Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Медіасистем та технологій
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)
(рівень вищої освіти)

Дослідження інструментів тестування веб додатків
(тема)

Виконав:
студентка 2 курсу, групи ТЕМВм-19-1

Бондаренко О.В.

Спеціальності 186 Видавництво та поліграфія

Тип програми Освітньо-професійна

Освітня програма
Технології електронних мультимедійних видань

Керівник проф. Манаков В.П.

Допускається до захисту
Зав. кафедри МСТ _____ Ткаченко В.П.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Медіасистем та технологій
Рівень вищої освіти другий (магістерський)
Спеціальність 186 Видавництво та поліграфія
Тип програми Освітньо-професійна
Освітня програма Технології електронних мультимедійних видань
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри МСТ_____
(підпис)
«26» жовтня 2020 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Бондаренко Олександр Васильович
(прізвище, ім'я, по батькові)
1. Тема роботи Дослідження інструментів тестування веб додатків

Затверджена наказом по університету від 23 жовтня 2020 р. № 1432 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 24 грудня 2020 р.

3. Вихідні дані до роботи Мета – дослідження інструментів тестування веб додатків та находження оптимальної платформи тестування
Об'єкт – процес тестування веб додатків.
Предмет – інструменти для тестування веб додатків.

4. Перелік питань, що потрібно опрацювати в роботі Вступ; 1 Аналіз стану проблеми та постановка задач дослідження; 2 Теоретичні дослідження; 3 Експериментальна частина дослідження; 4 Економічне обґрунтування проекту; Висновки.

5. Перелік графічного матеріалу із зазначенням обов'язкових креслеників, схем, плакатів, комп'ютерних ілюстрацій Титульна сторінка; Завдання на магістерську атестаційну роботу; Актуальність дослідження; Мета і задачі роботи; Об'єкт і предмет дослідження; Аналіз стану проблеми; Опис процесу проведення експерименту і отримання даних, Економічне обґрунтування проекту; Висновки.

6. Консультанти розділів роботи

Найменування розділу	Консультанта (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		Підпис	Дата
Основна частина	проф. Манаков В.П		
Економічна частина	проф. Полозова Т.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз стану проблеми і постановка задач дослідження	26.10.2020	викон.
2	Теоретичні дослідження	02.11.2020	викон.
3	Експериментальна частина дослідження	09.11.2020	викон.
4	Економічне обґрунтування проекту	16.11.2020	викон.
5	Оформлення пояснювальної записки	23.11.2020	викон.
6	Оформлення графічної частини	30.11.2020	викон.

Дата видачі завдання 26 жовтня 2020 р.

Студент _____ Бондаренко О.В.
(підпис)

Керівник роботи _____ проф. Манаков В.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 59 сторінок, 7 рисунків, 16 таблиць, 15 використаних джерел.

ТЕСТУВАННЯ, ВЕБ ДОДАТКИ, ІНСТРУМЕНТИ, ТЕСТ-КЕЙС, ТЕСТ МЕНЕДЖМЕНТ, УПРАВЛІННЯ ТЕСТУВАННЯМ, МЕТОД АНАЛІЗУ ІЄРАРХІЙ, КРИТЕРІЇ.

Метою атестаційної роботи є дослідження інструментів тестування веб додатків та находження оптимальної платформи для тестування.

Об’єктом дослідження даної роботи є процес тестування веб додатків. Предмет дослідження – інструменти для тестування веб додатків.

В процесі виконання роботи було проведено аналіз стану проблеми тестування веб додатків, аналітичний огляд літератури за темою атестаційної роботи, а ще проведено аналіз стану проблеми процесу находження оптимальї платформи тестування. Також було зроблено порівняльний аналіз існуючих програм тестування. Методом ієрархій була виявлена найбільш актуальна програма, на базі якої буде будуватися бібліотека. За результатами було сформовано критерії, які повинні бути присутні у бібліотеці тест-кейсів та розроблено рекомендацій щодо програм-бібліотек для збереження тест-кейсів.

Проведено економічне обґрунтування науково дослідної роботи та розраховано економічну ефективність даного дослідження.

РЕФЕРАТ

Пояснительная записка содержит 59 страниц, 7 рисунков, 16 таблиц, 15 использованных источников.

ТЕСТИРОВАНИЕ, ВЕБ ПРИЛОЖЕНИЯ, БИБЛИОТЕКА, ТЕСТ-КЕЙС, ТЕСТ МЕНЕДЖМЕНТ, УПРАВЛЕНИЕ ТЕСТИРОВАНИЕМ, МЕТОД АНАЛИЗА ИЕРАРХИЙ, КРИТЕРИИ.

Целью аттестационной работы является исследование инструментов тестирования веб приложений и нахождения оптимальной платформы для тестирования.

Объектом исследования данной работы является процесс тестирования веб приложений. Предмет исследования - инструменты для тестирования веб приложений.

В процессе выполнения работы был проведен анализ состояния проблемы тестирования веб приложений, аналитический обзор литературы по теме аттестационной работы, а еще проведен анализ состояния проблемы процесса нахождения оптимальи платформы тестирования. Также было сделано сравнительный анализ существующих программ тестирования. Методом иерархий была обнаружена самая актуальная программа, на базе которой будет строиться библиотека. По результатам был сформирован критерии, которые должны присутствовать в библиотеке тест-кейсов и разработаны рекомендации по программам библиотек для сохранения тест-кейсов.

Проведено экономическое обоснование научно-исследовательской работы и рассчитан экономическую эффективность данного исследования.

ABSTRACT

The explanatory note contains 59 pages, 7 pictures, 16 tables, 15 sources used.

TESTING, WEB APPLICATIONS, LIBRARY, TEST CASE, TEST MANAGEMENT, TEST MANAGEMENT, METHOD OF ANALYSIS OF HIERARCHIES, CRITERIA.

The purpose of the certification work is to study the tools for testing web applications and find the optimal platform for testing.

The object of research in this work is the process of testing web applications. The subject of research - tools for testing web applications.

In the process of performing the work, an analysis of the state of the problem of testing web applications, an analytical review of the literature on the topic of certification work, and an analysis of the state of the problem of finding the optimal testing platform. A comparative analysis of existing testing programs was also performed. The method of hierarchies revealed the most relevant program on the basis of which the library will be built. Based on the results, the criteria that should be present in the library of test cases were formed and recommendations for library programs for saving test cases were developed.

The economic substantiation of the research work is carried out and the economic efficiency of this research is calculated.

ЗМІСТ

С.

ВСТУП

В сьогоднішніх реаліях ІТ-спільнот представлена велика кількість якісних і корисних інструментів тестування, за допомогою функціоналу яких можна виконувати перевірки при розробці будь-якого програмного забезпечення.

Тестування веб-додатків - це комплекс послуг, який може включати в себе різні види тестування ПО.

Основна мета будь-якого тестування, в тому числі і тестування веб-додатків, виявити всі помилки в програмному забезпеченні і розробити рекомендації щодо їх запобігання в майбутньому.

У кожної компанії-розробника програмного забезпечення рано чи пізно виникає потреба у впровадженні автоматизації тестування. Зазвичай ставиться завдання автоматизувати велику кількість схожих дій, що виконуються в одній частині додатка, і регресивні тести, щоб звільнити ресурси ручного тестування на нові розробки.

У першому розділі атестаційній роботі був проведен аналіз проблеми тестування веб додатків та постановка мети і задач дослідження.

У другому розділі проведено теоретичне дослідження, розглянуті інструменти та платформи тестування, актуальність теми та життєвий цикл тестування.

У третьому розділі описана експериментальна частина роботи. Здійснюється порівнення інструментів тестування та пошук оптімальної платформи.

У четвертому розділі наведено економічне обґрунтування доцільності проведення даної науково-дослідної роботи в розділі «Економічна частина».порівнення їх а також перевірку результатів дослідження, використовуючи альтернативний метод дослідження.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз стану проблеми

Тестування визначаетсья я к складний процесс забеспечання якості, котрий охватуе усі етапи розробки програмного продукту в компнаии. В нього уходить вивчення процесів і визначення усіх умов та обстаин, котрі можуть вплинути на якість разробки і кінцевий продукт.

Забеспечення якості визначенно у стандарті ISO 9000:2005 «Системи менеджемнта якості. Основні положення і словник». Менеджмент якості в цьому стандарті уявлен як скоординована діяльність з керівництва та управління організацією стосовно до якості », а в примітці сказано, що він« зазвичай включає розробку політики і цілей в області якості, планування якості, управління якістю, забезпечення якості та поліпшення якості ».

Багаторазово підтверджений на практиці досвід підприємств - лідерів у сфері якості та положення авторитетного підручника з менеджменту показують, що якість продукції залежить від ряду зовнішніх і внутрішніх факторів:

До зовнішніх факторів належать:

– вимоги до якості (споживачі, прогрес, конкуренти);

– постачальники капіталу, трудових ресурсів, матеріалів, енергії, послуг;

– законодавство в сфері якості і робота державних органів.

Внутрішніми факторами забезпечення якості продукції служать:

– сучасна матеріальна база (інфраструктура, обладнання, матеріали, фінанси);

– застосування передових технологій;

– ефективний менеджмент (раціональна організація робіт і вміле управління підприємством в цілому і якістю зокрема);

– кваліфікований персонал, зацікавлений в хорошій роботі.

До сказаного можна додати, що кваліфікований і мотивований персонал і сучасна матеріальна база з передовою технологією визначають необхідну основу забезпечення якості продукції - базу якості. Причому, з усіх факторів, що впливають на якість, ключовим є людський фактор, а в ньому — зацікавленість працівників у хорошій роботі. Пояснюється це тим очевидним міркуванням, що незацікавлений працівник не буде добре працювати навіть на хорошому обладнанні, а зацікавлений шукатиме, знаходити і використовувати будь-які можливості для підвищення своєї кваліфікації і досягнення високої якості продукції.

Ефективний менеджмент з управлінням якістю доповнює базу якості, дозволяє реалізувати можливості, які створюються матеріальною базою і людським фактором. Бо не можна випускати продукцію, маючи тільки устаткування, матеріали і людей. Потрібно ще організувати роботу і налагодити управління.

Таким чином.

Принцип забезпечення якості продукції полягає в тому, щоб враховувати зовнішні фактори, які впливають на якість (постачальників, вимоги до якості, закони і державні органи) і створювати внутрішні фактори (матеріальну базу з передовою технологією, ефективний менеджмент з управлінням якістю і вмотивований, кваліфікований персонал). При цьому першорядну увагу потрібно приділяти мотивації персоналу.

Звідси стає ясно, як забезпечується якість продукції, а отже, — які заходи необхідні для його забезпечення.

Крім представленої плоскої схеми, принцип забезпечення якості продукції може бути показаний у вигляді просторової «Моделі якості», на якій показані не тільки склад і взаємозв’язок чинників, необхідних для забезпечення якості, але і взаємодію цих чинників і результат цієї взаємодії — підвищення якості продукції.

Тестування веб-додатків - це комплекс послуг, який може включати в себе різні види тестування ПО.

Основна мета будь-якого тестування, в тому числі і тестування веб-додатків, виявити всі помилки в програмному забезпеченні і розробити рекомендації щодо їх запобігання в майбутньому.

1.2 Постановка мети і задач дослідження

Метою магістерської атестаційної роботи є дослідження інструментів тестування веб додатків. Вибір найбільш оптимального інструмента для тестування веб-додатків.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– аналіз стану проблеми тестування веб-додатків;

– дослідження предметної області, вивчення видів тестування веб-додатків;

– огляд і аналіз існуючих інструментів тестування;

– сформувати список критеріїв, які важливі для оцінки якості тестування верстки;

– провести аналіз особливостей проектування тестів для перевірки веб-додатку;

– розробити методику оцінки якості інструмента тестування;

– провести оцінку якості результату.

Предмет дослідження – визначення найбільш ефективних функцій у існуючих інструментах.

Таким чином, в ході дослідження пропонується проаналізувати існуючі інструменти тестування, провести порівняльний аналіз можливостей даних інструментів, розробити прототип программи для тестування по отриманим результатам.

2 ТОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Види тестування

Для того щоб краще розуміти підходи до тестування програмного забезпечення, потрібно, звичайно ж, знати, які види і типи тестування в принципі бувають. Давайте почнемо з розгляду основних типів тестування, які визначають високорівневу класифікацію тестів.

Найвищим рівнем в ієрархії підходів до тестування буде поняття типу, яке може охоплювати відразу кілька суміжних технік тестування. Тобто, одного типу тестування може відповідати кілька його видів. Розглянемо, для початку кілька типів тестування, які відрізняються знанням внутрішнього устрою об'єкта тестування. Процес або результат формування необхідних властивостей і характеристик продукції в міру її створення, а також - підтримку цих характеристик при зберіганні, транспортуванні та експлуатації продукції.

Забезпечення якості визначено в стандарті ISO 9000: 2005 «Системи менеджменту якості. Основні положення і словник »як« частина менеджменту якості, зосереджена на створенні впевненості в тому, що вимоги до якості будуть виконані».

Менеджмент якості в цьому ж стандарті представлений як «скоординована діяльність з керівництва та управління організацією стосовно до якості», а в примітці сказано, що він «зазвичай включає розробку політики і цілей в області якості, планування якості, управління якістю, забезпечення якості та поліпшення якості».

Необхідними перевірками є:

– перехід по посиланнях;

– перевірка користувальницьких форм (валідація полів, обов’язкові / необов’язкові поля, повідомлення про помилки при неправильному введенні, додавання коментарів в блог, зворотний зв’язок);

– пошук і покупка товару, оформлення замовлення;

– звірка переданого замовником контенту з наявним на сайті;

– перевірка можливої авторизації / реєстрації;

– додавання, видалення і редагування даних користувачів, товарів і замовлень.

При структурному тестуванні програма розглядається як "білий ящик" тобто її текст відкритий для користування. Відбувається перевірка логіки програми. Повним тестуванням в цьому випадку буде таке, яке приведе до перебору всіх можливих шляхів на графі передач управління програми (її керуючому графові). Навіть для середніх по складності програм числом таких шляхів може досягати десятків тисяч. Якщо обмежитися перебором тільки лінійних незалежних шляхів, то і в цьому випадку вичерпне структурне тестування практично неможливо, т.к. неясно, як підбирати тести, щоб забезпечити "покриття" всіх таких шляхів. Тому при структурному тестуванні необхідно використати інші критерії його повноти, що дозволяють досить просто контролювати їх виконання, але не дають гарантії повної перевірки логіки програми.

Usability тестування.

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

навігаційне тестування сайту. Чи всі сторінки, кнопки і поля на них, зрозумілі у використанні, доступ до головної сторінки і меню з усіх інших сторінок можливий, навігація проста і інтуїтивно зрозуміла;

– тестування контенту. Відсутність граматичних / орфографічних помилок, контент інформативний і структурований, зображення і заголовки мають відповідні розміри і розміщені правильно;

– зручність використання. Чи зрозуміла структура веб-додатки, яке враження справляє і чи є зайві компоненти на сторінках;

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

Тестування сумісності (конфигурационное тестування).

Тип нефункціонального тестування програмного забезпечення, що дозволяє перевірити, чи може ПО працювати на іншому обладнанні, операційних системах, додатках, мережних середовищах або мобільних пристроях:

– крос-платформенне тестування сайту. Деякі функції можуть мати проблеми з певними операційними системами, тому необхідно перевіряти роботу програми в різних версіях Windows, Unix, Mac, Linux, Solaris і ін.;

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

– перегляд на мобільних пристроях. Незважаючи на перевірку роботи веб-додатків в різних дозволах на комп’ютері, найчастіше помилки на мобільних пристроях залишаються непоміченими. Отже, настійно рекомендується перевіряти коректне відображення і роботу вашого веб-додатки на мобільних пристроях різних операційних пристроїв, а також на планшетах;

– тестування БД. Необхідно перевірити правильність здійснення зв’язку з сервером, перевірити сумісність сервера з ПО, апаратними засобами, базою даних і мережею. Також потрібно перевірити що відбувається при перериванні якої-небудь дії, під час наступного з’єднання з сервером під час виконання операцій.

Тестування продуктивності.

Методика нефункціонального тестування, для вимірювання таких параметрів системи як чуйність і стабільність, при різних навантаженнях. Дозволяє досліджувати швидкість швидкодії сайту та можливості масштабованості додатки, наприклад, при додаванні нових користувачів.

Проводиться з метою з’ясувати яке навантаження сайт здатний витримати. Тестування продуктивності вимірює атрибути якості системи, такі як масштабованість, надійність і використання ресурсів.

Тестування навантаження - це метод тестування продуктивності, при якому реакція системи вимірюється в різних умовах навантаження. Відповідає за реакцію веб-додатки при збільшенні робочого навантаження. Навантажувальні випробування проводяться для нормальних і пікових навантажень (одночасна купівля товару або авторизація на сайті великої кількості користувачів).

Підхід навантажувального тестування:

- оцінити критерії прийнятності продуктивності;
- визначити критичні сценарії;
- модель робочого навантаження;
- визначте цільові рівні навантаження;
- дизайн тестів;
- виконати тести;
- проаналізуйте результати.

Завдання навантажувального тестування: час відгуку, пропускна здатність, утилізація ресурсів, максимальна призначена для користувача навантаження, бізнес-метрики.

Стрес-тестування (Stress Testing) перевіряє систему на її стійкість і обробку помилок в умовах надзвичайно високого навантаження (оцінює як система працює в екстремальних умовах, за межами обмежень і лімітів). Стрес-

тестування проводиться, щоб переконатися, що система не буде аварійно завершувати роботу в критичних ситуаціях.

Тестуваня стабільності / надійності (Stability / Reliability Testing) - тип тестування програмного забезпечення, який перевіряє, чи може програмне забезпечення виконувати безвідмовну роботу протягом певного періоду часу в вказаному середовищі.

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

Тестування паралелізму (Parallel Testing) - тип тестування програмного забезпечення, який перевіряє кілька додатків або подкомпонентов однієї програми одночасно, щоб скоротити час тестування. При паралельному тестуванні тестувальник запускає дві різні версії програмного забезпечення одночасно з одним і тим же введенням. Мета полягає в тому, щоб з’ясувати, чи ведуть себе колишня система і нова система однаково або по-різному.

2.2 Концепція тестування

Забеспечення якості це процес або результат формування необхідних властивойстей і характеристик продукції в міру її створення, а також підтримку цих характеристик при зберіганні. Існує два вида типа тестування.

Ручному тестуванні (manualtesting) тестувальники вручну виконують тести, не використовуючи ніяких засобів автоматизації. Ручне тестування - самий низькорівневий і простий тип тестування, які не потребують великої кількості додаткових знань.

Проте, перед тим як автоматизувати тестування будь-якої програми, необхідно спочатку виконати серію тестів вручну. Мануальне тестування вимагає значних зусиль, але без нього ми не зможемо переконатися в тому, чи

можлива автоматизація в принципі. Один з фундаментальних принципів тестування говорить: 100% автоматизація неможлива. Тому, ручне тестування - необхідність.

Міфи про ручному тестуванні:

– хто завгодно може провести ручне тестування. Ні, виконання будь-якого виду тестування вимагає спеціальних знань і професійної підготовки;

– автоматизоване тестування могутніше ручного. Повна автоматизація неможлива. Необхідно використовувати також і ручне тестування;

– ручне тестування - це просто. Тестування може бути дуже непростим заняттям. Проведення тестування для перевірки максимально можливої кількості шляхів виконання з використанням мінімального числа тест-кейсів вимагає серйозних аналітичних навичок.

Автоматизоване тестування передбачає використання спеціального програмного забезпечення (крім тестованого) для контролю виконання тестів і порівняння очікуваного фактичного результату роботи програми. Цей тип тестування допомагає автоматизувати часто повторювані, але необхідні для максимізації тестового покриття завдання.

Деякі завдання тестування, такі як низкоуровневое регресійне тестування, можуть бути трудомісткою і вимагають багато часу якщо виконувати їх вручну. Крім того, мануальное тестування може недостатньо ефективно знаходити деякі класи помилок. У таких випадках автоматизація може допомогти заощадити час і зусилля проектної команди.

Після створення автоматизованих тестів, їх можна в будь-який момент запустити знову, причому запускаються і виконуються вони швидко і точно. Таким чином, якщо є необхідність частого повторного прогону тестів, значення автоматизації для спрощення супроводу проекту і зниження його вартості важко переоцінити. Адже навіть мінімальні патчі і зміни коду можуть стати причиною появи нових багів.

Існує кілька основних видів автоматизованого тестування:

– автоматизація тестування коду (Code-driven testing) - тестування на рівні програмних модулів, класів і бібліотек (фактично, автоматичні юніт-тести);

– автоматизація тестування графічного інтерфейсу користувача (Graphical user interface testing) - спеціальна програма (фреймворк автоматизації тестування) дозволяє генерувати користувальницькі події - натискання клавіш, кліки мишкою, і відслідковувати реакцію програми на ці дії - чи відповідає вона специфікації;

– автоматизація тестування API (ApplicationProgrammingInterface) - програмного інтерфейсу програми. Тестуються інтерфейси, призначені для взаємодії, наприклад, з іншими програмами або з користувачем. Тут знову ж таки, як правило, використовуються спеціальні фреймворки.

Для складання автоматизованих тестів, QA-фахівець повинен вміти програмувати. Автоматичні тести - це повноцінні програми, просто призначені для тестування.

Коли, що і як автоматизувати і автоматизувати взагалі - дуже важливі питання, відповіді на які повинна дати команда розробки. Вибір правильних елементів програми для автоматизації в великій мірі буде визначати успіх автоматизації тестування в принципі. Потрібно уникати автоматизації тестування ділянок коду, які можуть часто змінюватися.

Порівнення ручного та автаматизованого тестування.

Як ручне, так і автоматизоване тестування можуть використовуватися на різних рівнях тестування, а також бути частиною інших типів і видів тестування.

Автоматизація зберігає час, сили і гроші. Одного разу автоматизований тест можна запускати знову і знову, докладаючи мінімум зусиль.

Вручну можна протестувати практично будь-який додаток, в той час як автоматизувати варто тільки стабільні системи. Автоматизоване тестування

використовується головним чином для регресії. Крім того, деякі види тестування, наприклад, ad-hoc чи дослідне тестування можуть бути виконані тільки вручну.

Мануальне тестування може бути повторюваним і нудним. У той же час, автоматизація може допомогти цього уникнути - за вас все зробить комп’ютер.

Таким чином, на реальних проектах часто використовується комбінація ручного і автоматизованого тестування, причому рівень автоматизації буде залежати як від типу проекту, так і від особливостей постановки виробничих процесів в компанії.

2.3 Життєвий цикл тестування програмного забеспечення

Тестування - це не ізольований процес. Це частина моделі життєвого циклу програмного забезпечення (Software Development Life Cycle, SDLC). Саме тому вибір засобів і методик тестування буде прямо залежати від обраної моделі розробки. У цьому розділі ми розглянемо найбільш часто застосовуються підходи до розробки програмного забезпечення, а також популярні сьогодні методології та практики, такі як Agile і Scrum.

Цикл розробки (рис. 2.1) пропонує шаблон, використання якого полегшує проектування, створення і випуск якісного програмного забезпечення. Це методологія, що визначає процеси і засоби, необхідні для успішного завершення проекту.

Хоча реалізація принципів побудови моделі життєвого циклу для різних компаній може істотно відрізнятися, існують стандарти, такі як ISO / IEC 12207, що визначають прийняті практики розробки і супроводу програмного забезпечення.

Мета використання моделі життєвого циклу - створити ефективний, економічно вигідний і якісний програмний продукт.

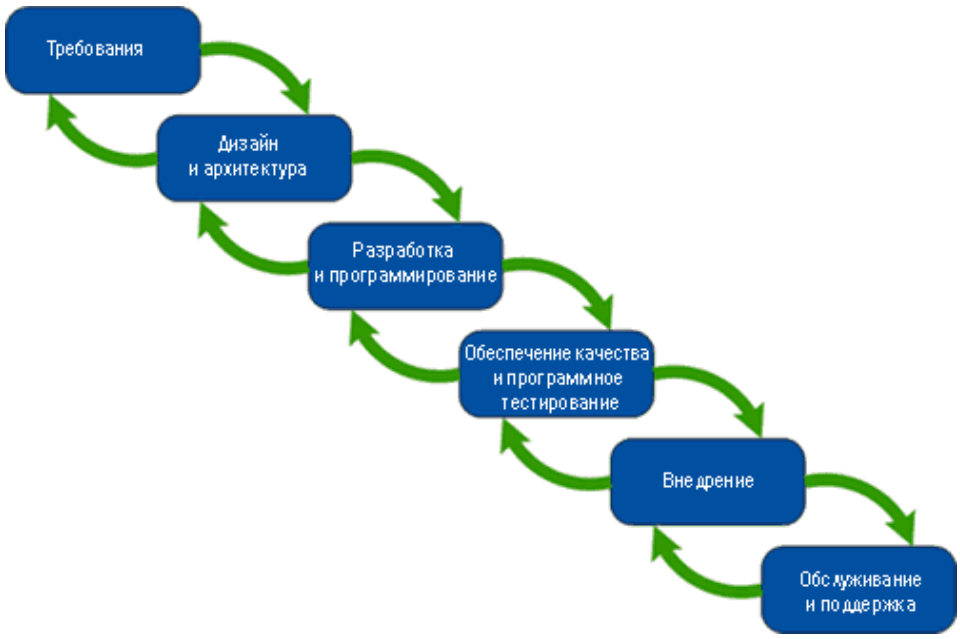

Рисунок 2.1 – Життєвий цикл програмного забеспечення

Життєвий цикл розробки ПЗ починається зі стадії аналізу, під час якого учасники процесу обговорюють вимоги, що пред'являються до кінцевого продукту. Мета цієї стадії - визначення детальних вимог до системи. Крім цього, необхідно переконатися в тому, що всі учасники правильно зрозуміли поставлені завдання і те, як саме кожна вимога буде реалізовано на практиці.

Найчастіше, в обговоренні беруть участь також і фахівці з тестування, які вже на стадії розробки вимог можуть вносити власні побажання і, при необхідності, коригувати процес.

Залежно від обраної моделі розробки, можуть відрізнятися підходи до визначення моменту переходу з однієї стадії на іншу. Наприклад, в каскадної або V-моделі стадія аналізу вимог закріплюється в документі - специфікації вимог до програмного забезпечення (Software Requirement Specification, SRS), оформлення якого повинне бути закінчене до переходу на наступну стадію.

Таким чином, цей етап передбачає збір вимог щодо розроблюваного програмного забезпечення, іхсістематізацію, документування, аналіз, а також виявлення та розв'язання суперечностей.

На стадії проектування (званої також стадією дизайну і архітектури) програмісти і системні архітектори, керуючись вимогами, розробляють високорівнева дизайн системи.

Різноманітні технічні питання, що виникають в процесі проектування, обговорюються з усіма зацікавленими сторонами, включаючи замовника. Визначаються технології, які будуть використовуватися в проекті, завантаження команди, обмеження, тимчасові рамки і бюджет. Відповідно до уточнених вимог вибираються найбільш підходящі проектні рішення.

Затверджений дизайн системи визначає перелік розроблюваних програмних компонентів, взаємодія з третіми сторонами, функціональні характеристики програми, які використовуються бази даних і багато іншого. Дизайн, як правило, закріплюється окремим документом - дизайн-специфікацією (Design Specification Document, DSD).

На цьому етапі для спрощення візуалізації процесу проектування використовуються так звані нотації - схематичне вираження характеристик разрабативемой системи. Основні використовувані нотації:

– блок-схеми;
– ER-діаграми;
– UML-діаграми;
– макети - наприклад, намальований в фотошопі прототип сайту.

Після того як вимоги та дизайн продукту затверджені, відбувається перехід до наступної стадії життєвого циклу - безпосередньо розробці. Тут починається написання програмістами коду програми відповідно до раніше визначених вимог.

Системні адміністратори налаштовують програмне оточення, front-end програмісти розробляють призначений для користувача інтерфейс програми і логіку її взаємодії з сервером.

Крім того, програмісти пишуть Unit-тести для перевірки правильності роботи коду кожного компонента системи, проводять рев'ю написаного коду,

створюють білди і розгортають готове ПО в програмному середовищі. Цей цикл повторюється до тих пір, поки всі вимоги не будуть реалізовані.

Тестировщики займаються пошуком дефектів в програмному забезпеченні і порівнюють описане в вимогах поведінку системи з реальним.

У фазі тестування виявляються пропущені при розробці баги. При виявленні дефекту, тестувальник складає звіт про помилку, який передається розробникам. Останні його виправляють, після чого тестування повторюється - але на цей раз для того, щоб переконатися, що проблема була виправлена, і саме виправлення не стало причиною появи нових дефектів в продукті.

Тестування повторюється до тих пір, поки не будуть досягнуті критерії його закінчення.

Види, методи і техніки тестування ми докладно розглянемо далі в цьому посібнику. Коли програма протестована і в ній більше не залишилося серйозних дефектів, приходить час релізу і передачі її кінцевим користувачам.

Після випуску нової версії програми в роботу включається відділ технічної підтримки. Його співробітники забезпечують зворотний зв’язок з користувачами, їх консультування та підтримку.

У разі виявлення користувачами тих чи інших пост-релізних багів, інформація про них передається у вигляді звітів про помилки команді розробки, яка, в залежності від серйозності проблеми, або негайно випускає виправлення (т.зв. hot-fix), або відкладає його до наступної версії програми.

Крім того, команда технічної підтримки допомагає збирати і систематизувати різні метрики - показники роботи програми в реальних умовах.

2.4 Актуальність етапу тестування

Спосіб мислення фахівця з тестування повинен відрізнятися від способу мислення розробника. Взагалі кажучи, програмісти цілком здатні самі тестувати

як власноруч написаний код, так і функціональність системи, над якою вони працюють. Але тестування не дарма проводиться незалежними фахівцями - люди схильні неправильно оцінювати результати власної роботи. Тому тестувальник, що володіє певним ступенем незалежності практично завжди буде ефективніше знаходити дефекти і збої в системі, ніж програміст. Тут потрібно зробити застереження, що незалежність не може бути заміною знань - певні завдання набагато простіше і швидше виконати програмістам, наприклад, провести модульне тестування, яке вимагає розуміння внутрішнього устрою програми.

Всього виділяють чотири рівні незалежності - від низького до високого:

– тести для програми розробляються і проводяться людиною, який є її автором;

– тести розробляються і виконуються іншими людьми (наприклад, іншим розробником);

– тести розроблені представниками іншої організаційної групи (наприклад, з відділу тестування) або спеціалізованими тестувальниками (наприклад, фахівцями з тестування продуктивності або безпеки);

– тести розробляються і виконуються фахівцями з іншої організації (наприклад, аутсорсингової або аудиторської).

В силу своєї діяльності, тестувальники займаються оцінкою чужої роботи, знаходять в ній недоліки, що часто сприймається як деструктивна діяльність, незважаючи на те, що її результатом стає виправлення помилок і покращення загальної якості продукту. Хороший тестувальник повинен володіти рядом особистих і професійних якостей: він повинен бути цікавим, критичним, уважним до деталей, комунікативним, зберігати професійний песимізм і мати достатній досвід для побудови припущень про можливі джерела помилок

Тестувальник, на відміну від програміста, головна мета якого - створити працюючий продукт, повинен вміти знайти всі закладені в цьому продукті недоліки. А для цього ми повинні, перш за все, сконцентруватися на тому, що

може піти не так. Дослідження показали, що, якщо людина, яка тестує програму, сприймає її як працює правильно, він знайде менше помилок, ніж той, хто буде впевнений в наявності в ній безлічі недоліків. Тому, тестувальник повинен завжди пам’ятати про те, що "Software has bugs".

І наостанок, ще один важливий момент: при написанні звіту про дефект будьте об’єктивні і ні в якому разі не вказуйте явно або неявно на його винуватця, навіть якщо він цілком того заслуговує. Пам’ятайте, Вам з програмістами ще працювати, не варто псувати відносини.

Є кілька простих порад для поліпшення комунікації з колегами:

– пам’ятайте про те, що всі ви працюєте над одним проектом і йдете до однієї мети - створення якісного і затребуваного продукту;

– оформляйте результати своєї роботи в нейтральному тоні, сфокусуйтеся на фактах;

– поставте себе на місце інших і спробуйте зрозуміти причини їх поведінки;

– завжди переконуйтеся в тому, що інша людина зрозумів Вас, а Ви його.

Спосіб мислення фахівця з тестування повинен відрізнятися від способу мислення розробника. Взагалі кажучи, програмісти цілком здатні самі тестувати як власноруч написаний код, так і функціональність системи, над якою вони працюють. Але тестування не дарма проводиться незалежними фахівцями - люди схильні неправильно оцінювати результати власної роботи. Тому тестувальник, що володіє певним ступенем незалежності практично завжди буде ефективніше знаходити дефекти і збої в системі, ніж програміст. Тут потрібно зробити застереження, що незалежність не може бути заміною знань - певні завдання набагато простіше і швидше виконати програмістам, наприклад, провести модульне тестування, яке вимагає розуміння внутрішнього устрою програми.

Всього виділяють чотири рівні незалежності - від низького до високого:

– тести для програми розробляються і проводяться людиною, який є її автором;

– тести розробляються і виконуються іншими людьми (наприклад, іншим розробником);

– тести розроблені представниками іншої організаційної групи (наприклад, з відділу тестування) або спеціалізованими тестувальниками (наприклад, фахівцями з тестування продуктивності або безпеки);

– тести розробляються і виконуються фахівцями з іншої організації (наприклад, аутсорсингової або аудиторської).

В силу своєї діяльності, тестувальники займаються оцінкою чужої роботи, знаходять в ній недоліки, що часто сприймається як деструктивна діяльність, незважаючи на те, що її результатом стає виправлення помилок і покращення загальної якості продукту. Хороший тестувальник повинен володіти рядом особистих і професійних якостей: він повинен бути цікавим, критичним, уважним до деталей, комунікативним, зберігати професійний песимізм і мати достатній досвід для побудови припущень про можливі джерела помилок

Тестувальник, на відміну від програміста, головна мета якого - створити працюючий продукт, повинен вміти знайти всі закладені в цьому продукті недоліки. А для цього ми повинні, перш за все, сконцентруватися на тому, що може піти не так. Дослідження показали, що, якщо людина, яка тестує програму, сприймає її як працює правильно, він знайде менше помилок, ніж той, хто буде впевнений в наявності в ній безлічі недоліків. Тому, тестувальник повинен завжди пам’ятати про те, що "Software has bugs".

І наостанок, ще один важливий момент: при написанні звіту про дефект будьте об’єктивні і ні в якому разі не вказуйте явно або неявно на його винуватця, навіть якщо він цілком того заслуговує. Пам’ятайте, Вам з програмістами ще працювати, не варто псувати відносини.

Є кілька простих порад для поліпшення комунікації з колегами:

– пам’ятайте про те, що всі ви працюєте над одним проектом і йдете до однієї мети - створення якісного і затребуваного продукту;

– оформляйте результати своєї роботи в нейтральному тоні, сфокусуйтеся на фактах;

– поставте себе на місце інших і спробуйте зрозуміти причини їх поведінки;

– завжди переконуйтеся в тому, що інша людина зрозумів Вас, а Ви його.

Як бачимо, чим пізніше дефект був виявлений, тим дорожче обійдеться його виправлення і тим більше зусиль для цього буде потрібно. Крім того, як ми пам’ятаємо, дефекти, закладені в систему на ранніх рівнях проектування особливо підступні - їх важко відстежити і правильно інтерпретувати. Висновок напрошується сам собою: чим раніше в життєвому циклі програми почнеться тестування, тим більшою мірою ми можемо бути впевнені в її якості.

Більшість фахівців сходяться в думці, що тестування потрібно починати ще на етапі складання вимог до системи. Хоча тут все буде залежати від обраної моделі розробки (про них ми поговоримо трохи пізніше). Наприклад, в каскадної моделі тестування проводиться на спеціально виділеному для нього етапі. Ітераційна ж модель дозволяє здійснювати тестування практично паралельно з розробкою нового функціоналу.

На різних етапах життєвого циклу ПЗ тестування проводиться в різних формах:

– на етапі визначення вимог: їх аналіз та верифікація також можуть вважатися тестуванням;

– контроль процесу проектування на етапі розробки дизайну системи - це теж форма тестування;

– як уже згадувалося, розробники теж беруть участь в тестуванні на рівні модульного тестування.

Важче визначити критерій закінчення тестування, оскільки, згідно з принципами тестування, ми ніколи не можемо бути впевнені в тому, що програма на 100% вільна від дефектів. Тому використовуються інші умови:

– граничні терміни, встановлені заздалегідь;

– виконання всіх передбачених тест-кейсів;

– досягнення певного рівня тестового покриття;

– коли після певного моменту, ми практично не знаходимо нових багів або критичних дефектів;

– рішення менеджменту.

2.5 Інструменти для тестування веб додатків

Кожен проект унікальний і у кожної команди свої запити. Але всіх нас об’єднує бажання працювати з якісними інструментами, які економлять час.

Я проаналізував перевірені часом і нові системи управління тестуванням, які зараз популярні на ринку. Вибрали функції, які повинні бути в ідеальній Test Management System, порівняли можливості продуктів і вивчили відгуки користувачів.

Що бажають от інструмента тестування:

– зручна установка і підтримка;

– створення та управління проектами;

– створення користувачів і ролей користувачів;

– зручна інтеграція з автоматичними тестами;

– створення тест-плану;

– створення тест-кейса;

– прогін тест-кейса;

– зрозуміла система звітності;

– вбудована система баг-трекінгу;

– можливість інтеграції з іншими інструментами.

Вирішити задачу створення єдиної системи для роботи з усією документацією проекту можна декількома способами.

Найдешевший спосіб - не морочитися і вибрати Google Docs для матриці трасуванню, а дефекти вести в open-source баг-трекері.

Інший спосіб - використовувати одну з популярних TMS'ок, інтегровану з баг-трекером компанії.

Next-level спосіб - вибрати Test Management System, виходячи із специфіки проектів, обсягів завдань, типів документації і використовуваних видів тестування.

Дуже важливо підійти до питання вибору TMS відповідально, адже для компанії, ціна помилки може виявитися високою.

Популярні інструменти:

- Test Link;
- qTest;
- Test IT;
- PractiTest;
- Zephyr.

3 ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА ДОСЛІДЖЕННЯ

У цьому розділі було проведннo аналіз вихідних даних, в процесі котрого треба виявити проблеми обьектів дослідження.

3.1 План експериментальних досліджень

Планування експерименту - комплекс заходів, спрямованих на ефективну постановку дослідів. Основна мета планування експерименту - досягнення максимальної точності вимірювань при мінімальній кількості проведених дослідів і збереженні статистичної достовірності результатів.

Планування експерименту застосовується при пошуку оптимальних умов, побудові інтерполяційних формул, виборі значущих чинників, оцінки якості та уточнення констант теоретичних моделей і ін.

Отже, план проведення експерименту наступний:

- встановлення мети експерименту;
- уточнення умов проведення експерименту;
- виявлення і вибір вхідних та вихідних параметрів;
- встановлення потрібної точності результатів вимірювань;
- складання плану та проведення експерименту;
- статистична обробка результатів експерименту;
- пояснення отриманих результатів;
- висновки.

3.2 Проведення дослідження

На початку експерименту було визначено мету та задачі атестаційної роботи. Оскільки, мета дослідження – це запланований результат, на досягнення

якого спрямоване дослідження, то відповідно до теми роботи метою дослідження є пошук оптимальної платформи тестування.

Після цього, треба проаналізувати найпоширеніші існуючи програми для тестування. Було розглянуто п’ять програм: Test Link, Test IT, Zephyr, qTest та PractiTest.

Test Link (рис. 3.1) – один з небагатьох open-source інструментів управління тестуванням, який доступний на ринку. Це веб-інструмент з типовими функціями, такими як управління вимогами, створення і супровід тест сьютів, прогін тестів, відстеження помилок, інтеграція з відомими баг-трекінгові системами.

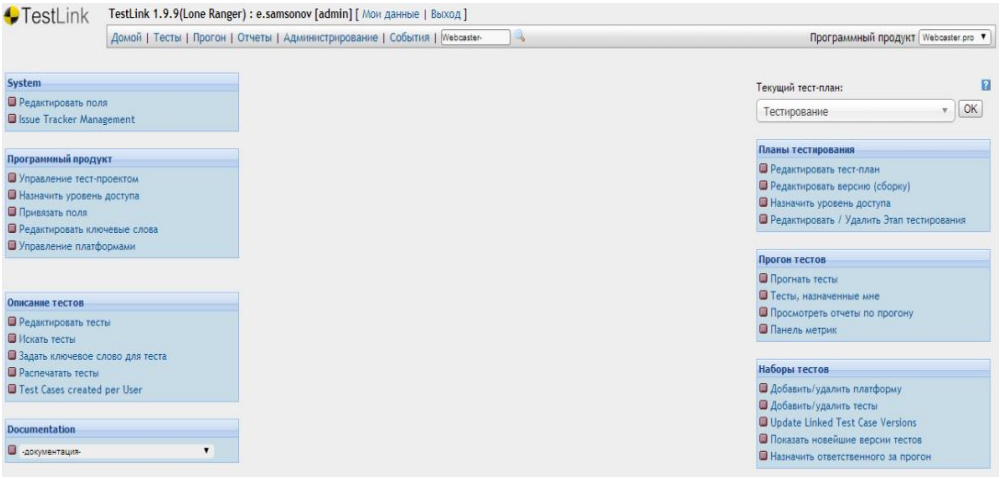

Рисунок 3.1 – Інтерфейс TestLink

Для відстеження прогресу проекту доступні звіти і діаграми, а додаткові функції включають тегірованіе ключовими словами, вказівка вимог і журнал подій.Код проекту часто оновлюється і доповнюється.

Можливості:

– управління вимогами;

– специфікація - визначення тест кейсів шляхом угруповання в різні набори тестів;

– призначення виконання тест сьютів на рівні збірки;

– централізоване управління користувачами і ролями;

– кастомізація настроюються користувачем полів.

Test Collab (рис. 3.2) – Це сервіс з корисним набором функцій, який необхідний для швидкого старту використання інструменту. Плюс, трохи приємних бонусів: зручний інтерфейс, кастомизация фільтрів і полів, time-трекер для кожного члена команди, і внутрисистемное спілкування.

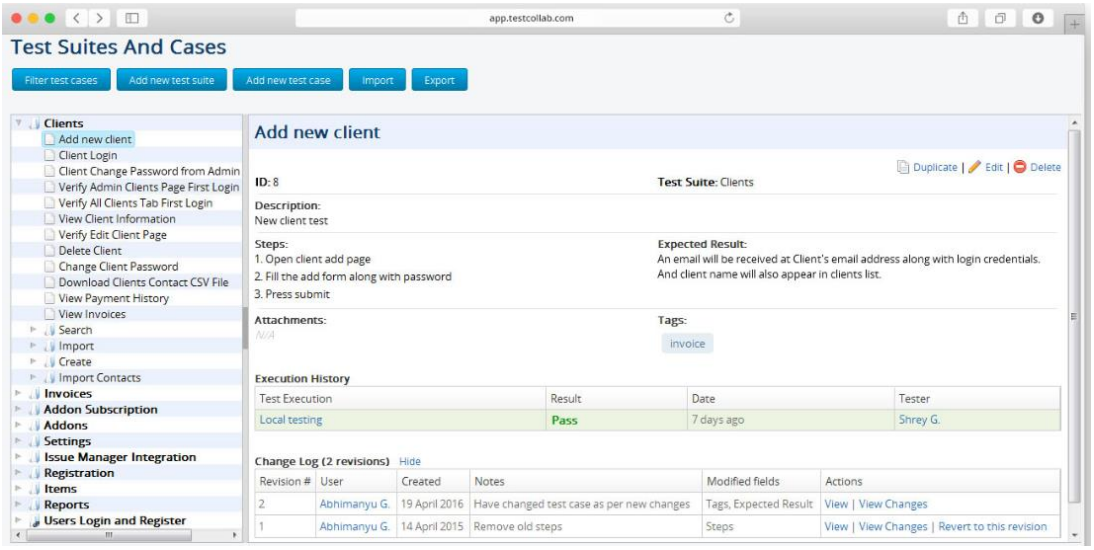

Рисунок 3.2 – Інтерфейс Test Collab

Можливості:

– угруповання тест кейсів по етапах або білд;

– управління вимогами;

– перевикористання тестів;

– налаштування спринтів;

– звіти за результатами тестів;

– коментування тест кейсів.

Інтеграція з JIRA, Redmine, Bugzilla, Asana, Trello, YouTrack, GitHub.

Радує: широка настройка поштових повідомлень (з можливістю редагувати шаблони листів); об'єднання тестранов в тест-плани; приємний інтерфейс з

підказками (які можна відключити); функція повторного використання кроків (reusable steps); можливість коментування кейса при редагуванні; теги; вибір типу форматування тексту (Markdown, HTML / WYSIWYG, Plain Text); додавання призначених для користувача полів; налаштовуються шаблони для вибірки кейсів в тестран; вбудовані вимоги; ціна (для невеликих проектів є безкоштовний план з 200 кейсами і 400 виконаними кейсами).

Не тішить: не можна редагувати системні поля (зокрема Priority); немає можливості зберегти відображаються поля в списках (при оновленні сторінки скидаються за замовчуванням); є невеликі огріхи в відображенні елементів UI; немає друкованих форм ні для кейсів ні для звітів (напевно, зараз їх рідко використовують, але інші інструменти пропонують таку функцію); дані на загальному дашборда оновлюються з затримкою; експорт кейсів тільки для одного тестового набору (при цьому вкладені набори обробляє тільки експорт в XML); незручна вставка зображень в кроки кейса (через посилання); немає свого багтрекер.

Zephyr (рис. 3.3) - це плагін для всім відомої JIRA, який інтегрує тестування в проектний цикл, дозволяючи вам відстежувати якість програмного забезпечення та приймати рішення в стилі go / no-go.

Тест кейси можуть створюватися, виконуватися і трекаться так само, як і будь-які інші завдання в JIRA. Для більш оптимальної фіксації процесу тестування є інтеграція з інструментами управління якістю, автоматизації, безперервної інтеграції та аналітики.

Крім того, у продукту швидко відповідає тих підтримка.

Можливості:

- посилання на user stories, завдання, вимоги, дефекти;
- конфігурації деплоя: в хмарі, на сервері, в дата-центрі;
- розширена інформація на дашборда аналітики і devops;
- інтеграція з jira, confluence, selenium, jenkins і bamboo.

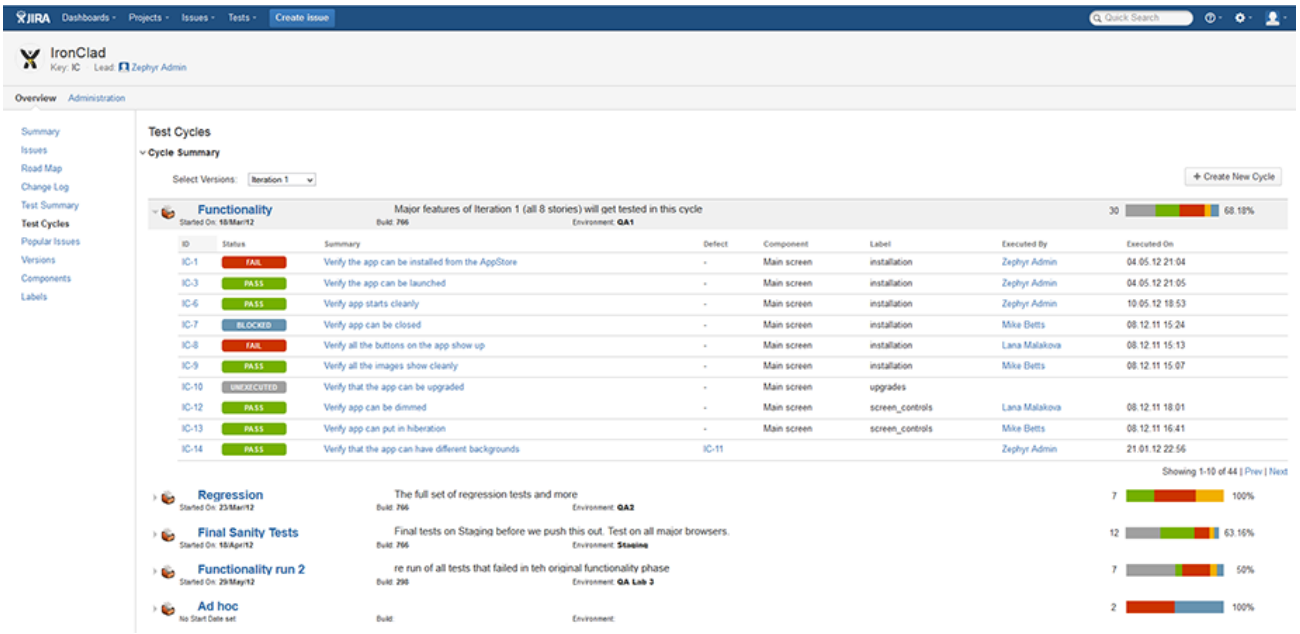

Рисунок 3.3 – Інтерфейс Zephyr

qTest (рис. 3.4) - Інструмент корисний не тільки тестувальникам, але і всій команді. Інтерфейс qTest нативної зрозумілий, мануали прості в освоєнні. Це дозволяє швидко і ефективно створювати, організовувати і управляти тест кейсами.

Згідно з аналізом ринку, qTest є одним з найбільш швидкозростаючих рішень для управління тестуванням серед команд Agile Development.

Можливості:

– планування тестів
– створення та управління вимогами;
– інтуїтивний drag-n-drop інтерфейс;
– комплексна матриця трасуванню;
– наочні звіти з докладними графіками;
– інтеграція зі сторонніми інструментами баг-трекінгу;
– детальний контроль доступу користувачів;
– хмарний інструмент інтеграції з jira.

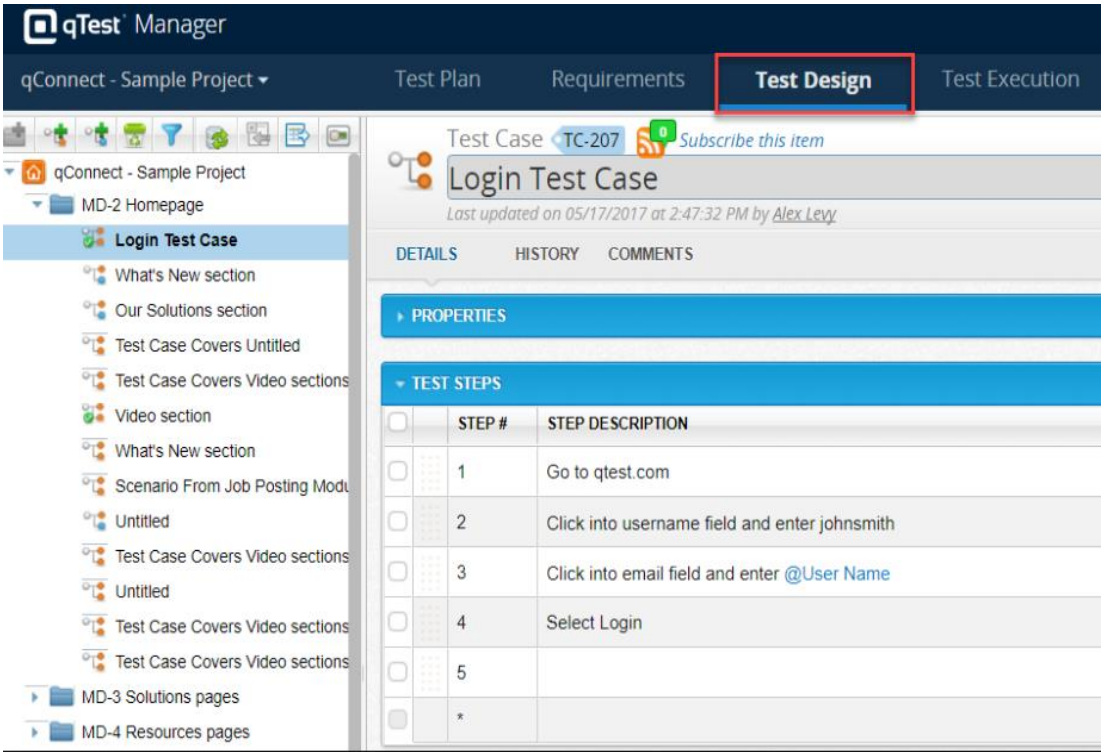

Рисунок 3.4 – Інтерфейс qTest

PractiTest (рис. 3.5) - це комплексне засіб управління тестами. Воно забезпечує повну наочність процесу тестування і більш глибоке розуміння результатів тестування.

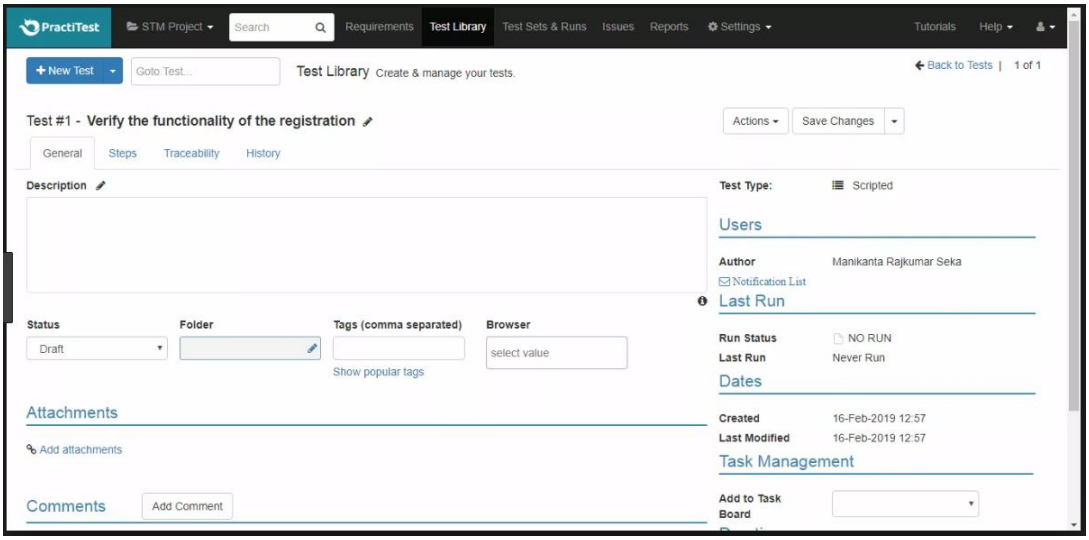

Рисунок 3.5 – Інтерфейс PraciTest

Цей інструмент допоможе організувати тест сьюти відповідно до ваших циклами і спринті. Тестові набори можна формувати за різними критеріями, таким як компоненти, версії або типи. Тул заточений на Agile тестування, регресійні тестування, тестування мікросервісов і DevOps.

А в тих підтримку працюють навчені QA співробітники, які можуть швидко зрозуміти вашу проблему.

Можливості:

– легке додавання тестів нових фіч в регресійне тестування;

– угруповання тестів на основі мікросервісов, які вони охоплюють, навіть крос-сервісні;

– різна відображення інформації для різних груп користувачів;

– дашборда в реальному часі показують стан тестів, прогонів на етапах розробки і при деплой на продакшн;

– інтеграція з jira, redmine, jenkins, gitlab і slack.

Таким чином, після аналізу інструментів, що використовуються для тестування веб додатків, потрібно сформувати критерії, за якими вони будуть оцінюватися експертами.

Для оцінки були виділені такі критерії:

– функціональність (чи виконує програма свої функції без збоїв?);

– user interface (UI);

– інтеграція зі сторонніми сервісами;

– автоматизація;

– up-time monitoring (постійний моніторинг) / підтримка.

Експерт - компетентне для вироблення оцінки особа, яка має спеціальний досвід в конкретній галузі і яка бере участь в дослідженні як джерело отримання інформації. Очевидно, в якості експертів необхідно використовувати тих людей, чиї судження допоможуть прийняттю адекватного рішення. При підборі експертів слід враховувати небезпеку особистої зацікавленості в тому чи іншому

рішенні, який може стати суттєвою перешкодою для отримання об'єктивного рішення.

Сутність методів експертних оцінок полягає в тому, що в основу прийнятого рішення, прогнозу, виведення закладається думка фахівця чи колективу фахівців, засноване на їх знаннях і практичному професійному досвіді. В першу чергу, експертної гідна називатися тільки та оцінка, яка дотримується правил об'єктивності і чесності.

На основі колективної роботи експертної групи, коли підсумкова оцінка являє собою колективну думку експертів, отримане методом консенсусу - прийняттям рішення на основі загальної згоди без проведення голосування.

На основі індивідуального думки членів експертної групи, незалежно один від одного формулюють оцінку або на підставі думки лідера думки.

Методи колективної роботи експертної групи передбачають отримання загальної думки в ході спільного обговорення вирішуваної проблеми.

Мозкова атака (мозковий штурм) - виступ експертів, на які накладено одне обмеження - не можна критикувати пропозиції інших.

Метод «635» - один з різновидів мозкової атаки. Цифри 6, 3, 5 позначають шість учасників, кожен з яких повинен записати три ідеї протягом п'яти хвилин. Лист ходить по колу. Таким чином, за півгодини кожен запише в свій актив 18 ідей, а все разом – 108.

Ділова гра - метод, заснований на моделюванні функціонування соціальної системи управління при виконанні операцій, спрямованих на досягнення поставленої мети.

Оцінка комісією - один з методів експертних оцінок, заснований на роботі спеціальних комісій. Групи експертів за "круглим столом" обговорюють ту чи іншу проблему з метою узгодження точок зору та вироблення єдиної думки.

"Суд" - метод, який реалізується за аналогією з веденням судового процесу, коли в ролі "підсудних" виступають обрані варіанти вирішення.

Методи отримання індивідуальної думки членів експертної групи засновані на попередньому отриманні інформації від експертів, опитуваних незалежно один від одного, з подальшою обробкою отриманих даних.

Метод "Дельфі" - розробка програми послідовних многотурових індивідуальних опитувань.

Метод інтерв'ю передбачає бесіду з експертом по схемі питання - відповідь.

Метод доповіді передбачає ретельну самостійну роботу експерта над аналізом, з наданням думки у вигляді аналітичної записки.

Способи вироблення як колективних, так і персональних експертних оцінок.

Оцінка на основі асоціацій - спосіб, заснований на вивченні схожого за властивостями об'єкта з іншим об'єктом.

Оцінка на основі по-парних (бінарних) порівнянь - спосіб, заснований на зіставленні експертом альтернативних варіантів.

Оцінка на основі векторів переваг - спосіб, заснований на експертному аналізі та перебору всього набору альтернативних варіантів і визначенні найкращого.

Оцінка на основі фокальних об'єктів - метод, заснований на перенесенні ознак випадково відібраних аналогів на досліджуваний об'єкт.

Оцінка на основі пошуку середньої точки - формулюються два альтернативних варіанти вирішення. Після цього експерт повинен підібрати третій альтернативний варіант, оцінка якого розташована між значень першої і другої альтернативи.

Застосування методу експертних оцінок (табл. 3.1). Експертні оцінки застосовуються на будь-якому етапі дослідження: у визначенні мети і завдання самого дослідження, в побудові і перевірці гіпотез, при виявленні проблемних ситуацій, в ході інтерпретації якіех-яких процесів, подій або фактів, для обґрунтування адекватності використовуваного інструментарію, в процесі

вироблення рекомендацій та т.д. Експертні методи оцінки застосовують в ситуаціях, коли вибір, обгрунтування і оцінка рішень не можуть бути виконані на основі точних розрахунків.

Таблиця 3.1 – Шкала оцінки

Значення	Важливість параметрів оцінки
1	однакова
3	незначна
5	значна
7	явна
9	абсолютна

Індивідуальні оцінки засновані на використанні думки окремих експертів, незалежних один від одного. Колективні оцінки засновані на використанні колективної думки експертів. Грубо кажучи, до першої групи відноситься оцінка статей на Хабре, голосування в опитуваннях і т.д., коли кожен експерт приймає рішення самостійно. Підбір (відсів) експертів здійснюється за допомогою карми. Саме перша група превалює в інтернеті 2 за рахунок можливості охоплення більшого числа експертів. Ранжування - це розташування об'єктів в порядку зростання або зменшення будь-якого властивого їм властивості.

Ранжування (рис. 3.6) дозволяє вибрати з досліджуваної сукупності факторів найсуттєвіший.

	1	2	…	j	…	m
1	a_{11}	a_{12}	…	a_{1j}	…	a_{1m}
2	a_{21}	a_{22}	…	a_{2j}	…	a_{2m}
…	…	…	…	…	…	…
i	a_{i1}	a_{i2}	…	a_{ij}	…	a_{im}
…	…	…	…	…	…	…
n	1_{ni}	a_{n2}	…	a_{nj}	…	a_{nm}

Рисунок – 3.6 Метод простого ранжування

Парне порівняння (табл. 3.2) - це встановлення переваги об'єктів при порівнянні всіх можливих пар. Тут не потрібно, як при ранжируванні, впорядковувати всі об'єкти, необхідно в кожній з пар виявити більш значимий об'єкт або встановити їх рівність. Безпосередня оцінка. Часто буває бажаним не тільки впорядкувати (ранжувати об'єкти аналізу), але і визначити, на скільки один фактор найбільш значущий, ніж інші. В цьому випадку діапазон зміни характеристик об'єкта розбивається на окремі інтервали, кожному з яких приписується певна оцінка (бал), наприклад, від 0 до 10. Саме тому метод безпосередньої оцінки іноді називають також бальним методом.

Таблиця 3.2 – Таблиця попарного порівняння критеріїв

	Функціонал	UI	Інтеграція	Автоматизація	Підтримка
Функціонал	1/7	1/5	1/5	1/9	1
UI	1/3	3	5	1	9
Інтеграція	1/7	5	1	1/5	5
Автоматизація	1/7	1	1/5	1/3	5
Підтримка	1	7	7	3	7
Σ	1,72	16,2	13,4	6,4	27

Нормалізація матриці – табл. 3.3.

Таблиця 3.3 – Нормалізація матриці

						Середнє значеня
Функціонал	0,47	0,05	0,43	0,58	0,25	0,27
UI	0,05	0,01	0,06	0,08	0,18	0,19
Інтеграція	0,03	0,07	0,3	0,08	0,18	0,09
Автоматизація	0,16	0,37	0,37	0,18	0,33	0,25
Підтримка	0,01	0,08	0,06	0,04	0,08	0,04

Попарне порівнення та нормалізація матриці (табл. 3.5-3.6).

Попарно порівняня UI та нормалізація матриці (табл. 3.6-3.7).

Таблиця 3.4 – Порівняння функціональності

	Z	PT	TC	qT	TL
Z	5	1	7	1	1
PT	1	1/5	1	5	1
TC	7	5	5	5	1
qT	9	1/7	7	9	7
TL	9	6	7	9	9
Σ	31	12,25	27	28	19

Таблиця 3.5 – Нормалізація матриці

Z	0,02	0,04	0,02	0,02	0,02	0,12
PT	0,09	0,22	0,06	0,08	0,06	0,25
TC	0,04	0,06	0,04	0,07	0,06	0,5
qT	0,4	0,33	0,13	0,19	0,5	0,22
TL	0,44	0,33	0,45	0,04	0,28	0,29

Таблиця 3.6 – Порівняння за критерієм UI

	Z	PT	TC	qT	TL
Z	1/5	1/7	1/3	1/5	1/5
PT	1	5	3	1/3	1/3
TC	7	9	1	1/3	1/7
qT	7	7	9	3	1/7
TL	5	5	7	5	3
Σ	20,5	26.25	20,33	8,58	3,8

Таблиця 3.7 – Нормалізація матриці

						Середнє значеня
Z	0,3	0,02	0,02	0,03	0,08	0,04
PT	0,13	0,52	0,18	0,04	0,08	0,12
TC	0,04	0,37	0,06	0,03	0,11	0,42
qT	0,21	0,02	0,03	0,18	0,08	0,4
TL	0,3	0,07	0,52	0,55	0,67	0,39

Попарно порівняня варіанти за інтеграцією зі сторонніми сервісами – таблиця 3.8. Нормалізація матриці попарного порівняння за критерієм наявності інтеграцій (табл. 3.9). Попарно порівняня варіанти за критерієм наявності автоматизації – таблиця 3.10. Нормалізація матриці попарного порівняння за критерієм наявності автоматизації (табл. 3.11).

Таблиця 3.8 – Порівняння за інтеграцією зі сторонніми сервісами

	Z	PT	TC	qT	TL
Z	1/7	1/5	7	1/7	1/9
PT	9	1	3	1/5	1/7
TC	1	1	3	1	1/5
qT	9	7	5	3	1
TL	7	5	5	1	3
Σ	22,14	14,5	21	4,81	4,38

Таблиця 3.9 – Нормалізація матриці

						Середнє значеня
Z	0,05	0,01	0,28	0,08	0,02	0,09
PT	0,23	0,07	0,12	0,11	0,03	0,11
TIT	0,01	0,02	0,04	0,08	0,03	0,17
qT	0,41	0,51	0,27	0,18	0,23	0,32
TL	0,32	0,37	0,27	0,55	0,68	0,44

Таблиця 3.10 – Порівняння за критерієм наявності автоматизації

	Z	PT	TC	qT	TL
Z	3	1	5	1/5	1/5
PT	1/5	3	1/3	3	1/9
TC	1/5	1	1	1/5	1/3
qT	1	9	3	1	1/3
TL	5	7	7	3	1.96
Σ	10	21	18/3	4,54	1

Таблиця 3.11 – Нормалізація матриці

						Середнє значення
Z	0,1	0,15	0,38	0,04	0,1	0,15
PT	0,03	0,05	0,16	0,03	0,05	0,06
TIT	0,01	0,02	0,05	0,04	0,17	0,12
qT	0,48	0,34	0,26	0,22	0,17	0,29
TL	0,48	0,44	0,16	0,7	0,51	0,46

Попарно порівняня програми за наявності підтримки (таблиця 3.12) та зроблено матрицю нормалізації за критерієм (таблиця 3.13)

Таблиця 3.12 – Порівняння за наявності підтримки

	Z	PT	TC	qT	TL
Z	1	3	5	1/3	7
PT	1/3	1	3	1/5	5
TC	1/5	1/3	1	1/7	3
qT	3	5	7	1	1/3
TL	1/7	1/5	1/3	3	1
Σ	4,67	9,53	16,33	4,67	16,33

Таблиця 3.13 – Нормалізація матриці

						Середнє значеня
Z	0,21	0,31	0,31	0,07	0,02	0,18
PT	0,07	0,1	0,18	0,04	0,31	0,14
TC	0,04	0,03	0,06	0,03	0,18	0,19
qT	0,64	0,52	0,43	0,21	0,43	0,45
TL	0,03	0,02	0,02	0,64	0,06	0,15

Кроком це розрахунок коефіцієнта для кожної програми.

Розраховано зведений коефіцієнт Zephyr:

$$П(Z) = 0,27×0,12+0,19×0,04+0,09×0,09+0,25×0,15+0,04×0,18= 0,16.$$

Розраховано зведений коефіцієнт PractiTest:

$$П(PT) = 0,27×0,25+0,19×0,12+0,09×0,11+0,25×0,06+0,04×0,14=0,25.$$

Розраховано зведений коефіцієнт для бібліотеки на основі Test Collab:

$$П(TC) = 0,27×0,5+0,19×0,42+0,009×0,17+0,25×0,12+0,03×0,19=0,26.$$

Розраховано зведений коефіцієнт для бібліотеки на основі qTest:

$$П(qT) = 0,27×0,22+0,19×0,4+0,09×0,32+0,25×0,29+0,04×0,45=0,25.$$

Розраховано зведений коефіцієнт для бібліотеки на основі TestLink:

$$П(TL) = 0,27×0,29+0,19×0,39+0,09×0,44+0,25×0,46+0,04×0,15=0,31$$

Після проведення експерименту та розрахунків, отримано результати, що наведені в табл. 3.14.

Таблиця 3.14 – Результати експерименту

Zyphyr	0,16	qTest	0,25
PractiTest	0,25	TestLink	0,31
Test Collab	0,26		

3.3 Розрахунок узгодженості експертної оцінки

Визначення узгодженості оцінок експертів необхідно для підтвердження правильності гіпотези про те, що експерти є досить точними вимірювачами, і виявлення можливих угруповань в експертній групі. Оцінка узгодженості думок експертів характеризує ступінь близькості індивідуальних думок.

Розраховано узгодженість думок експертів щодо важливості обраних критеріїв за допомогою значення коефіцієнту конкордації Кенделла:

$$W = \frac{(n+m)S}{n^2(m^3-m)}. \quad (3.1)$$

де S – сума квадратів відхилень всіх оцінок рангів кожного критерія від середнього значення; n – кількість експертів; m – кількість критеріїв.

Коефіцієнт конкордації змінюється у діапазоні $0<W<1$, звідси 0 – повна неузгодженість, а 1 – повна узгодженість (табл. 3.15).

Визначення величини коефіцієнта конкордації:

$$W=10\times340,2/25\times(125-5) = 0,89.$$

Таблиця 3.15 – Розрахунок узгодженності думок

Експерти (5)	Критерії (5)					Σ				
	1	2	3	4	5					
Сума рангів	27	6,4	13,4	16,2	1,72	64,72				
Відхилення від середньої суми рангів					14,06	6,54	0,46	3,26	-11,2	13,12
Квардрат відхилення					197,68	42,77	0,21	10,63	125,9	340,2

Рівень узгодженості думок експертів впливає на коректність результатів виконуваного дослідження. Як показано у розрахунках, що наведено вище, у даному випадку узгодженість експертної оцінки висока.

3.4 Рекомендації

Данні інструменти були вибрані за більшу кількість користовачив, так як аналогічних програм дуже багато, і всі вони відрізняються.

В першу чергу можно виділити ціну, так як ціна залежить від кількості користовачів, місяц за одного юзера. І деякі інструменти коштують багато, но як показує практика, у великих організаціях викуповують ліцензію або купують декілька аккаунтів і під час тестування на цей аккаунт можуть зайти декілька користовачив або почерзі, що не є зручно під час роботи. Таким чином більш приваблива пропозиція є та яка коштуе менше, к чому приводить до меньшого функціоналу. Деякі платформи мають безкоштовний період. Тому ми і робили крітерії до них

Які основні крітерії користовачі хотять бачити.

- зручна установка і підтримка;
- створення та управління проектами;

– створення користувачів і ролей користувачів;
– спокійна інтеграція з автоматичними тестами;
– написання тест-плану;
– написання тест-кейса;
– виведення тест-кейса;
– зрозуміла система звітності;
– вбудована система багтрекінгу;
– спосіб інтеграції з іншими інструментами.

Не усі платформи мають свою багтрекінгову системи, томи користовачі и чекают от інструмента інтеграцію с іншими багтрекінговими платформами. Ще достатньо важлива частина це формування звіту, на це може уйти багато часу, но встроена функція написання звіту може с економити багато часу.

Експорт та імпорт це теж є не відьемлива частина в частності і для написання звіту, різні інструменти підтримують різні формати, і якщо інструмент підтримуе усі формати, то становиться зручніше іх видкривати на інших платформах. Інтеграція с іншими платформами це дуже важлива частина, як і написаня тестів.

3.5 Результат дослідження

Після експерементальної частини наукової работи все одно ми не можемо сказати яка платформа буде краща. Так как як користувач вибирае сам що цому більш подобаеться і якої зручно користуватися.

В даної наукової роботі були виявлені основні крітерії якимі повина мати сачасна платформа тестування, був проведен аналіз стану та порівняня методом експертної оцінки.

Експертне оцінювання - процес отримання оцінки чого-небудь, на основі думки експертів, з метою подальшого прийняття рішення або вибору.

Індивідуальні оцінки засновані на використанні думки окремих експертів, незалежних один від одного.

Колективні оцінки засновані на використанні колективної думки експертів.

Спільне думку володіє більшою точністю, ніж індивідуальна думка кожного з фахівців. Даний метод застосовують для отримання кількісних оцінок якісних характеристик і властивостей. Наприклад, оцінка декількох технічних проектів по їх ступеня відповідності заданим критерієм, під час змагання оцінка суддями виступу фігуриста.

Відомі такі методи експертних оцінок:

Метод асоціацій. Заснований на вивченні схожого за властивостями об'єкта з іншим об'єктом.

Метод парних (бінарних) порівнянь. Заснований на зіставленні експертом альтернативних варіантів, з яких треба вибрати найкращі.

Метод векторів переваг. Експерт аналізує весь набір альтернативних варіантів і вибирає найкращі.

Метод фокальних об'єктів. Заснований на перенесення ознак випадково відібраних аналогів на досліджуваний об'єкт.

Індивідуальний експертне опитування. Опитування у формі інтерв'ю або у вигляді аналізу експертних оцінок. Чи означає бесіду замовника з експертом, в ході якої замовник ставить перед експертом питання, відповіді на які значимі для досягнення програмних цілей. Аналіз експертних оцінок передбачає індивідуальне заповнення експертом розробленого замовником формуляра, за результатами якого проводиться всебічний аналіз проблемної ситуації і виявляються можливі шляхи її вирішення. Свої міркування експерт виносить у вигляді окремого документа.

Метод середньої точки. Формулюються два альтернативних варіанти вирішення, один з яких менш привабливий. Після цього експерт повинен

підібрати третій альтернативний варіант, оцінка якого розташована між значень першої і другої альтернативи.

Умовами невизначеності вважається ситуація, коли результати прийнятих рішень невідомі. Невизначеність підрозділяється на стохастическую (мається інформація про розподіл ймовірності на безлічі результатів), поведінкову (мається інформація про вплив на результати поведінки учасників), природну (мається інформація тільки про можливі результати і відсутня про зв'язок між рішеннями і результатами) і апріорну (немає інформації і про можливі результати). Завдання обгрунтування рішень в умовах невизначеності всіх типів, крім апріорної, зводиться до звуження початкової множини альтернатив на основі інформації, якою володіє ЛПР. Якість рекомендацій для прийняття рішень в умовах стохастичної невизначеності підвищується при обліку таких характеристик особистості ЛПР, як ставлення до своїх виграшів і програшів, схильність до ризику. Обгрунтування рішень в умовах апріорної невизначеності можливо побудовою алгоритмів адаптивного управління

4 ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ПРОЕКТУ

4.1 Характеристика наукового дослідження

У даному розділі роботи розраховуються витрати на проведення дослідження інструментів тестування веб додатків. У ході обґрунтування необхідно провести розрахунок: трудовитрат та заробітної плати працівникам; одноразових витрат; прибутку; оцінки роботи; економічної ефективності НДР.

Метою атестаційної роботи є дослідження інструментів тестування веб додатків. Оптимальна платформа яка підтримує інтеграцію с багтрекінговими системами.

Об’єктом дослідження даної роботи є процес тестування веб додатків. Предмет дослідження – інструменти для тестування веб додатків.

У ході дослідження проводився аналіз предметної області, визначення основних критеріїв, які повинні бути присутні, огляд існуючих інструментів для тестування веб додатків, порівняльний аналіз даних бібліотек, а також було розроблено набір рекомендацій для створення бібліотек для збереження тест-кейсів.

4.2 Етапи виконання НДР, їх трудомісткість та заробітна плата

У процесі виконання НДР було вирішено такі задачі:

– проведено аналіз стану проблеми тестування веб додатків;

– проведено аналітичний огляд літератури за темою атестаційної роботи;

– проведено порівняльний аналіз існуючих інструментів;

– сформовано критерії для платформ тестування;

– розроблено рекомендації щодо інстументів тестування.

Умовно науково-дослідну роботу можна розділити на три етапи: підготовчий, основний і заключний.

На стадії виконання підготовчого етапу було проведено аналіз процесу тестування мобільних додатків, аналіз процесу створення основних критеріїв до інструментів та проведено аналітичний огляд літератури за темою атестаційної роботи. Пошук інформації проходив у мережі Internet, а також було розглянуто літературні джерела.

Роботами, що було проведені на основній стадії дослідження, були:

– порівняльний аналіз існуючих інструментів тестування веб додатків;

– формування критеріїв інструментів тестування веб додатків;

– розробка рекомендацій щодо інструментів тестування веб додатків.

На заключному етапі науково-дослідної роботи було проведено аналіз результатів виконання НДР, визначення методики, складання звіту по НДР, а також захист звіту. При плануванні науково-дослідної роботи потрібно провести розрахунок трудомісткості робіт, що є найбільш відповідальною частиною етапу. Трудові витрати часто становлять основну частину вартості науково-дослідних робіт і безпосередньо впливають на часові строки розробки.

У даній роботі біло задіяно два фахівця: керівник роботи та тестувальник. Середньомісячна заробітна плата кваліфікованого тестувальника становить 11000 грн/місяць, керівник роботи отримує 17000 грн/місяць.

Проведемо розрахунок трудовитрат і заробітної плати виконавця робіт.

Середньоденна заробітна плата виконавця робіт ($З_{ср.дн.}$) розраховується:

$$З_{ср.дн.} = \frac{З_{ср.міс.}}{n}, \qquad (4.1)$$

де $З_{ср.міс.}$ – середньомісячна зарплата виконавця роботи;

n – число робочих днів у місяці, (n=22).

Середньоденна заробітна плата тестувальника та кеівника роботи складає відповідно:

$$З_{ср.дн} = \frac{11000}{22} = 500(грн), \quad З_{ср.дн} = \frac{17000}{22} = 773\ (грн).$$

Етапи виконання НДР, перелік і зміст робіт, трудомісткість їх виконання, заробітна плата виконавців робіт представлені в таблиці 4.1.

Таблиця 4.1 – Етапи виконання НДР

Перелік робіт	Кількість виконавців	Посада виконавця	Трудомісткість робіт, люд. днів	Середньоденна заробітна плата, грн	Сума заробітної плати, грн
1. Підготовчий етап					
1.1. Розробка та затвердження ТЗ	1	керівник роботи	1	773	773
1.2 Підготовка довідкових матеріалів та даних для виконання НДР	1	керівник роботи	1	773	773
2. Основний етап					
2.1 Постановка задачі	1	керівник роботи	1	773	773
2.2 Аналіз аналогів	1	тестуваль-ник	2	500	1000
2.3 Аналіз програми	1	тестуваль-ник	1	500	500
2.4 Проведення порівняння аналогів програм	1	тестуваль-ник	2	500	1000
2.5 Формування основних критеріїв	1	тестуваль-ник	1	500	500
2.6 Розробка рекомендацій щодо програм-бібліотек для збереження тест-кейсів	1	тестуваль-ник	1	500	500
3. Заключний етап					
3.1 Аналіз результатів проведення роботи	2	керівник роботи, тестуваль-ник	1	1273	1273
3.2. Формування звіту, висновків та пропозицій за темою дослідження	1	керівник роботи	2	773	1546
Всього			15		8138

4.3 Розрахунок одноразових витрат на НДР

Калькуляція собівартості розраховується відповідно до існуючих нормативних актів України. До складу калькуляції входять такі статті витрат:

– матеріальні витрати;

– витрати на оплату праці;

– єдиний соціальний внесок;

– амортизація основних засобів (вартість машинного часу);

– витрати на електроенергію;

– інші витрати.

До інших витрат відносяться адміністративні витрати (водопостачання, водовідведення, опалення, освітлення) та вартість послуг зв'язку.

Матеріальні витрати визначаються витратами на матеріали та визначені їх потребою для виконання робіт, і цінами, що діють на момент складання калькуляції. Для проведення НДР потрібно: 1 олівець, 2 шт. ручки та 2 шт. блокноти.

Матеріальні витрати розраховуються за формулою:

$$M = \sum_{j=1}^{n} Q_j \cdot Ц_j , \qquad (4.2)$$

де М – сумарні витрати на матеріали, в тому числі малоцінні предмети, що швидко зношуються (носії, папір, канцелярське приладдя тощо), або на літературу, яка необхідна для проведення роботи, тощо;

Q_j – кількість використаних одиниць j-го виду матеріалів, j =(1÷n);

$Ц_j$ – ціна одиниці *j*-го виду матеріалів.

Розрахунок матеріальних витрат представлено в табл. 4.2.

Таблиця 4.2 – Розрахунок матеріальних витрат

Найменування	Од. вим.	Кількість,шт	Ціна, грн	Сума, грн.
Олівець	шт.	1	2	2
Ручка	шт.	2	5	10
Блокнот	шт.	2	10	20
Всього				32

Витрати на оплату праці розраховуються виходячи з необхідного для виконання робіт складу й кількості працівників, а також із середньомісячної заробітної плати. Згідно проведеним раніше розрахунком, витрати на оплату праці дорівнюють 8138 грн.

Єдиний внесок на загальнодержавне соціальне страхування (ЄСВ) – консолідований страховий внесок, збір якого здійснюється в систему загальнообов'язкового державного соціального страхування в обов'язковому порядку і на регулярній основі з метою забезпечення захисту у випадках, передбачених законодавством, прав застрахованих осіб і членів їх сімей на отримання страхових виплат (послуг) за діючими видами загальнообов'язкового державного соціального страхування.

Для об'єкта дослідження ставка єдиного соціального внеску дорівнює 22% від витрат на оплату праці, тобто розмір ЄСВ дорівнює 1790,36 грн.

При виконанні НДР застосовувалось наступне обладнання: 2 комп'ютера вартістю 20000 грн. Вищенаведене устаткування є власністю організації, тому доцільно розрахувати суму амортизаційних відрахувань на період виконання НДР:

$$АВ = \sum_{k=1}^{L} \frac{ВО_k}{ТЕ_k} \cdot Т, \quad (4.3)$$

$$АВ = \frac{20000 \times 8}{560} + \frac{20000 \times 6}{560} = 428{,}57 \text{ (грн).}$$

де АВ – сума амортизаційних відрахувань, нарахованих під час проведення НДР;

$ВО_k$ – вартість основних засобів *k*-го виду;

$ТЕ_k$ – термін експлуатації основних засобів *k*-го виду, днів;

Т – термін науково-дослідницької роботи, днів;

L – кількість видів обладнання.

Витрати на використану обладнанням електроенергію:

$$З_е = М \cdot t \cdot Т_{кВт}, \quad (4.4)$$

$З_з = 0,60 \times 48 \times 1,68 + 0,60 \times 56 \times 1,68 = 48,38 + 56,44 = 104,82$(грн),

де М – потужність устаткування, тобто кількість енергії, споживаної за одиницю часу (кВт/година);

t – кількість годин використання устаткування за період проведення науково-дослідницької роботи;

$Т_{кВт}$ – тариф, тобто вартість використання 1 кВт електроенергії.

Споживна потужність комп'ютера складає 0,65 кВт за годину. Тариф споживачів за першим класом напруги, тобто 35 кВт та більше), складає 1,68 грн/кВт·годин (без ПДВ).

До інших статей витрат відносяться такі:

– адміністративні витрати: (водопостачання, водовідведення, освітлення, опалення), які прийнято у розмірі 20% від витрат на оплату праці;

– вартість оплати послуг зв'язку.

Вартість оплати послуг зв'язку становитиме інтернет – із розрахунку 300 грн на місяць (тариф на доступ до мережі інтернет у нежитлових приміщеннях); всього 150 грн за 15 днів виконання НДР.

PractiTest – 1400 грн.

За час виконання НДР витрати на відрядження, аутсорсинг, інформаційні послуги не мали місця. Результати розрахунку кошторису витрат, тобто одноразових витрат, на виконання НДР наведені в таблиці 4.3.

Таблиця 4.3 − Кошторис витрат на розробку та ціна НДР

№ з/п	Показник	Сума, грн
1	Заробітна плата	8138
2	Єдиний соціальний внесок (22,0 % від п.1)	1790,36
3	Матеріальні витрати	32
4	Амортизація основних засобів	428,57
і5	Витрати на електроенергію	104,82
6	Інші витрати, у тому числі:	
6.1	вартість ліцензії	1400
6.2	вартість послуг інтернету	150
6.3	адміністративні витрати (20% від п.1)	1627,6
7	Всього	13671,35

Таким чином, кошторис витрат на виконання даної НДР відбиває сумарні витрати за статтями п.1÷п.6 та складає 13671,35грн.

4.4 Оцінка результатів науково-дослідної роботи

Результат – це завершальний наслідок послідовності дій, виражений якісно або кількісно. В загальному випадку оцінка результатів НДР – це визначення ефективності отриманих рішень порівняно з сучасним науково-технічним рівнем. Відповідно до теми даної роботи можна зробити висновок про те, що у якості результату впровадження НДР є зменшення часу, що має досить велике значення для розробників, та як наслідок, користувачів:

$$\Delta P_j = \left| Хб_j - Хн_j \right|, \tag{4.5}$$

$$\Delta P_1 = |25{,}35 - 12{,}25| = 13{,}1.$$

де ΔP_j – покращення j-ої характеристики досліджуваного процесу за рахунок впровадження результатів НДР ($j=1,m$);

m – кількість досліджуваних характеристик;

$Хб_j$ – базове значення j-ої характеристики, тобто до впровадження результатів НДР;

$Хн_j$ – нове значення j-ої характеристики після впровадження пропонованих рішень.

У якості досліджуваної характеристики виступає оцінка експертів обраної платформи на етапі тестування веб додатків (PractiTest) та оцінка після аналізу та формування рекомендацій щодо бібліотек, що в подальшому будуть створені, спираючись на ці рекомендації. Оцінка експертів визначається сумарною бальною оцінкою бібліотек з урахуванням ваги кожного критерію, яка дорівнює до впровадження 12,25 та після формування рекомендацій – 25,35.

Далі проведено оцінку економічної ефективності отриманого результату виконаної науково-дослідної роботи.

4.5 Визначення економічної ефективності результатів НДР

Для визначення економічної ефективності результатів НДР необхідно порівняти витрати на розробку НДР з отриманими результатами.

Основним показником економічної ефективності науково-дослідної роботи є коефіцієнт «ефект-витрати», який розраховується за такою формулою:

$$K_{ев} = \frac{\Delta P_j}{B_p}, \tag{4.6}$$

$$K_{ев} = \frac{13{,}1}{13671{,}35} = 0{,}0009\ (\%).$$

де B_p – витрати (кошторисна вартість) на виконання НДР, грн.;

$K_{ев}$ – коефіцієнт «ефект-витрати», який відбиває, наскільки кожна гривня витрат НДР змінює j-ту характеристику досліджуваного процесу.

Підставивши раніше визначені значення до (4.6), розрахуємо чисельне значення коефіцієнту «ефект-витрати»:

У результаті проведених досліджень, можна зробити висновок про те, що кожна гривня витрат на розробку НДР є більш оптимальна платформа тестування веб додатків чим у інших платформ 0,0009 %. Дана науково-дослідна робота має позитивний показник економічної ефективності. Роботу у цілому можна враховувати ефективною або такою, що має науковий та технічний рівень.

ВИСНОВКИ

В результаті проведення наукової роботи було проведено дослідження інструментів тестуванн веб додатків та находження оптимальної платформи тестування.

Для досягнення поставленої мети було проведено аналіз стану проблеми тестування веб додатків, аналітичний огляд літератури за темою атестаційної роботи, а ще проведено аналіз стану проблеми процесу створення та збереження тест-кейсів.

Також було зроблено порівняльний аналіз існуючих платформ тестувань. Це було досягнуто, використовуючи метод ієрархій, а також метод експертних оцінок та попарного порівняння обраних програм.

Завдяки цьому методу була виявлена найбільш актуальна програма, на базі якої буде будуватися висновки. За результатами було сформовано критерії, які повинні бути присутні у інструментах та платформах тестування.

Проведено економічне обґрунтування науково дослідної роботи та розраховано економічну ефективність даного дослідження.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Вибір системи управління тестуванняя. URL: https://habr.com/ru/post/463333/#qtest-manager (дата зверненя 25.11.2020).

2. Особоливості тестування веб додатків. URL: https://qualitylab.ru/blog/key-principles-of-web-testing/ (дата зверненя 30.11.2020).

3. Стадии цикла разработки ПО. URL: https://qalight.com.ua/bazaznaniy/stadii-tsikla-razrabotki-po/ (дата звернення 20.11.2020).

4. Інструменти упарвління процессом тестування. URL: https://testmatick.com/ru/top-17-instrumentov-testirovaniya-v-2019-godu/ (дата звернення 20.11.2020).

5. Автоматізація тестування. URL: https://softwaretesting.ru/library/testing/testing-automation (дата звернення 20.11.2020).

6. Забеспечення якості. URL: http://www.protesting.ru/qa/ (дата звернення 20.11.2020).

7. 10 лучших инструментов для автоматизации тестирования ПО. URL: https://habr.com/ru/post/481294/ (дата звернення 20.11.2020).

8. Тестування програмного забезпечення. URL: https://uk.wikipedia.org/wikiwiki/Тестування_програмного_забезпечення#Методи_тестування (дата звернення 20.11.2020).

9. Найкращі системи управлінням тестуванням. URL: wiki/Тестування_програмного_забезпечення#Методи_тестування (дата звернення 20.11.2020).

10. Якість програмного забеспечення (ISO/IEC 25010). URL: https://qalight.com.ua/baza-znaniy/kachestvo-programmnogo-obespecheniya/ (дата звернення 20.11.2020).

11. Універсальна схема для тестування веб додатків. URL: https://dou.ua/lenta/articles/scheme-for-qa/ (дата звернення 20.11.2020).

12. Забезпечення якості. URL: https://ru.wikipedia.org/wiki/Обеспечение_качества (дата звернення 20.11.2020).

13. ISO 9000. URL: https://ru.wikipedia.org/wiki/ISO_9000 (дата звернення 20.11.2020).

14. Кулішова Н.Є. Методичні вказівки з виконання магістерської атестаційної роботи для напряму підготовки 6.051501 «Видавничо-поліграфічна справа» (освітньо-кваліфікаційний рівень – магістр). Харків: ХНУРЕ, 2010. 44 с.

15. Методичні рекомендації до виконання економічної частини дипломних проектів, робіт для студентів денної та заочної форми навчання усіх спеціальностей / Л.В. Соколова та ін. Харків: ХНУРЕ, 2015. 49 с.