також допускаються символи «-» (дефіс) та «'» (апостроф).

Опишіть які позитивні та негативні тести краще використовувати для перевірки даного поля для найменшої витрати часу, використовуючи

метод розбиття на класи еквівалентності? А також наведіть приклади тестів, використовуючи дані класи.

Теоретичний матеріал

Техніки тест-дизайну – це методи створення тестів. Техніки містять, як теоретичну частину (деякі рекомендації по складанню тестів), а також практичну. Кожна техніка тест-дизайну повинна надати практичні поради щодо організації правильного процесу тестування. Тому особливо важливо не тільки вивчити техніку, а й спробувати проробити її на практиці.

Техніки можуть містити рекомендації не тільки по тест-дизайну (розробки тестів), але й по виконанню тестів.

Існує незліченна кількість технік тест-дизайну. Кожен тестувальник має право використовувати підходи, які вважає доцільними в тій чи іншій ситуації на свій розсуд.

Найбільш поширені техніки тест-дизайну перераховані нижче:

1. **Еквівалентний поділ** (*Equivalence Partitioning*) – це техніка, яка полягає в розбитті всього набору тестів на класи еквівалентності з подальшим скороченням кількості тестів. **Клас еквівалентності** (*Equivalence class*) – це набір даних, що обробляється однаковим способом мислення й приводить до однакового результату. Наприклад, є діапазон допустимих значень від 1 до 10, необхідно вибрати одне валідне значення всередині інтервалу, скажімо, 5, і одне невалідне значення поза інтервалу - 0.
2. **Аналіз граничних значень** (*Boundary Value Analysis*) – це техніка перевірки поведінки продукту на крайніх (граничних) значеннях вхідних даних. Граничне тестування також може включати тести, що перевіряють поведінку системи на вхідних даних, що виходять за допустимий діапазон значень. При цьому система повинна певним (заздалегідь обумовленим) способом обробляти такі ситуації. Наприклад, за допомогою виняткової ситуації або повідомлення про помилку. Якщо взяти приклад вище, в якості значень для позитивного тестування виберемо мінімальну і максимальну межі (1 і 10), а також в якості значень для негативного тестування – значення більше і менше кордонів (0 і 11). Аналіз граничних значень може бути застосований до полів, записів, файлів, або до будь-якого роду сутностей, які мають обмеження.
3. **Причина/Наслідок** (*Cause/Effect*) – це, як правило, введення комбінацій умов (причин), для отримання відповіді від системи

(наслідків). Наприклад, під час перевірки можливості додавати клієнта, використовуючи певну екранну форму. Для цього необхідно заповнити кілька полів, таких як «Ім'я», «Адреса», «Номер телефону», а потім натиснути кнопку «Додати» – це «Причина». Після натискання кнопки «Додати», система додає клієнта в базу даних і показує його номер на екрані – це «Наслідок».

4. **Передбачення помилки** (*Error Guessing*) – це коли тест-аналітик використовує свої знання системи та здатність до інтерпретації специфікації на предмет того, щоб «передбачити», за яких вхідних умов система може видати помилку. Наприклад, у специфікації зазначено: «Користувач повинен ввести код». Тест-аналітик може задати наступні питання: «А якщо я не введу код і натисну кнопку?», «А якщо я введу неправильний код?» і так далі. Це і є передбачення помилки.
5. **Вичерпне тестування** (*Exhaustive Testing*) – у межах цієї техніки перевіряються всі можливі комбінації вхідних значень, що має допомогти знайти всі проблеми. На практиці застосування цього методу не є можливим через величезну кількість вхідних значень, застосовується лише у крайніх випадках.

Контрольні запитання

1. В яких випадках група тестів являє клас еквівалентності. Навести приклади на практиці.
2. Навести приклади класів еквівалентності при валідному та невалідному введенні даних.
3. Як здійснюється тестування граничних значень? Навести приклад на практиці.
4. Які існують особливості техніки тест дизайну «передбачення помилок»?
5. Для чого використовується та як визначається тестове покриття? Навести приклади на практиці.

Список літератури:

1. Мысли об основах тест дизайна [Электронный ресурс]. – Режим доступа: http://33testers.blogspot.com/2013/07/blog-post_20.html.
2. Техники тест дизайна [Электронный ресурс]. – Режим доступа: http://www.protesting.ru/testing/testdesign_technics.html.
3. Тест-дизайн и ручное тестирование [Электронный ресурс]. – Режим доступа: <http://software-testing.ru/forum/index.php?/forum/111-test-dizajn-i-ruchnoe-testirovanie/>.
4. Эффективное применение комбинаторных техник тест дизайна [Электронный ресурс]. – Режим доступа: <http://w1zle.blogspot.ru/>.
5. Коберн А. Современные методы описания функциональных требований к системам. - М.: Лори, 2002.
6. Классы эквивалентности, граничные значения [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/equivalence-classes-and-boundary-values>.

ЛАБОРАТОРНА РОБОТА №12

ТЕСТ-ПЛАНИ

Мета лабораторної роботи: ознайомлення зі стандартами написання тест-планів, отримання базових теоретичних та практичних навичок в написанні тест-планів.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Тест-плани».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Тест-плани».
3. Виконати завдання до лабораторної роботи «Тест-плани» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Тест-плани».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1. Створити таблицю порівняння шаблонів тест-планів за стандартами IEEE 829 і RUP на основі лекції «Тест-плани». Визначити, який з шаблонів зручніший для використання?

Завдання 2. Створити стислий тест-план для сайту <http://prestashop.qatestlab.com.ua/en/> згідно прикладу.

Порядок виконання завдання:

1. Завантажити приклад тест-плану в Особистому кабінеті у завданні до лабораторної роботи «Тест-плани».
2. Створити стислий тест-план для сайту <http://prestashop.qatestlab.com.ua/> згідно прикладу.
3. Вказати своє прізвище та ім'я у назві документу.
4. Створений тест-план в форматі «.docx», «.doc» або «.pdf» додати до відповіді.

Теоретичний матеріал

Тест-план (Test Plan) – це документ, що описує весь обсяг робіт з тестування, починаючи з опису об'єкта, стратегії, розкладів, критеріїв початку та закінчення тестування, вимог до необхідного в процесі роботи обладнання, спеціальних знань, а також оцінки ризиків з варіантами їх вирішення.

Rational Unified Process (RUP) – адаптивна ітеративна структура для процесу розробки ПЗ, що складається з чотирьох фаз життєвого циклу проекту: початок, проектування, побудова, впровадження.

Хороший тест-план повинен, як мінімум, описувати наступне:

1. **Що потрібно тестувати?** – Опис об'єкта тестування: системи, додатки, обладнання.
2. **Що будете тестувати?** – Список функцій та опис системи, що тестується, та її компонент окремо.
3. **Як будете тестувати?** – Стратегія тестування, а саме: види тестування та їх застосування по відношенню до об'єкта тестування.
4. **Коли будете тестувати?** – Послідовність проведення робіт: підготовка (*Test Preparation*), тестування (*Testing*), аналіз результатів (*Test Result Analysis*) у розрізі запланованих фаз розробки.
5. **Критерії початку тестування:**
 - готовність тестової платформи (*тестового стенду*);
 - завершеність розробки необхідного функціоналу;
 - наявність всієї необхідної документації.
6. **Критерії завершення тестування** – результати тестування задовольняють критеріям якості продукту:
 - вимоги до кількості відкритих дефектів виконані;
 - витримка певного періоду без зміни початкового коду додатка Code Freeze (*CF*);
 - витримка певного періоду без відкриття нових дефектів Zero Bug Bounce (*ZBB*).

Види тест-планів:

1. *Майстер Тест-план (Master Plan or Master Test Plan).*
2. *Тест-план (Test Plan).*
3. *План Приймального Тестування (Product Acceptance Plan)* – документ, що описує набір дій, пов'язаних з приймальним тестуванням (стратегія, дата проведення, відповідальні працівники та ін.).

Явна відмінність Майстер Тест-плану від просто Тест-плану в тому, що Майстер Тест-план є більш статичним в силу того, що містить у собі

високорівневу (High Level) інформацію, яка не схильна до частоті зміни в процесі тестування і перегляду вимог. Сам же детальний тест-план, який містить більш конкретну інформацію по стратегії, видам тестування, розклад виконання робіт, є «живим» документом, який постійно зазнає зміни, що відображають реальний стан справ на проєкті.

Контрольні запитання

1. По яким основним пунктам здійснювалось порівняння шаблонів тест-планів? Обґрунтувати відповідь.
2. Вказати, який з шаблонів тест-плану краще використовувати при тестуванні програмного продукту, відповідь обґрунтувати.
3. З якою метою пишуть тест-плани, чи можна провести тестування програмного продукту без тест-плану? Надати відповідь обґрунтувати та навести приклади на практиці.
4. З якою метою проводиться рецензування та затвердження тест-плану?
5. Хто затверджує та рецензує тест-план?

Список літератури:

1. Тест план (План тестирования) ПО [Електронний ресурс]. – Режим доступу: <http://www.protesting.ru/testing/plan.html>.
2. Тест-план [Електронний ресурс]. – Режим доступу: [http://wiki.software-testing.ru/Тест-план_\(NR\)](http://wiki.software-testing.ru/Тест-план_(NR)).
3. Test Plan Outline (IEEE 829 Format) [Електронний ресурс]. – Режим доступу: <https://jmpovedar.files.wordpress.com/2014/03/ieee-829.pdf>.
4. RUP Test Plan Template [Електронний ресурс]. – Режим доступу: http://www.protesting.ru/documentation/test_plan_template_rup.zip.
5. Test Plan (a Real Sample) [Електронний ресурс]. – Режим доступу: https://www.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Live_Project_Test_Plan_SoftwareTestingHelp.pdf.
6. Клуб успешных менеджеров программистов [Електронний ресурс]. – Режим доступу: <https://ru.bookmate.com/authors/H0Yia9Lh>.
7. Что такое тест-план, для чего он нужен и из чего состоит [Електронний ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/test-plan>

ЛАБОРАТОРНА РОБОТА №13

ЗВІТИ ПРО ТЕСТУВАННЯ

Мета лабораторної роботи: ознайомлення зі структурою звіту про тестування, отримання базових теоретичних та практичних навичок в складанні звіту про тестування.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Звіти про тестування».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Звіти про тестування».
3. Виконати завдання до лабораторної роботи «Звіти про тестування» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Звіти про тестування».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Скласти звіт про тестування сайту <http://prestashop.qatestlab.com.ua/>, виходячи з результатів виконання лабораторних робіт «Кросбраузерне тестування», «Тестування веб-проектів», «Функціональне тестування».

Теоретичний матеріал

Мета звіту – дати розробнику інформацію про поточний стан програмного продукту. Повна інформація про стан продукту є тільки у команди, що приймала участь в процесі тестування, яка перевірила всі функції додатку, володіє інформацією про кількість дефектів та динаміку зміни кількості, наприклад, порівняно з попереднім білдом.

Звіт про результати тестування в основному потрібен:

- менеджеру проекту;

- лідеру команди розробників;
- лідеру команди тестувальників;
- замовнику.

Якщо звіт пишеться для розробника або менеджера, то ніяких запитань щодо надання звіту не виникає – необхідна термінологія та опис проблем буде легко сприйматися людиною, яка буде читати представлений звіт. Але якщо поставлена мета написати звіт для людей, які не знайомі зі специфікою та технічною стороною програмного забезпечення, то з'являється задача, для вирішення якої краще використовувати графічний вид надання інформації: графіки та діаграми.

Це може бути, наприклад, графік по пройдених тест-кейсах з кількістю дефектів (рис. 13.1).



Рис. 13.1. Графік по пройдених тест-кейсах і кількості дефектів

Також це може бути, наприклад, графік пройдених тестових випадків (*test case*) по модулям, який наочно покаже, який обсяг роботи в кожному з модулів вже пророблений та допоможе виявити проблеми (рис. 13.2).

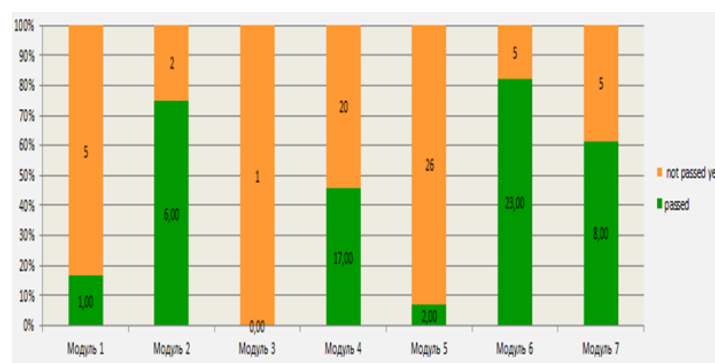


Рис. 13.2. Графік пройдених ТК по модулям

На додаток до діаграми або графіка необхідно надати зведену таблицю з усіма даними (табл.13.1).

Таблиця 13.1. Зведена таблиця

Модуль	121 Загальна кількість ТК по модулю	Проходження	
		passed	not passed yet
Модуль 1	6	1	5
Модуль 2	8	6	2
Модуль 3	1	0	1
Модуль 4	37	17	20
Модуль 5	28	2	26
Модуль 6	28	23	5
Модуль 7	13	8	5

Також, наприклад, наводиться зведена таблиця, з інформацією про помилки, з якими команди тестувальників доводилося мати справу за весь час роботи з проектом, яку можна зобразити у вигляді графіка (рис. 13.3).

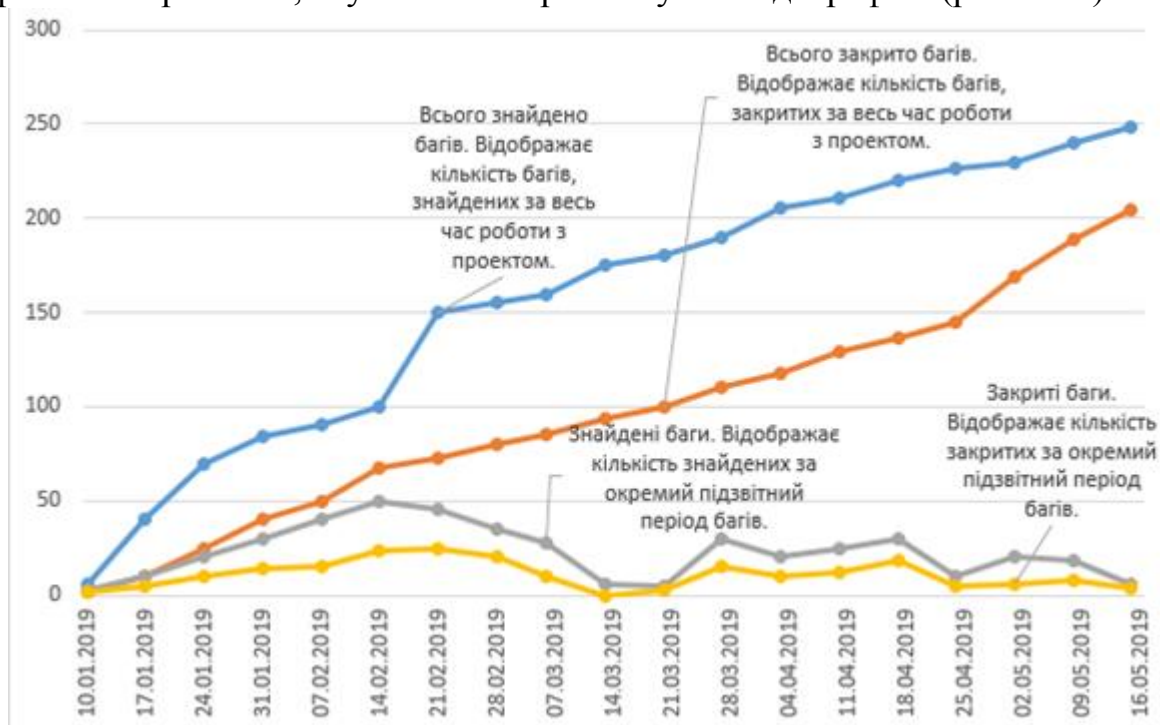


Рис.13.3. Графік з інформацією про помилки

В звіт можна додати графік, котрий показує, яка кількість тестових випадків перейшла в стан автоматизованих за певний проміжок часу (рис. 13.4).

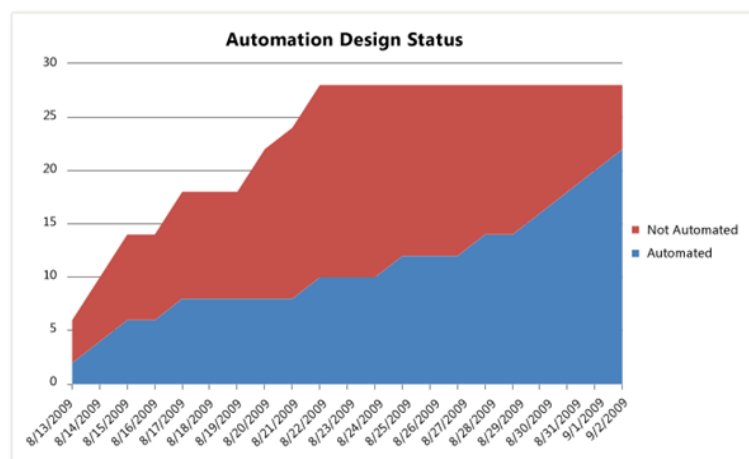


Рис.13.4. Кількість тестових випадків, яка перейшла в стан автоматизованих за певний проміжок часу.

При створенні звіту обов'язково необхідно вказувати:

1. Склад команди;
2. Терміни виконання, за які складається звіт;
3. Опис процесів тестування;
4. Зміни тестової моделі, доповнення ТК;
5. Відсоток пройдених ТК;
6. Критичні та блокуючі проблеми та вжиті заходи по їх усуненню;
7. Результати регресу (акцент на невирішених проблемах);
8. План на наступну ітерацію\тиждень\місяць.

Пункти 3, 4, 6 та 8 варто писати з урахуванням цільової аудиторії звіту, 7 пункт варто вказувати тоді, коли проводилося «регрес-тестування», зазвичай цей пункт фігурує в «версійних» звітах. Пункт 8 з підсумкового звіту виключається.

Контрольні запитання

1. Які основні пункти входять до звіту тестування? Обґрунтувати їх необхідність.
2. Вказати, які існують особливості при кросбраузерному тестуванні.
3. На яку аудиторію орієнтувалися при написанні звіту про тестування? Надану відповідь обґрунтувати.

4. Яку інформацію обов'язково потрібно вказувати при написанні будь-якого звіту про тестування?
5. Які додаткові документи можуть бути додані до звіту про тестування?

Список літератури:

1. Создание понятных отчетов о тестировании [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/company/performance_lab/blog/207512/.
2. Как написать отчёт о тестировании [Электронный ресурс]. – Режим доступа: <https://geteasyqa.com/ru/qa/write-test-report/>
3. Test Team Progress Excel Report [Электронный ресурс]. – Режим доступа: <https://msdn.microsoft.com/ru-ru/library/ee730420.aspx>.
4. Всё, что вам нужно знать о форматах отчётов в тестировании ПО [Электронный ресурс]. – Режим доступа: <https://dou.ua/lenta/articles/software-test-formats/>
5. Как написать отчёт о тестировании [Электронный ресурс]. – Режим доступа: <https://geteasyqa.com/ru/qa/write-test-report/>
6. Как улучшить структуру Вашего отчета о тестировании? [Электронный ресурс]. – Режим доступа: <http://blog.i.ua/user/6074047/1297477/>.

ЛАБОРАТОРНА РОБОТА №14

МОБІЛЬНЕ ТЕСТУВАННЯ

Мета лабораторної роботи: ознайомлення з основними відмінностями мобільних веб-сайтів та додатків від їх аналогів на ПК, отримання базових теоретичних та практичних навичок в вимірюванні швидкості роботи мобільних додатків.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Мобільне тестування».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Мобільне тестування»..
3. Виконати завдання до лабораторної роботи «Мобільне тестування». в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Мобільне тестування»..
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Встановити будь-який безкоштовний додаток з магазину на мобільний пристрій та провести тестування мобільного додатку, використовуючи контрольний список.

Порядок виконання завдання:

1. Завантажити приклад чекліста в Особистому кабінеті у завданні до лабораторної роботи «Мобільне тестування».
2. Вказати прізвище та ім'я власника у назві контрольного списку.
3. Додати мінімум 5-7 пунктів, які відповідають тестуванню на мобільному пристрої, до існуючих пунктів контрольного списку на вкладці «Моб.пристрій (додаток)», не повторюючись з вже доданими, та відмітити їх будь-яким кольором.

4. Провести тестування додатку, перевіривши всі пункти контрольного списку на вкладці «Моб.пристрій (додаток)».
5. Вказати назву/модель мобільного пристрою та версію ОС. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на звіти у системі відслідковування помилок Mantis Bug Tracker <http://mantis.gatestlab.net> або додати тему багу за принципом «Що? Де? Коли?».
6. Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» додати до відповіді.

Теоретичний матеріал

Тестування мобільних додатків – процес, під час якого програмна частина додатку для мобільних пристроїв тестується на функціональність, зручність та змістовність. Таке тестування може бути ручним або автоматизованим.

Основні відмінності мобільних та десктопних додатків:

- екран;
- датчики та пристрої введення;
- телефонні функції;
- енергоспоживання;
- мережа;
- особливості платформи;
- вузька спеціалізація;
- норми та рекомендації.

Мобільний web-сайт – спеціалізований сайт, адаптований для перегляду та функціонування на мобільному пристрої.

Тестування веб-додатків має дати оцінку якості мобільних веб-додатків при використанні різних браузерів на різних мобільних пристроях. На відміну від мобільних додатків, веб-додатки зазвичай дозволяють звертатися до функцій базового серверу через тонкий мобільний клієнт. Таким чином, крім аналізу функціональності та поведінки, виконання вимог до QoS, зручності використання, безпеки та конфіденційності, тестування мобільних веб-додатків повинно враховувати ще й зв'язність.

Мобільний додаток – це спеціально розроблений додаток під конкретну мобільну платформу або декілька платформ.

Спеціально призначені для тестування програми встановлюються та запускаються на мобільних пристроях і, як правило, звертаються до спеціальних інтерфейсів API (наприклад, до API GPS) та апаратних компонентів (наприклад, до камери). Веб-додатки складаються з сервера

додатків та клієнтського ПЗ, виконуваного в середовищі браузерів, через які користувачі отримують доступ до сервісів.

Тестування додатків, спеціально призначених для мобільних пристроїв, дає змогу оцінити якість мобільних програм, які завантажуються та виконуються на різних мобільних пристроях обраної мобільної платформи. Тестування спрямоване на аналіз функціональності та поведінки (у тому числі підтримки специфічних для пристрою функцій – наприклад, управління за допомогою жестів), виконання вимог до QoS, зручності використання, безпеки та конфіденційності.

Контрольні запитання

1. Дайте визначення поняттю «тестування мобільних додатків».
2. Вказати особливості при формуванні контрольного списку для тестування мобільного додатку.
3. Перелічіть основні програми для налагодження додатків (Android/iOS).
4. Обґрунтувати або спростувати доцільність використання емуляторів при тестуванні мобільних додатків.

Список літератури:

1. Особенности тестирования приложений на мобильных устройствах [Электронный ресурс]. – Режим доступа: http://www.enterra.ru/blog/mobile_qa/.
2. Тестирование мобильных приложений [Электронный ресурс]. – Режим доступа: <http://www.osp.ru/os/2014/03/13040836/>.
3. Особенности тестирования мобильных приложений (iOS, Android) [Электронный ресурс]. – Режим доступа: <http://mabee.ru/services/razrabotka-mobilnykh-prilozheniy/testirovanie-mobilnykh-prilozheniy/>
4. Особенности тестирования приложений на мобильных устройствах [Электронный ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/testing-mobile-devices/>
5. Тестирование мобильных приложений [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/hub/mobile_testing/.
6. Software Quality Characteristics [Электронный ресурс]. – Режим доступа: http://thetesteye.com/posters/TheTestEye_SoftwareQualityCharacteristics.pdf.
7. Что нужно знать дизайнеру о мобильных устройствах [Электронный ресурс]. – Режим доступа: <http://proweb63.ru/help/design/design-knows>

ЛАБОРАТОРНА РОБОТА №15

МОБІЛЬНЕ ТЕСТУВАННЯ ВЕБ-ПРОЕКТІВ

Мета лабораторної роботи: отримання базових теоретичних та практичних навичок в написанні чеклістів для веб-проектів, проведення тестування сайту на основі створеного чекліста.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Мобільне тестування веб-проектів».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Мобільне тестування веб-проектів».
3. Виконати завдання до лабораторної роботи «Мобільне тестування веб-проектів» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Мобільне тестування веб-проектів».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1

Провести тестування веб-сайту <http://prestashop.qatestlab.com.ua/en/> в різних мобільних браузерах на мобільному пристрої, використовуючи контрольний список.

Порядок виконання завдання:

1. Завантажити приклад чекліста в Особистому кабінеті у завданні до лабораторної роботи «Мобільне тестування веб-проектів».
2. Вказати прізвище та ім'я власника у назві контрольного списку.
3. Додати мінімум 5-7 пунктів, які відповідають тестуванню на мобільному пристрої, до існуючих пунктів контрольного списку на вкладці «Моб.пристрій (сайт)», не повторюючись з вже доданими, та відмітити їх будь-яким кольором.

4. Провести тестування сайту <http://prestashop.qatestlab.com.ua/en/>, перевіривши всі пункти контрольного списку на вкладці «Моб.пристрій (сайт)» мінімум в 2х браузерах в останніх або передостанніх версіях.
5. Вказати назву та версію браузера. Вказати назву/модель мобільного пристрою та версію ОС. Результат перевірки відзначити «Passed/Failed». Для «Failed» вказати в примітці або нотатках посилання на звіти у системі відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> або додати тему багу за принципом «Що? Де? Коли?».
6. Доповнений та пройдений чекліст в форматі «.xlsx» або «.xls» додати до відповіді.

Завдання 2

1. Внести 5-7 знайдених багів у систему відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net>.
2. У відповіді до завдання вказати посилання на звіти про дефекти.
3. Порівняти мобільну та десктопну версію сайту, проаналізувати знайдені дефекти, які відтворюються виключно на мобільній версії.

Теоретичний матеріал

Існує поняття **Responsive web-design** («адаптивний дизайн»). Даний термін українською звучить як «адаптивний веб-дизайн», звідси походить назва виду верстки – адаптивна верстка, яка забезпечується за допомогою використання css media-запитів.

Адаптивна верстка – це основне, що перевіряється при мобільному тестуванні веб проектів. Для успішної верстки необхідно перевірити виконану роботу адаптації. Далі розглянемо доступні на сьогодні способи такої перевірки.

Тестування з використанням мобільних пристроїв – це найбільш якісний і, в той же час, найбільш дорогий підхід до тестування, оскільки необхідна велика кількість пристроїв.

Багато в чому мобільне тестування веб-проектів нічим не відрізняється від десктопного тестування. Однак є кілька ключових відмінностей, пов'язаних з конструктивними особливостями мобільних пристроїв.

Перше – це розміри екрана. Мобільні пристрої мають значно менші розміри екрану, ніж монітор комп'ютера. Сторінка, яку неможливо масштабувати, як правило повинна міститися по ширині екрану пристрою,

горизонтальної прокрутки не повинно бути. Весь контент повинен добре читатися та бути доступним для перегляду.

Друге – це навігація по сторінках сайту і взаємодія з активними елементами на сайті за допомогою сенсорного екрану, а не мишки і клавіатури.

Область тапа на мобільному пристрої повинна бути такою, щоб користувач мав можливість управляти всіма активними елементами на сайті. При подвійному тапі на активні елементи вони повинні автоматично збільшуватися. При тапі на текстові поля вони також автоматично збільшуються і з'являється клавіатура. При подвійному тапі на текстовий блок текст повинен вирівнюватися щодо меж екрану і масштабуватися для комфортного читання.

Третє – можливість повороту на мобільному пристрої з портретного режиму в альбомний і навпаки. **Основні проблеми при повороті пристрою:**

- 1) можуть зникати відкриті вікна, спливаючі підказки і деякі інші активні елементи;
- 2) якщо на момент повороту курсор знаходиться в текстовому полі, іноді зникає клавіатура або фокус переноситься на інший елемент;
- 3) при повороті пристрою може не змінитися ширина клавіатури;
- 4) при одночасному введенні тексту і повороті пристрою введений текст може зникати.

Четверте – особливості роботи з мультимедіа. Зокрема, відтворення відео через Adobe flash player. Тут проблеми можуть виникнути у зв'язку з тим, що корпорація Adobe припинила підтримку флеш на Android, тому не у всіх сучасних пристроях на операційній системі Android встановлений Adobe Flash Player. Розробники замінюють його іншими технологіями відтворення відео – наприклад HTML5.

Особливу увагу слід звернути на розмітку сторінки. У різних режимах перегляду розмітка сторінки і взаємне розташування різних блоків може відрізнятись.

Використання реальних пристроїв для тестування особливо актуально при тестуванні функціоналу сайту, хоча, під час попереднього тестування, для цих цілей можна використовувати спеціально-розроблені програми-симулятори.

У найбільш популярних веб-браузерах існують вбудовані засоби для мобільного тестування. До таких браузерів відносяться: Google Chrome, Mozilla Firefox, Opera. Найпростіший спосіб – зміна ширини вікна браузера. Це можна робити як вручну, так й за допомогою спеціальних плагінів. Крім того, існує безліч плагінів для браузерів для мобільного тестування:

- Google Chrome – плагіни Responsive Viewer, Resolution Test, Window Resizer, Screen Resolution Tester, Viewport Resizer, Mobile View, Browser Resize;
 - Opera – плагін Resize Me, Responsive Web Design Tester.
- Однак можна знайти інші плагіни на будь-який смак.

Контрольні запитання

1. Обґрунтувати, що навігація по сторінці сайту та взаємодія з активними елементами на мобільному пристрої відрізняється від десктопного.
2. Вказати особливості роботи з мультимедіа на мобільних пристроях. Навести приклади.
3. Обґрунтувати твердження, що використання реальних мобільних пристроїв актуально при тестуванні функціоналу сайту.
4. Пояснити та вказати причини знайдених відмінностей при тестуванні сайту на мобільному та десктопному пристроях?
5. Яку роль відіграє адаптивний дизайн в якості розробленого сайту?

Список літератури:

1. Особенности тестирования приложений на мобильных устройствах [Електронний ресурс]. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/testing-mobile-devices/>
2. Тестирование для мобильных устройств: эмуляторы, симуляторы и удаленная отладка [Електронний ресурс]. – Режим доступу: <http://habrahabr.ru/post/237499/>.
3. Как протестировать мобильный сайт: проверка верстки и отображения дизайна в мобильных браузерах [Електронний ресурс]. – Режим доступу: <http://great-world.ru/kak-protestirovat-mobilnyj-sajt/>.
4. Тестирование мобильных приложений [Електронний ресурс]. – Режим доступу: <http://www.osp.ru/os/2014/03/13040836/>.
5. Особенности тестирования мобильных приложений [Електронний ресурс]. – Режим доступу: <https://qaevolution.ru/osobennosti-testirovaniya-mobilnyx-prilozhenij/>

ЛАБОРАТОРНА РОБОТА №16

ІНСТРУМЕНТИ ТЕСТУВАННЯ МОБІЛЬНИХ ДОДАТКІВ

Мета лабораторної роботи: отримання базових теоретичних та практичних навичок у тестуванні мобільних додатків з використанням інструментів тестування, написанні тестових випадків для тестування мобільних додатків та проведенні тестування сайту на основі створених тестових випадків.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Інструменти тестування мобільних додатків».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Інструменти тестування мобільних додатків».
3. Виконати завдання до лабораторної роботи «Інструменти тестування мобільних додатків» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Інструменти тестування мобільних додатків».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1.

Встановити будь-який безкоштовний додаток з магазину на мобільний пристрій, створити тест-кейси для тестування мобільного додатку у системі TestLink <http://tl-univer.qa-testlab.net>.
У відповіді вказати ID створених тест-кейсів.

Завдання 2.

Занести у систему відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net> 5-7 багів, знайдених під час проведення тестування мобільного додатку. У відповіді додати посилання на баг-репорти.

Завдання 3.

Користуючись матеріалами лекції, встановити один з наступних інструментів: Xcode, ADB, Windows Phone SDK 8. У відповіді додати скріншоти встановленої програми.

Теоретичний матеріал

Xcode – інтегроване середовище розробки програмного забезпечення під mac OS та iOS, розроблена корпорацією Apple.

Має широкий функціонал:

- встановлення програм;
- видалення програм;
- зняття креш-логів;
- замір витрати пам'яті;
- завантаження скріншотів / фотографій;
- перегляд логів в real-time з пристрою.

Копіювання crash логів з iOS на Windows

1. Підключити пристрій до ПК;
2. Автоматично відкриється iTunes;
3. У iTunes зайти в потрібний девайс;
4. Синхронізувати пристрій з ПК, для цього потрібно натиснути на кнопку «Синхронізувати».
5. Почекати закінчення синхронізації;
6. Тепер весь список з пристроїв записаний в папку виду:

Windows XP:

C:\Documents and Settings\<USERNAME>\Application
Data\Apple
Computer\Logs\CrashReporter\MobileDevice\<DEVICE_NAME>

Windows Vista or 7-10:

C:\Users\<USERNAME>\AppData\Roaming\Apple
Computer\Logs\CrashReporter\MobileDevice\<DEVICE_NAME>
Де <USERNAME> - ім'я поточного користувача системи.

7. Далі вибрати папку з назвою потрібного пристрою та скопіювати потрібний лог. У імені кожного з логів відображається час створення.

Safari Web Inspector

Почавши працювати з Inspector можете:

- бачити та вносити зміни у HTML та CSS;
- отримувати доступ до сховищ та переглядати куки, локальну, а також сесійну та SQL бази даних;
- здійснювати профільну перевірку продуктивності веб-додатку, включаючи тестування мережеских запитів, а також перевірку шарів, візуалізації та JavaScript подій. Це безумовно великий крок у розвитку інструментів для оптимізації продуктивності;
- проводити пошук по DOM;
- одночасно бачити всі попередження та повідомлення про помилки;
- управляти Web Workers (потокими);
- управляти зупинками виконання JavaScript та Uncaught, а також задавати винятки;
- отримувати доступ до консолі та виконувати JavaScript;
- займатися налагодженням JavaScript коду;
- перевіряти на дотик: за допомогою маленької іконки у вигляді долоні можете перевіряти механіка сенсорного введення на пристрої та налагоджувати DOM-елементи.

Налаштування Web Inspector:

1. Підключити iOS пристрій до Mac.
2. Запустити Safari на iPad/iPhone.
3. Перейти в налаштування Settings→ Safari→ Advanced.
4. Включити Web Inspector
5. Запустити Safari на Mac
6. Відкрити настройки браузера→ Preferences→ Advanced
7. Увімкнути настройку Show Develop menu in menu bar.
8. Вибрати Develop menu.
9. Знайти в списку підключений iOS пристрій, потім вибрати тестовий веб-додаток.

ADB

Абревіатура **ADB** розшифровується як **Android Debug Bridge** (відладочний міст Андроїд). ADB є складовою частиною **Android SDK**.

Так як операційна система Android є різновидом Linux, для її налаштування часто виникає необхідність роботи через командний рядок. Звичайно, існують програми-емулятори терміналу, які дозволяють виконувати команди прямо на пристрої але, по-перше, на маленькому екрані телефону робити це незручно, а по-друге, іноді потрібно доступ до пристрою через комп'ютер, і в цих та багатьох інших випадках програма ADB просто

незамінна. Програма ADB встановлює зв'язок між пристроєм та комп'ютером й дозволяє прямо на комп'ютері виконувати різні маніпуляції з системою Android.

Зняття логів з Android пристроїв за допомогою LogCat (Android Device Monitor):

1. Встановити ADB.
2. Зайти в налаштування на телефоні або планшеті.
3. В кінці списку вибрати «About device».
4. Натискати на номер білду до тих пір, поки не ввімкнеться режим розробника.
5. Перейти в розділ «Developer options» який з'явився.
6. Відзначити чекбокс «USB debugging».
7. Підключити телефон або планшет до ПК.
8. Підтвердити у діалоговому вікні на пристрої, що ми довіряємо ПК.
9. Запустити файл «monitor.exe», який знаходиться в папці з інструментами. У нашому випадку адреса буде: папка Android → sdk → tools → lib → monitor-x86_64. У разі, якщо операційна система хоче «захистити» Вас від запуску даного файлу, потрібно вибрати «More info → Run anyway».
10. Мобільне тестування показує, що перший запуск може зайняти до 15-20 секунд.
11. У вікні необхідно вибрати пристрій, з якого буде проводитися логування.
12. Після виконання дій, які повинні бути залоговані, необхідно вибрати потрібну ділянку, використовуючи мишку і клавішу SHIFT, або виділити все поєднанням CTRL + A.
13. Зберегти лог у файл.

.XAP – формат файлу, який використовується в Windows Phone для установки програм. Являє собою формат файлів для архівів із стисненням. Всередині знаходиться додаток, супутні папки і кілька xml-файлів, що відповідають за безпеку і порядок доступу до бібліотек програми.

Покрокова інструкція по установці .XAP файлів:

1. На ПК має бути встановлений Windows Phone SDK.
2. Підключити Windows Phone смартфон до комп'ютера за допомогою USB кабелю.
3. Запустити Application Deployment, який встановлюється разом з Windows Phone SDK.

4. У програмі Application Deployment вибрати мету «Device» (це смартфон, який підключений по USB).
5. У рядку «ХАР-файл» натискаємо кнопку «Огляд» і вибираємо завантажений вже раніше файл додатку (додатки під операційну систему Windows Phone мають розширення * .xap).
6. Натиснути на кнопку «Розгорнути».
7. Почекаати завершення установки.

Windows Phone Power Tools - спеціальний інструмент для розробників додатків, але корисний він не тільки для них, а й для звичайних користувачів. Це програмне забезпечення має функціонал, який не доступний на Windows Phone SDK, що робить його кращим.

Windows Phone Power Tools має такі можливості як:

- установка, оновлення, видалення ХАР додатків розробника - дуже корисна функція, яку оцінить кожен з користувачів;
- завантаження файлів, а також візуальний браузер з Isolated Storage;
- можливість отримання детальної інформації про ваш пристрій.

Контрольні запитання

1. Які додаткові інструменти були використані при тестуванні додатку встановленого на мобільний пристрій?
2. Обґрунтувати необхідність проведення замірів пам'яті.
3. Обґрунтувати необхідність креш-логів та яким чином можна отримати інформацію про аварійне завершення програми?
4. З якою метою при тестуванні мобільних додатків отримують root права?
5. Описати процес отримання root прав.

Список літератури:

1. Android Debug Bridge (adb) [Електронний ресурс]. – Режим доступу: <http://developer.android.com/tools/help/adb.html>.
2. LogCat [Електронний ресурс]. – Режим доступу: <http://developer.alexanderklimov.ru/android/debug/logcat.php>
3. Windows Phone SDK [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Windows_Phone_SDK.
4. Как настроить и пользоваться программой iTools: инструкция по применению [Електронний ресурс]. – Режим доступу: <http://bit.ly/2GIyKl2>
5. Как пользоваться утилитой Instruments в Xcode: [Електронний ресурс]. – Режим доступу: <http://habrahabr.ru/post/168491/>.
6. Ищем утечки памяти в iPhone приложениях [Електронний ресурс]. – Режим доступу: <http://heximal.ru/blog/coding/ishhem-utechki-pamyati-v-iphone-prilozheniyah/>.
7. Сбор логов и снятие скриншотов с iPhone и iPad [Електронний ресурс]. – Режим доступу: http://mobile-testing.ru/how_to_test_mobileapp/logs_and_screens_apple/
8. Просмотр системных логов [Електронний ресурс]. – Режим доступу: <http://bulkin.me/notes/2884>.
9. Устанавливаем тестируемое приложение на Ваше устройство [Електронний ресурс]. – Режим доступу: <http://www.akhozya.com/2012/05/blog-post.html>.

ЛАБОРАТОРНА РОБОТА №17

ТЕСТУВАННЯ ІГОР

Мета лабораторної роботи: ознайомлення з видами тестування ігор, отримання базових теоретичних та практичних навичок в розпізнаванні дефектів в іграх за видами.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Тестування ігор».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Тестування ігор».
3. Виконати завдання до лабораторної роботи «Тестування ігор» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Тестування ігор».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Виконати ad-hoc тестування обраної гри та внести в баг-трекер Mantis <http://mantis.qatestlab.net> функціональні та локалізаційні баги.

Порядок виконання завдання:

1. Встановити сервіс цифрового розповсюдження ігор Steam <https://store.steampowered.com/about> (створити акаунт у встановленому клієнті Steam та встановити безкоштовну гру у ранньому доступі) або завантажити та встановити будь-яку безкоштовну гру з інтернету.
2. Знайти та внести 3 функціональних бага та 3 бага локалізації у систему відслідковування помилок Mantis Bug Tracker <http://mantis.qatestlab.net>.
3. У відповіді до завдання вказати посилання на звіти про дефекти.

Теоретичний матеріал

Види тестування ігор:

1. ***Функціональне тестування.*** Мета – виявити відхилення від функціональних вимог. Зводиться до багаторазового проходження гри, виявленню неполадок й умов, в яких їх можна виправити.

2. ***Навантажувальне тестування.*** При тестуванні ігор доцільно створити ситуації, які вимагають великого обчислювального навантаження. Таким чином, тестувальник може перевірити продуктивність системи в стресовій ситуації. Так легше помітити й вчасно виправити потенційно ненадійні ділянки коду.

3. ***Тестування локалізації.*** Ігри часто перекладають на мови тих країн, в яких їх планують випускати на ринок. У деяких випадках, перекладачі не можуть надати абсолютно точний переклад, який би повністю відповідав подіям гри. Навіть правильно перекладена фраза може не відповідати ситуації та не сприйматися носієм мови. Тому, після локалізації корисно провести тестування гри мешканцями тих країн, де передбачається реалізація кінцевого продукту.

4. ***Тестування на сумісність.*** Найчастіше програмування ігор проводять на персональних комп'ютерах або ноутбуках. Однак багато ігор можуть бути призначені для інших пристроїв: ігрових приставок, мобільних телефонів, комунікаторів та ін. Розробка здійснюється на симуляторах даних пристроїв, однак вони можуть значно відрізнятися від оригіналу. Тому, надалі можуть виникнути труднощі при запуску гри на оригінальному пристрої. Крім того, слід звернути особливу увагу на ліцензування програмного забезпечення.

При будь-яких відхиленнях гру можуть повернути на доопрацювання, що займає додатковий час та втрату грошових коштів.

Отже, дуже важливо перевірити гру на відповідність заявленим параметрам апаратного забезпечення.

Ігрові механіки (game mechanics)

Ігрові механіки – внутрішньо-ігрові механізми, які допомагають гравцям досягти мети гри, іншими словами це визначений набір правил та петель зворотного зв'язку, які призначені для створення сценаріїв комп'ютерної гри (***gameplay***), які приносять задоволення від ігрового процесу та є будівельними блоками, які можуть застосовуватися та комбінуватися з ігровим й не ігровим контекстом.

Механіка «point-and-tap»

Предмет має два стани:

- активний;
- неактивний;
- при натисканні на предмет, він переходить в активний стан;
- поки предмет активний, натискання гравця по ігровому полю сприймається, як спроба застосувати предмет до точки натискання;
- повторне натискання на предмет переводить його в неактивний стан;
- у разі якщо предмет застосовується в неправильному місці, то він так само переходить в неактивний стан.

Механіка «drag-and-drop»

Використовуючи механіку «drag-and-drop» предмет має також два стани:

- активний;
- неактивний;
- здійснювати «drag&drop» з неактивним предметом неможливо;
- щоб активізувати предмет гравцеві необхідно натиснути на нього;
- захопив активізований предмет, його можна переміщати по ігровому полю за допомогою механіки «drag&drop»;
- наблизивши предмет до краю екрану, екран повинен почати прокручуватись (*scroll*);
- швидкість прокрутки (*scroll*) повинна збільшуватися залежно від того, наскільки близько гравець підвів предмет до краю екрана;
- предмет застосовується по контуру об'єкту (*sprite*), а не по точці під пальцем;
- якщо гравець відпустить предмет, відбувається подія завершення (*release*), над зоною скомпонованого об'єкту, частина якого є активується предметом, таким чином предмет вважається зібраним;
- якщо гравець відпустить предмет над зоною скомпонованого об'єкта, частиною якого не активується предметом, то предмет залишається на тому ж місці, куди його перемістив гравець.

Контрольні запитання

1. Вказати та охарактеризувати існуючі платформи для тестування ігор.
2. Що входить до фокус-тестування ігор? Навести приклади знайдених дефектів та обґрунтувати відповідь.
3. Вказати основну мету проведення тестування локалізації. Навести приклади знайдених дефектів та обґрунтувати відповідь.

4. Які дефекти в грі вважаються критичними? Навести приклади знайдених дефектів та обґрунтувати відповідь.
5. Вказати особливості тестування казуальних ігор. Навести приклади та обґрунтувати відповідь.

Список літератури:

1. Кожаев В. Тестирование интереса к игре [Електронний ресурс]. – Режим доступу: <http://dSPACE.nbuv.gov.ua/bitstream/handle/123456789/14703/3%D0%A06%20%D1%81%20452-456.pdf?sequence=1>.
2. Олейник О. Презентация с SQA Days #11. Тестирование игр на мобильных устройствах и 3D телевизорах [Електронний ресурс]. – Режим доступу: <http://bit.ly/2Zvqk8o>
3. Нечаева Ю. О тестировании одной игры с картинками [Електронний ресурс]. – Режим доступу: <http://habrahabr.ru/post/98203/>.
4. Черный Л. Организация процесса тестирования компьютерной игры [Електронний ресурс]. – Режим доступу: <http://www.dailytelefrag.com/articles/print.php?id=1269>
5. Анкудинов Д. CodeFest 2012. О специфике мультиплатформенного тестирования игр (+ Видео) [Електронний ресурс]. – Режим доступу: <http://2012.codefest.ru/lecture/405>.
6. Тестирование интереса к игре [Електронний ресурс]. – Режим доступу: <http://dSPACE.nbuv.gov.ua/bitstream/handle/123456789/14703/3%D0%A06%20%D1%81%20452-456.pdf?sequence=1>
7. Game Testing All in One by Charles P. Schultz, Robert Bryant and Tim Langdell [Електронний ресурс]. – Режим доступу: <http://computernote.files.wordpress.com/2011/04/course-technology-game-testing-all-in-one-feb20051.pdf>
8. Claudio Redavid, Adil Farid. An Overview of Game Testing Techniques [Електронний ресурс]. – Режим доступу: <https://pdfs.semanticscholar.org/4361/a1882ca8ea296ff6411dbbaa90ca5fbc3ed4.pdf>
9. Зальцман М. Компьютерные игры: как это делается (Глава 15 – Тестирование) [Електронний ресурс]. – Режим доступу: https://gcup.ru/load/books/mark_zalcman_kompjuternye_igry_kak_ehto_del_aetsja/7-1-0-1973

ЛАБОРАТОРНА РОБОТА №18

РОЛІ В ПРОЦЕСІ ТЕСТУВАННЯ ПЗ. КОМУНІКАЦІЇ У СФЕРІ ТЕСТУВАННЯ

Мета лабораторної роботи: ознайомлення з основними ролями у процесі розробки програмного забезпечення та комунікації у сфері тестування.

Практична частина

Завдання до лабораторної роботи:

1. Ознайомитися з матеріалом лекції «Ролі в процесі розробки ПЗ. Комунікації у сфері тестування».
2. Ознайомитися з теоретичним матеріалом до лабораторної роботи «Ролі в процесі розробки ПЗ. Комунікації у сфері тестування».
3. Виконати завдання до лабораторної роботи «Ролі в процесі розробки ПЗ. Комунікації у сфері тестування» в Особистому кабінеті.
4. Зробити висновки до лабораторної роботи та надати відповіді на контрольні питання.

Порядок виконання лабораторної роботи:

1. Відкрити сайт <http://clients.qatestlab.com> та авторизуватися в системі.
2. Відкрити розділ «Тести».
3. Знайти тест Лабораторна робота «Ролі в процесі розробки ПЗ. Комунікації у сфері тестування».
4. Натиснути кнопку «Почати проходження».

Зміст тесту:

Завдання 1.

Обговорити ситуацію:

Замовник повідомив, що реліз не відбудеться в потрібну дату, оскільки продукт виглядає жахливо. Відомо, що розробник А вніс в останній день перед релізом нову особливість (*feature*), а тестувальник В, тестуючи її, не перевіряв працездатність продукту в цілому. Хто несе відповідальність?

Завдання 2.

Робота в групі:

Потрібно розділитися на групи, щоб змодельовати процес розробки ПЗ. Кожен учасник виконує одну роль в команді:

- замовник;
- менеджер проекту;
- QA менеджер;
- розробник;
- тестувальник.

Група визначає вид проекту, його назву, етап розробки (стартап, реліз). Після обговорення менеджер проекту описує основні проблеми в комунікаціях та як проблеми були вирішені.

По завершенню оформити звіт по результатам та прикріпити до відповіді.

Теоретичний матеріал

Процес розробки програмного забезпечення (англ. *Software development process, software process*) – структура, згідно якої побудована розробка програмного забезпечення (ПЗ).

Існує кілька моделей такого процесу, кожна з яких описує свій підхід, у вигляді завдань і/або діяльності, які мають місце в ході процесу.

Для уникнення можливих конфліктів у команді потрібно чітко розмежувати ділянку робіт кожного учасника, в тому числі замовника. Для цього у кожного учасника проекту є відповідна роль.

Для ефективної роботи команди потрібне дотримання наступних умов:

- ясне і чітке розуміння кожним членом команди своєї ролі, що дозволяє кожному виконувати свої завдання;
- специфікація проекту і графік робіт узгоджені з усіма членами команди;
- члени команди добре взаємодіють один з одним і відчують взаємну повагу до професійних якостей один одного;
- всі члени команди мають чітке уявлення про модель процесу, яка використовуватиметься в ході виконання проекту;
- кожен член команди повинен ґрунтовно знати всі аспекти плану проекту.

Ролі в команді розробників ПЗ:

Менеджер Проекту (PM) відповідальний за:

- організацію процесу розробки;

- координацію і контроль всіх видів діяльності в проекті;
- розробку плану проекту (Project Plan);
- проведення регулярних статус мітингів у проектній групі;
- контроль готовності нового білду для QA;
- надання замовнику документації та проміжних версій для перегляду та затвердження або коментування;
- регулярне спілкування із замовником, з'ясування вимог;
- надання звітів про статус проекту.

Бізнес аналітик (Business Analyst) відповідальний за:

- з'ясування і аналіз всіх вимог замовника;
- фіксування всіх вимог замовника (у баг-трекінговій системі та в функціональних специфікаціях), простежування всіх змін у вимогах;
- написання та підтримка специфікацій.

Системний аналітик (Technical Leader) відповідальний за:

- координацію і контроль діяльності по дизайну, архітектурі та кодуванню;
- підтримку контролю версій;
- налаштування скрипту для авто-білдера та своєчасну збірку версій.

QA менеджер (QA manager) відповідальний за:

- організацію і контроль процесу тестування в проекті;
- планування тестування;
- участь в адаптації процесу розробки під проект, аналіз його якості;
- аналіз результатів тестування та якості продукту;
- участь в управлінні вимогами;
- участь в налаштуванні баг-трекінгової системи, повне простежування багів;
- контроль готовності нового білду для QA.

QA аналітик (QA Analyst) відповідальний за:

- підготовку тест-дизайну;
- написання тест-кейс специфікацій;
- проведення тестування;
- реєстрацію багів;
- простежування і перевірку багів;
- написання документації користувача.

Розробник (Developer) відповідальний за:

- розробку якісного коду;
- проведення модульного тестування;
- підтримку контролю версій;

- написання користувальницької документації, що відноситься до інсталяції та адміністрування.

Замовник (Customer) відповідальний за:

- своєчасний перегляд специфікацій та інших документів, що надсилаються, з метою затвердити документ, дати коментарі, виправити неточності тощо;
- внесення зауважень, дефектів, побажань в баг-трекінгову систему.
- своєчасний перегляд ПЗ після його поставки та надання коментарів.

Знаючи, яку роботу треба виконати кожному учаснику в певний момент часу, можна позбавити себе від можливих конфліктів при реалізації проекту.

Контрольні запитання

1. Визначити, які моделі використовуються в процесі розробки програмного забезпечення та вказати їх необхідність.
2. Перерахувати основні умови при виконанні роботи над проектом та обґрунтувати їх важливість.
3. Вказати, які проблеми можуть виникнути при розробці програмного продукту, та шляхи їх вирішення.
4. Навести приклади випадків, коли допускають суміщення ролей в процесі розробки ПЗ.
5. Які вимоги висуваються перед замовником проекту? Обґрунтувати їх доцільність.

Список літератури:

1. Команда проекта и ее роль в процессе разработки крупных [Електронний ресурс]. – Режим доступа: http://old.ci.ru/inform20_97/ast1.htm.
2. Роли в команде разработки ПО [Електронний ресурс]. – Режим доступа: <http://android.mobile-review.com/articles/52366/>
3. Разработка программного обеспечения [Електронний ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Разработка_программного_обеспечения.
4. Royce, Winston (1970), Managing the Development of Large Software Systems [Електронний ресурс]. – Режим доступа: <http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf>

5. Процесс разработки программного обеспечения [Электронный ресурс]. – Режим доступа: <http://www.williamspublishing.com/PDF/5-8459-0276-2/part1.pdf>.
6. Гленфорд Майерс, Том Баджетт, Кори Сандлер. Искусство тестирования программ, 3-е издание = The Art of Software Testing, 3rd Edition. – М.: «Диалектика», 2012. – 272 с. [Электронный ресурс]. – Режим доступа: <http://www.dialektika.com/books/978-5-8459-1974-8.html>.
7. Лайза Кристин, Джанет Грегори. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд = Agile Testing: A Practical Guide for Testers and Agile Teams. – М.: «Вильямс», 2010. – 464 с. [Электронный ресурс]. – Режим доступа: <http://www.twirpx.com/file/1626830/>.
8. Канер Сем, Фолк Джек, Нгуен Енг Кек. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений. – Киев: ДиаСофт, 2001. – 544 с. [Электронный ресурс]. – Режим доступа: http://lib.mdpu.org.ua/e-book/vstup/L/testirovanie_programmnogo_obespecheniia.pdf
9. Калбертсон Роберт, Браун Крис, Кобб Гэри. Быстрое тестирование. — М.: «Вильямс», 2002. — 374 с. [Электронный ресурс]. – Режим доступа: <http://lib.mdpu.org.ua/e-book/vstup/L/Kalbertson.pdf>
10. Сеницын С. В., Налютин Н. Ю. Верификация программного обеспечения. – М.: БИНОМ, 2008. – 368 с. [Электронный ресурс]. – Режим доступа: <http://window.edu.ru/resource/700/41700>.
11. Бейзер Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем. – СПб.: Питер, 2004. – 320 с. [Электронный ресурс]. – Режим доступа: <http://testingbooks.ru/тестирование-черного-ящика-технолог>.
12. Майк Кон. Scrum: гибкая разработка ПО = Succeeding with Agile: Software Development Using Scrum (Addison-Wesley Signature Series). – М.: «Вильямс», 2011. – С. 576. [Электронный ресурс]. – Режим доступа: <http://diamail.com.ua/book/4914.html>.
13. Роберт С. Мартин, Джеймс В. Ньюкирк, Роберт С. Косс. Быстрая разработка программ. Принципы, примеры, практика = Agile software development. Principles, Patterns, and Practices. – Вильямс, 2004. – 752 с. [Электронный ресурс]. – Режим доступа: <http://www.ozon.ru/context/detail/id/1573723/>.
14. James A. Highsmith. Agile Software Development Ecosystems. – Addison-Wesley Professional, 2002. [Электронный ресурс]. – Режим доступа: <http://www.amazon.com/Agile-Software-Development-Ecosystems-Highsmith/dp/0201760436>.

