Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерної інженерії та системного програмування

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Інформаційна система для автоматизації тестування Web-застосувань
Назва теми

КвРКІ.170135.17.01.03 ПЗ
Шифр

Галузь знань 12 «Інформаційні технології»
Шифр, назва

Спеціальність 123 «Комп'ютерна інженерія»
Шифр, назва

Освітня програма «Комп'ютерна інженерія»
Назва

Виконав: студент IV курсу, група КІ–17–1 _____ О. В. Гладкий
Підпис Ініціали, прізвище

Керівник _____ О. В. Бармак
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної
Інженерії та системного
Програмування _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

« 18 » червня 2021 р.

Хмельницький 2021

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Програмування та комп'ютерних і телекомунікаційних систем

Кафедра Комп'ютерної інженерії та системного програмування

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітня програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

"11" 01 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Гладкому Олександру В'ячеславовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інформаційна система для автоматизації тестування Web-застосувань

Керівник проекту (роботи) Бармак О.В., д.т.н., проф.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2021 р. № 11

2. Строк подання студентом проекту (роботи) на кафедру 07.06.2021 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Дослідження предметної області та постановка задачі

Проектування програмного рішення та набір інструментів та технологій для розробки

Розробка проекту та його тестування

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

UML діаграма класів

Блок-схема роботи усіх тестових сценаріїв

Блок-схема роботи класу TestRegistration

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання при
Нормоконтроль	Лисенко С.М., професор кафедри КІСП		
Антиплагіат	Нічепорук А.О., доцент кафедри КІСП		

7. Дата видачі завдання «_11_»_____01_____2021 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Пр
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	11.01.2021	вик
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2021	вик
3	Робота над розділом 1 – аналіз проблем процесу тестування при розробці Web-застосування	01.03.2021	вик
4	Робота над розділом 2 – інструменти та способи автоматизованого тестування Web-застосувань	01.04.2021	вик
5	Робота над розділом 3 – програмно апаратна реалізація та тестування програмно технічного засобу	30.04.2021	вик
6	Оформлення пояснювальної записки згідно вимог	31.05.2021	вик
7	Попередній захист ВКР	02.06.2021	вик
8	Захист ВКР на засіданні ЕК	Червень 2021 року	

Студент _____ О. В. Гладкий
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О. В. Бармак
Підпис Ініціали, прізвище

№ рядка

форма

1 2 3 4

Зм Розр Пере Н. ко За

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 170135.17.01.03 ПЗ	Пояснювальна записка	65		
			Графічні матеріали			
2		КвРКІ 170135.17.01.03 Е8	UML діаграма класів	1		
3		КвРКІ 170135.17.01.03 Е8	Блок-схема виконання	1		
			класу TestRegistration			
4		КвРКІ 170135.17.01.03 Е2	Блок-схема виконання	1		
			усіх класів			

КвРКІ 170135.17.01.03 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Гладкий		
Перевір.		Бармак		
Н. контр.		Лисенко		
Затв.		Говорущенко		18.06

Відомість проекту

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ-17-1

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Інформаційна система для автоматизації тестування Web-застосувань».

Автор роботи: Гладкий Олександр В'ячеславович.

Керівник роботи: Бармак Олександр Володимирович.

Пояснювальна записка: 65 с., 24 рис., 1 табл., 4 дод., 45 джерел.

Графічна частина: 7 презентаційних слайдів.

ТЕСТУВАННЯ, ВЕБДОДАТКИ, БЕЗПЕРЕРВНА ІНТЕГРАЦІЯ, JAVA, SELENIUM, ПІРАМІДА ТЕСТУВАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

Метою роботи є побудова системи автоматизованого тестування програмного забезпечення або веб-додатку за рахунок поєднання у собі кількох способів тестування та налаштування безперервної інтеграції для проекту.

У цій роботі розроблена інформаційна система для автоматизації тестування Web-застосувань. Розроблена система керування реалізована на основі застосування мови програмування Java та інструмента Selenium. Розроблена система керування відтворення тестових сценаріїв, реалізований Robot Framework для розробки приймальних авто-тестів, дозволяє здійснювати автоматизоване тестування , а також система реалізує контроль якості проекту на основі логування.

Підпис студента

17.06.2021
Дата

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

АТ - автоматизоване тестування.

БД - база даних.

ОС - операційна система.

ПЗ - програмне забезпечення.

ПО - це сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм.

ОЗУ - це швидкодійна комп'ютерна пам'ять, призначена для запису, зберігання та читання інформації у процесі її обробки.

ЖЦАТ - життєвий цикл автоматизації тестування.

API - прикладний програмний інтерфейс.

GUI - графічний інтерфейс користувача.

Java - об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java.

JSON - це текстовий формат обміну даними між комп'ютерами.

QA engineer - фахівець, який бере участь в тестуванні компонента або системи.

Selenium (SE) - інструмент для автоматизації роботи в web-браузері.

SEO - пошукова оптимізація сайту

SDLC - життєвий цикл програмного забезпечення

Test case - це артефакт, що описує сукупність кроків, конкретних умов та параметрів, необхідних для перевірки реалізації функції

ВСТУП

Сьогодення відзначається значним розвитком інформаційних та комп'ютерних технологій. Вже важко уявити людину яка не буде мати комп'ютера, ноутбука чи простого смартфона. Крім цього варто зазначити їхню здатність обчислювати велику кількість інформації. До прикладу смартфон вже давно почав перемагати по розмірах та обчислювальних можливостях перший персональний комп'ютер Altair 8800. Для сучасної ж людини смартфон це вже як повсякденність і вже важко уявити людину без нього.

Паралельно до розвитку обчислювальних можливостей різної техніки значних змін зазнали і програмне забезпечення яке удосконалило взаємодію між комп'ютером і людиною та представлене у вигляді інтерфейса. Разом з тим їхня складність в плані розробки, а також підтримки підвищується в рази. Але незважаючи на цю збиткову тенденцію, розробка нових програмних продуктів продовжується і все більше компаній цим займаються у різних сферах, до прикладу:

1) підприємництво;
2) транспорт та інфраструктури;
3) телекомунікація, медіа, кіно і IT-технології;
4) освіта;
5) бізнес;
6) медицина.

Не дивлячись на те, що метою нових програмних продуктів є спрощення роботи та максимальна зручність використання для користувачів. Майже ніхто не звертає увагу, на те як важко розробити сам концепт проекту, налагодити взаємодію між розробниками та тестувальниками, а також вибрати метод, який буде допомагати командам вести спільну роботу. На даний час виділяють декілька методів управління проектами:

1) Six Sigma;
2) Lean;
3) Scrum;

4) Kanban;

5) Agile;

6) класичний проектний менеджмент.

Взаємодія між командами в останній час став важливою складовою успішного програмного продукту як ніколи раніше. І тестування розроблюваного проекту займає 2 позицію по важливості після його розробки. Це пояснити дуже легко, команда розробників може бути кількість в десятки, сотні чи навіть тисячі. І враховуючи той факт, що всі вони будуть працювати паралельно друг до друга над одним проектом, виникнення помилок в програмі або по іншому багів, суттєво збільшується. А важливість віднайдення цієї помилки до того як це зробить сам користувач взагалі безцінно. Тому й все більшість IT-компаній затрачають великі ресурси та бюджети на етап тестування ПЗ. А сама посада QA інженер або просто тестувальник набирає популярність в сфері IT.

Градація тестувальників відбувається по кількості досвіду роботи. В більшості компаній виділяють такі ступені як:

1) trainee QA;

2) junior QA;

3) middle QA engineer;

4) senior QA engineer.

Крім цього є і поділ по ролях, серед них 4 основні:

1) test analyst;

2) test designer;

3) test executor;

4) test manager.

Саме тестування ПЗ також поділяють на декілька видів. Існують кілька факторів, за якими відбувається класифікація. До прикладу:

1) за об’єктом тестування;

2) в залежності від цілей тестування;

3) за знанням ПЗ чи системи;

4) за ступенем ізоляції частин;

5) за періодом тестування;

6) за показником позитивності сценарію;

7) за рівнем підготовки до тестування;

8) за рівнем автоматизації процесу тестування.

Всього є 3 рівні автоматизації процесу тестування. А саме:

1) ручне тестування;

2) автоматизоване тестування;

3) напів-автоматизоване тестування.

Серед них найпопулярніший рівень це ручне тестування. Що пояснюється відносно низьким порогом входу. Але все більше ІТ-компанії бажають максимально автоматизувати тестування при розробці ПЗ, з метою зниження витрат ресурсів та кількості часу на цей процес. Також ретельні підготовлені сценарії тестування які будуть автоматизовані позбавляють можливості появи помилок в процесі тестування. Проте повністю відкидати ручне тестування в ніякому разі не можна. Тому що були і будуть такі моменти при тестуванні коли раціональніше перевірити крихітну нову функцію проекту вручну і не затрачувати час який би пішов на автоматизацію цього тесту. Враховуючи це в найближчий час планується збільшення використання саме методу напів-автоматизованого тестування. Суть якого полягає у поєднанні двох частин - ручного тестування і використання автоматизованого скрипту. Що дає змогу використовувати переваги обох методів тестового виконання.

1 АНАЛІЗ ПРОБЛЕМ ПРОЦЕСУ ТЕСТУВАННЯ ПРИ РОЗРОБЦІ WEB-ЗАСТОСУВАННЯ

1.1 Аналіз ролі тестування у розробці веб-застосувань та якості програмного забезпечення

Наразі використання веб-застосувань зростає з кожним днем. Спостерігається значне збільшення стандартів та потреб в Інтернеті через запровадження нових технологій веб-додатків. Більше того, різні люди часто змішують веб-додаток із веб-сайтом. Але веб-застосування або ж веб-додаток є більш надійними, масштабованими та надійними порівняно з веб-сайтами. Все більше розробників програмного забезпечення та тестувальників програмного забезпечення, мають справу саме з веб-вебдодатками.

Під веб-додатком розуміють прикладну програму, що зберігається на сервері, і до неї можна отримати доступ з будь-якого браузера. Фактично це веб-сайт, який виконує певну кількість функцій для користувачів. Тому схожість сайту і веб-додатку важко заперечувати. Найчастіше веб-додаток створюється архітектурно за системою клієнт-сервер. В середовищі клієнт-сервер присутні велика кількість комп'ютерів які можуть здійснювати обмін інформацією, зазвичай це збереження інформації в базі даних. У свою чергу "клієнт" використовується для введення інформації, а "сервер" як сховок для інформації.

Інтерфейс зазвичай створюється з використанням таких мов, як HTML, CSS, Javascript, які підтримуються основними браузерами. Хоча back end міг би використовувати будь-яку мову програмування, такий як LAMP, MEAN тощо. На відміну від мобільних програм, для розробки веб-додатків не існує конкретного SDK.

В загальному веб-додатки поділяються на дві основні групи: статичні та динамічні. Крім цього динамічні веб-додатки поділяються на інші підтипи.

Основні типи веб-додатків:

1) статичні веб-додатки;
2) динамічні веб-додатки;

3) односторінкові додатки;

4) багатосторінкові додатки;

5) портальні веб-додатки;

6) анімовані веб-додатки;

7) насичені Інтернет-додатки;

8) веб-додатки на базі JavaScript;

9) прогресивні веб-додатки;

10) веб-додатки електронної комерції.

Розподіливши веб-додатки по різних типах на основі вмісту, що вони відображають, розглянемо їх детальніше та різницю між ними.

Статичні веб-додатки:

Ці додатки вважаються сторінками, вони зазвичай сформовані сервером з дуже малою інтерактивністю або взагалі її відсутністю. Статичні веб-додатки складаються з обмеженого вмісту і не мають гнучкості. Також як правило, у них немає персоналізації тому після її повного завантаження на сторінці відбуваються зміни.

Тобто наяву статичні веб-додатки відображаються клієнтам такими ж, якими їх було збережено на сервері(рисунок 1.1). Зміст вмісту на сервері додатків не вноситься до того, як сторінка буде відправлена у веб-браузер.

У більшості випадків мови програмування, що використовуються для створення статичних веб-додатків, включають HTML, CSS, JavaScript тощо. Хоча до статичних веб-додатків можна включити анімовані об'єкти, GIF-файли, відео тощо, оновлювати їх дуже важко.

Також варто виділити, що статичні веб-додатки неможливі для мобільних середовищ. Проте вони підходять для додатку, який хоче передавати точну інформацію, і де не потрібна взаємодія.

Серед прикладів статичних веб-додатків: професійні портфоліо, цифрові резюме, цільова сторінка для маркетингу тощо. Всі вони доволі легкі в проектуванні і не затрачають багато ресурсів при розробці.

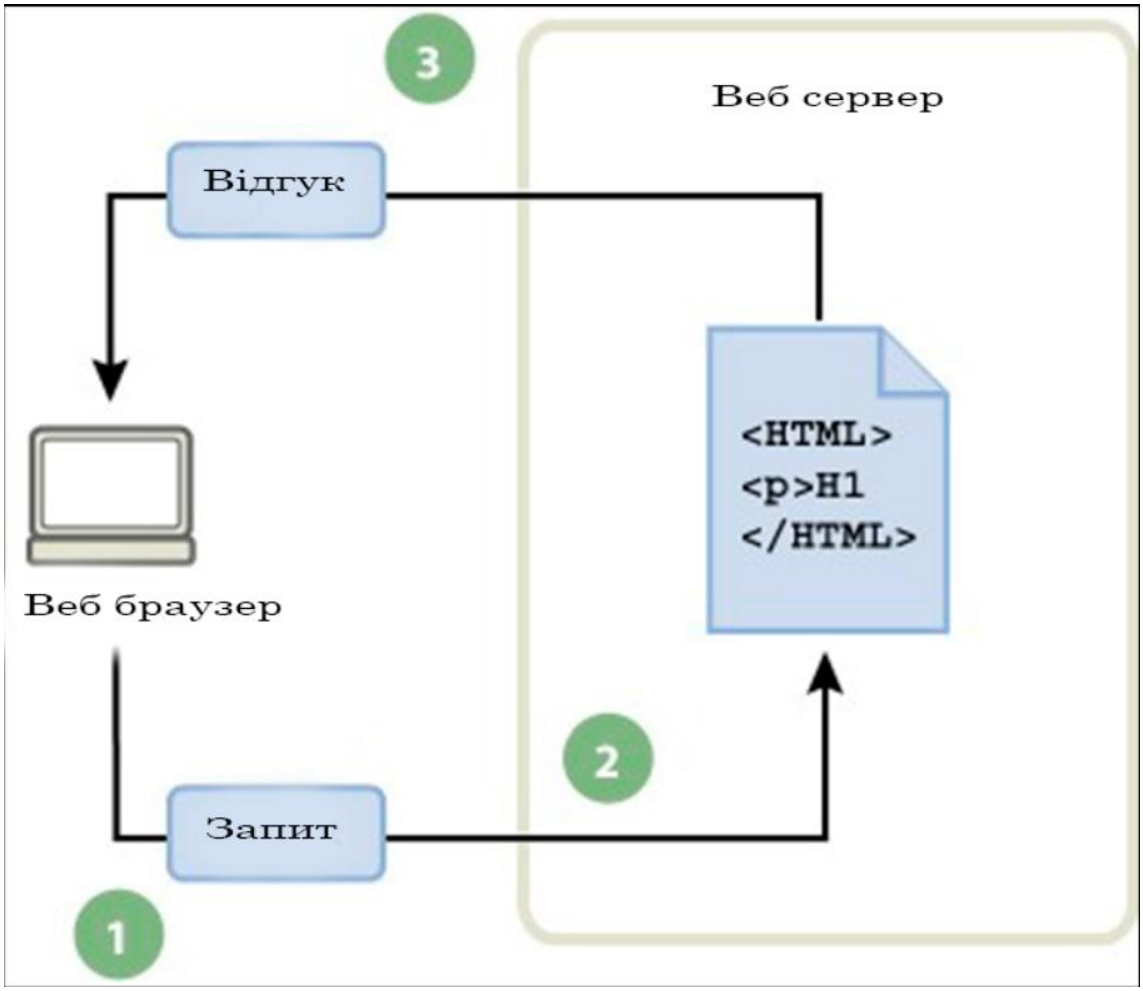


Рисунок 1.1 - Процес роботи статичного веб-додатку

Наступний тип це динамічні веб-додатки. Переважна кількість розробників вважають, що цей вид один з найкращих веб-додатків, через те, що вони отримують дані в режимі реального часу на основі запиту користувачів. Також вони мають покращену технічну складність порівняно зі статичними веб-додатками.

Принцип роботи цих додатків полягає в тому, що коли веб-сервер отримує запит на певну сторінку, запит на сторінку надходить до програмного забезпечення, відомого як сервер додатків. Бази даних знаходяться на стороні сервера, що забезпечує можливість користувачеві отримувати оновлений вміст.

Тому для цього типу веб-додатків просто необхідна база даних(рисунок 1.2) для зберігання даних і коли користувач отримає доступ до них то її вміст оновиться, і це буде відбуватися щоразу. Це можливо за допомогою системи

управління вмістом, такої як WordPress, в якій присутня вбудована панель адміністрування.

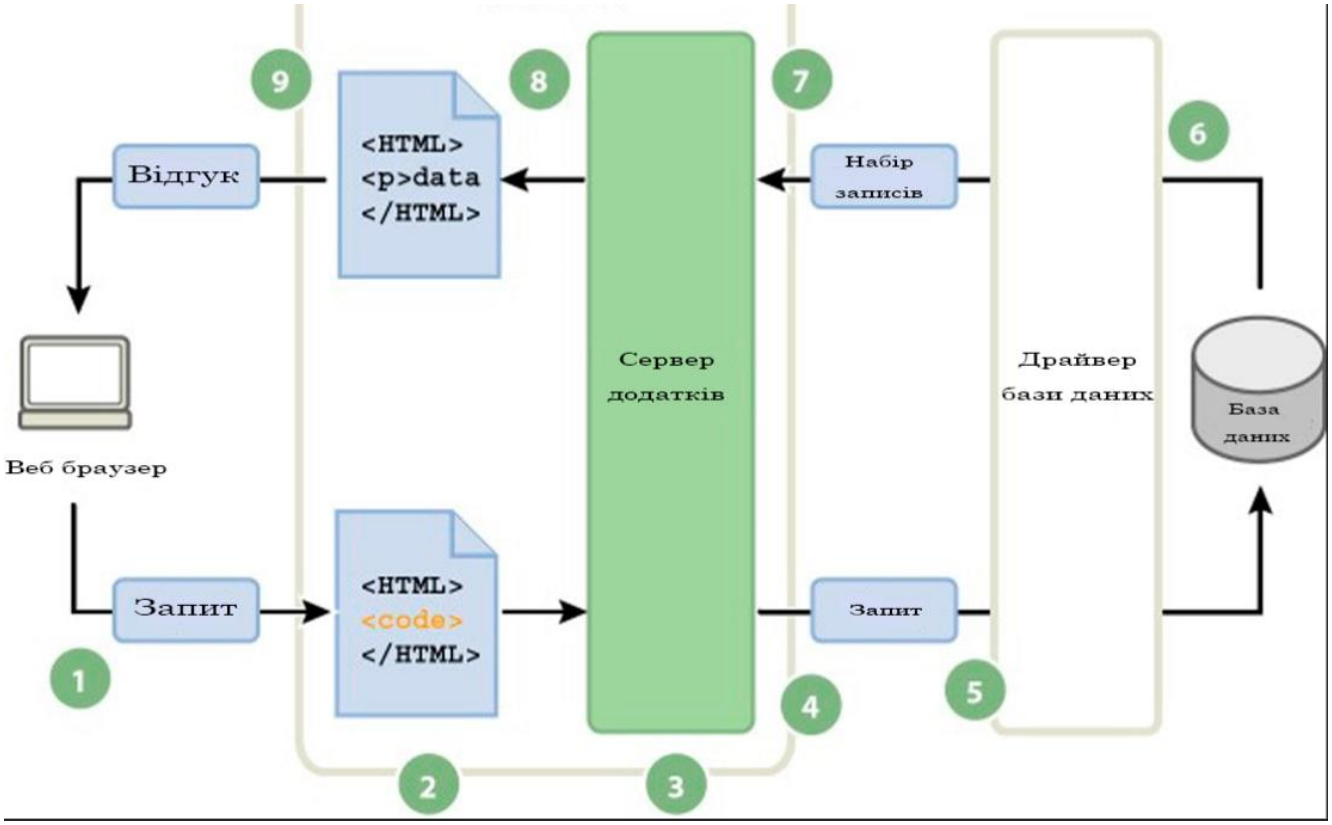


Рисунок 1.2 - Процес роботи динамічного веб-додатку

Для динамічних веб-додатків використовуються різні типи мов, серед них такі як Node.js, Ruby on Rails, jQuery, HTML, CSS, PHP, Perl, Python тощо. Для інтерфейсної розробки віддають перевагу Angular, React та JavaScript. Node.js та Laravel обирають розробники для внутрішньої розробки.

Односторінкові додатки. В свою чергу односторінкові додатки дозволяють користувачам безперешкодно взаємодіяти з веб-сторінкою. Тут запити та відповіді відбуваються ефективно завдяки невеликому обсягу даних. Тобто якщо порівнювати їх з іншими веб-додатками то системи односторінкових додатків набагато швидші, через те що вони лише виконують логіку у веб-браузері, а не на сервері. Також перевагою залишається і те що будь-які односторінкові додатки можна оновлювати відповідно до вимог у майбутньому.

Багатосторінкові додатки. Багатосторінкові додатки працюють аналогічно традиційним веб-додаткам, в яких додаток перезавантажує та відображає нову

сторінку із сервера в браузері в будь-який час, коли якийсь користувач виконує нову дію. У цих типах веб-додатків логіка зберігається у серверній системі, отже запити від клієнтів відправляються на сервер, після повертаються. Сереж популярних мов програмування для створенння багатосторінкових додатків використовують, такі як HTML, CSS, JavaScript, AJAX, jQuery тощо.

Портальні веб-додатки. Ще один тип серед веб-додатків це портальні веб-додатки. Притаманна їм особливість це головна сторінка на якій доступні різні розділи або категорії. Раніше згадана сторінка складається з різних деталей, таких як чати, електронні листи, форуми, реєстрація користувачів тощо. Також враховуючи розподілений доступ конкретні функції можуть бути обмежені певними користувачами. Для створення портальних веб-додатків використовують мови програмування, такі як Node.js, Angular.js, Laravel, ASP.net тощо. Саме вони найбільш зручні для створення веб-порталу.

Анімовані веб-додатки. В анімованих веб-додатках регулярно використовується технологія FLASH. При створенні таких типів веб-додатків аби представити вміст, використовують різні анімовані ефекти. Це приводить до повної свободи дизайнерів UI / UX, що робить додаток більш креативним та інтегрує деталі, які неможливі у інших типах веб-додатків.

Насичені Інтернет-додатки. Насичені Інтернет-додатки – це переважно додатки, що мають кілька функцій настільних програм. Вони використовуються для вирішення обмежень браузера, в той же час в них є залежність від плагінів на стороні клієнта, таких як Flash, Shockwave та Silverlight. Враховуючи те що ці додатки створюються за допомогою цих інструментів, їх робота доволі ефективна і дуже цікава. Більше того, насичені Інтернет-додатки забезпечують вражаючий досвід користувачів та високу інтерактивність у порівнянні з традиційними програмами браузера.

Але варто зауважити що в вище згаданих додатках є і проблеми. Їх лише дві, а саме вразливість та незручності, які вони створили. Наприклад, припустимо таку ситуацію, що плагін застарів, тоді кілька частин додатку або і весь додаток не буде працювати правильно.

В списку різних інструментів за допомогою можна побудувати насичені Інтернет-додатки відзначаються Adobe Flash, Adobe Flex, JavaFX тощо.

Веб-додатки на базі JavaScript:

Дані веб-додатки побудовані за допомогою провідних фронтенд-фреймворків JavaScript, а саме таких як Angular.js, React.js, Vue.js, Node.js та інші. Вищезазначені додатки пропонують підвищену продуктивність, також різні рівні взаємодії з користувачем та у більшості випадків оптимізовані для SEO. На даний момент часу загальноклієнтська логіка може виконувати різні серверні завдання, такі як обробка запитів користувачів, надання відповідей тощо.

Поступові веб-додатки. Поступові веб-додатки пердставляють веб-сайти які схожі на мобільні додатки. Користувачі мають можливість отримати доступ до всієї інформації та всіх функцій веб-додатка за допомогою будь-якого мобільного браузера.

Основна ідея цих додатків полягає в тому, що не відбувається застосування нових правил в архітектурі, а обраний шлях покращення швидкості та мобільну адаптованість веб-додатків. Також тут вдосконалено кешування, передачу даних та встановлення головного екрана.

До того ж, поступові веб-додатки дозволяють покращити мобільний досвід роботи в Інтернеті та надавати свої послуги користувачам, не беручи у рахунок повільне або ж погане підключення до Інтернету.

Веб-додатки електронної комерції. Розробка таких веб-додатків стає все більш складною, оскільки доводиться обробляти транзакції та інтегрувати різні способи оплати, такі як PayPal, дебетова або кредитна картка тощо.

До основних функцій веб-додатку електронної комерції слід включити додавання нових продуктів, видалення старих продуктів, обробку платежів, зручний інтерфейс тощо. Для контролю над усіма цими завданнями, адміністратору потрібна ефективна панель управління.

Тому незважаючи на вибраний тип веб-додатку слід пам’ятати, що веб-додаток - це не мобільний додаток, навіть не беручи до уваги те, що вони здаються ідеальними. Як і при розробці веб-сайтів, є необхідність впровадити

правила щодо файлів cookie та покращити захист додатків від будь-яких кібератак або загроз.

1.2 Обґрунтування вибору апаратних ресурсів, мови програмування та CASE-засобів

Процес створення належного веб-застосунку не може бути задумом, оскільки він може створити або зламати проект. Кожна велика частина програмного забезпечення починається з плану та чіткого процесу. Крім цього існує безліч процесів розробки веб-додатку, але єдине залишається одне це етапи життєвого циклу розробки. Більшість виділяє 7 основних етапів(рисунок 1.3). Серед них:

1) планування проекту;
2) збір та аналіз вимог;
3) дизайн;
4) кодування або реалізація;
5) тестування;
6) розгортання;
7) технічне обслуговування та оновлення.

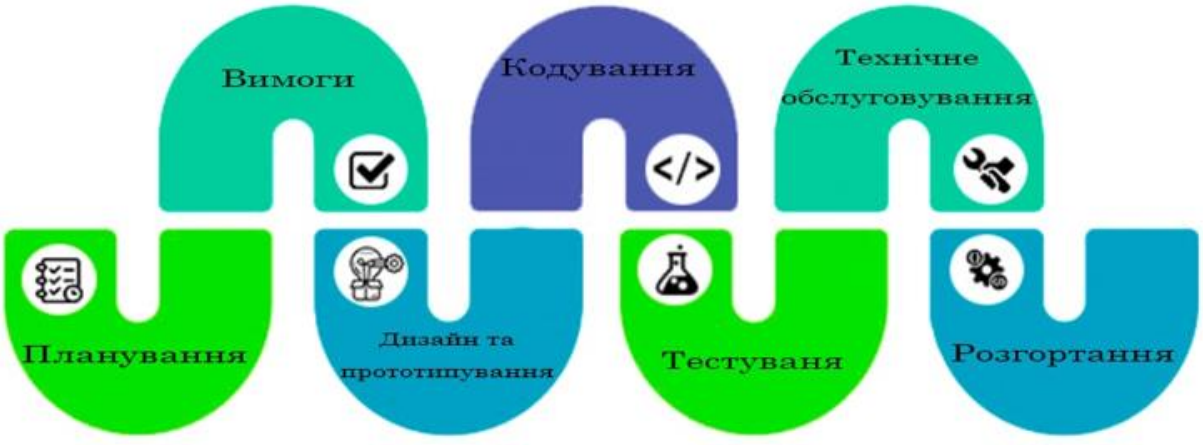


Рисунок 1.3 - 7 етапів життєвого циклу розробки

Незважаючи на те, що представлені вище етапи, можуть здатися покроковим планом створення веб-додатку, це насправді більше орієнтир. Звісно ж потрібно пройти кожний етап, щоб переконатися, що ви розробляєте та підтримуєте веб-додаток правильно. Але як саме, коли і в якому порядку – вибір залишається вільним. Протягом багатьох років було оформлено ряд різних процесів розробки програмного забезпечення для вирішення все більш складних проектів.

Серед усіх процесів розробки виділяють 5 найкращих, а саме:

1) Waterfall;
2) Agile and Scrum;
3) Incremental and Iterative;
4) V-Shaped;
5) Spiral.

У кожного процесу розробки є свої переваги та недоліки, а також різні ситуації під які обирається певний процес.

Процес розробки Waterfall також відомий як класична модель життєвого циклу є однією з найстаріших і найбільш традиційних моделей. У найосновнішій формі може бути представлений як послідовність виконання кожного кроку SDLC (рисунок 1.4), де потрібно закінчити кожен, послідовно, перш ніж рухатися далі. Однак на практиці у більшості випадків етапи злегка перекриваються, і зворотній зв'язок та інформація передаються між ними.

Завдяки своїй жорсткій структурі та великому часу попереднього планування, процес розробки Waterfall найкраще працює, коли цілі, вимоги та стек технологій навряд чи кардинально зміняться під час процесу розробки.

Хоча вище згаданий процес є простим, найбільшим недоліком є відсутність гнучкості. Тобто неможливо створювати та тестувати мінімально життєздатний продукт чи прототипи та змінювати думку на цьому шляху. І через це, якщо область дії не написана чітко, ви можете в кінцевому підсумку зробити помилковий шлях, не знаючи про це до дня запуску.

Рисунок 1.4 - Модель життєвого циклу Waterfall

Хоча вище згаданий процес є простим, найбільшим недоліком є відсутність гнучкості. Тобто неможливо створювати та тестувати мінімально життєздатний продукт чи прототипи та змінювати думку на цьому шляху. І через це, якщо область дії не написана чітко, ви можете в кінцевому підсумку зробити помилковий шлях, не знаючи про це до дня запуску.

На відміну від суворого, послідовного потоку процесу Waterfall, у Agile міжфункціональні команди працюють у "спринтах" від 2 тижнів до 2 місяців, щоб створювати та випускати корисне ПЗ або веб-застосування для клієнтів і отримувати зворотній зв'язок. В той же час процес розробки Agile і його найпопулярніша методологія - Scrum вибирає ітеративний та динамічний підхід до розробки.

Agile - це швидка розробка, частий випуск нової версії проекту і реагування на реальні потреби користувачів, навіть якщо це суперечить початковому плану. Це означає, що перед початком роботи не потрібен повний перелік вимог та повний звіт про роботу. По суті, відбувається покроковий рух в одному напрямку з розумінням того, що будете змінюватись курс на шляху розробки(рисунок 1.5).

Завдяки своєму динамічному та орієнтованому на користувача характеру, Agile - це процес розробки, який підтримують більшість стартапів та технологічних компаній, які тестують нові продукти або постійно оновлюють дані.

Рисунок 1.5 - Модель життєвого циклу Agile

По мірі того, як стає легше робити невеликі випуски та збирати відгуки користувачів, Agile дозволяє компаніям рухатись швидше та перевіряти теорії, не ризикуючи всіма засобами для існування під час великого випуску, який ненавидять їх користувачі. Крім того, оскільки тестування відбувається після кожної невеликої ітерації, простіше відстежувати помилки або повертатися до попередньої версії продукту, якщо була допущена критична помилка.

З іншого боку, динамічний характер Agile означає, що проекти можуть легко перевищувати початкові терміни та бюджет, або створювати конфлікти з існуючою архітектурою чи збиватися з плану через неправильне управління. Це означає, що це не найкращий вибір для команд, які не ризикують або не мають ресурсів.

Крім того, використання Agile and Scrum вимагає відданості та глибокого розуміння основного процесу, щоб правильно вийти. Ось чому важливо мати у своїй команді принаймні одного спеціалізованого майстра Scrum, щоб переконатися, що спринти та етапи вражаються, а проект не зупиняється.

У свою чергу Incremental і Iterative процеси розробки програмного забезпечення є щось середнім між структурою та попереднім плануванням процесу Waterfall та гнучкістю Agile.

Хоча обидва процеси дотримуються ідеї створення невеликих частин проекту та надання їх користувачам для зворотнього зв'язку, вони відрізняються тим, що ви створюєте під час кожного випуску.

Обидва процеси додають певний рівень гнучкості процесу розробки, не відходячи від загального плану, що робить їх ідеальними для великих проектів із визначеним обсягом або для команд з меншою толерантністю до ризику.

Завдяки додатковому процесу є можливість отримувати ранні відгуки про основну функцію, що допомагає негайно перевірити працездатність. Тоді як ітераційний підхід дає користувачам можливість швидко переглянути, яким може бути весь продукт. Це надає можливість отримувати кращі та цілеспрямованіші відгуки.

Рисунок 1.6 - модель життєвого циклу Incremental and Iterative

На жаль, спроба додати структуру до гнучкого підходу має і свої проблеми. Можливо, цілі, процедури чи технології ІТ-компанії з часом змінюються, роблячи попередні ітерації марними або з дефектами. Або ще 1 варіант розвиток подій, що код стає надто великим та неефективним через додавання функціональності.

Крім того, обидві моделі і особливо ітераційний підхід вимагають ретельного планування та побудови архітектури на ранніх стадіях розробки як продемонстровано на рисунку 1.6. Це означає, що вони не ідеальні для невеликих проектів або команд, які все ще випробовують нові варіанти побудови проекту.

Наступний процес розробки це - V-Shaped. Він є класичним методом Waterfall, який компенсує його найбільший недолік, а саме відсутність тестування. Замість того, щоб послідовно працювати над процесом розробки та зберігати все тестування до кінця, за кожним етапом V-Shaped процесу, є суворий крок перевірки, де вимоги перевіряються перед тим, як рухатись далі.

V-Shaped процес розробки ПЗ чудово підходить для невеликих проектів із відносно чіткими і статичними вимогами та обсягом. Замість того, щоб

ризикувати та дотримуватися плану лише для того, щоб знайти проблеми в самому кінці, він же надає широкі можливості для тестування на цьому шляху.

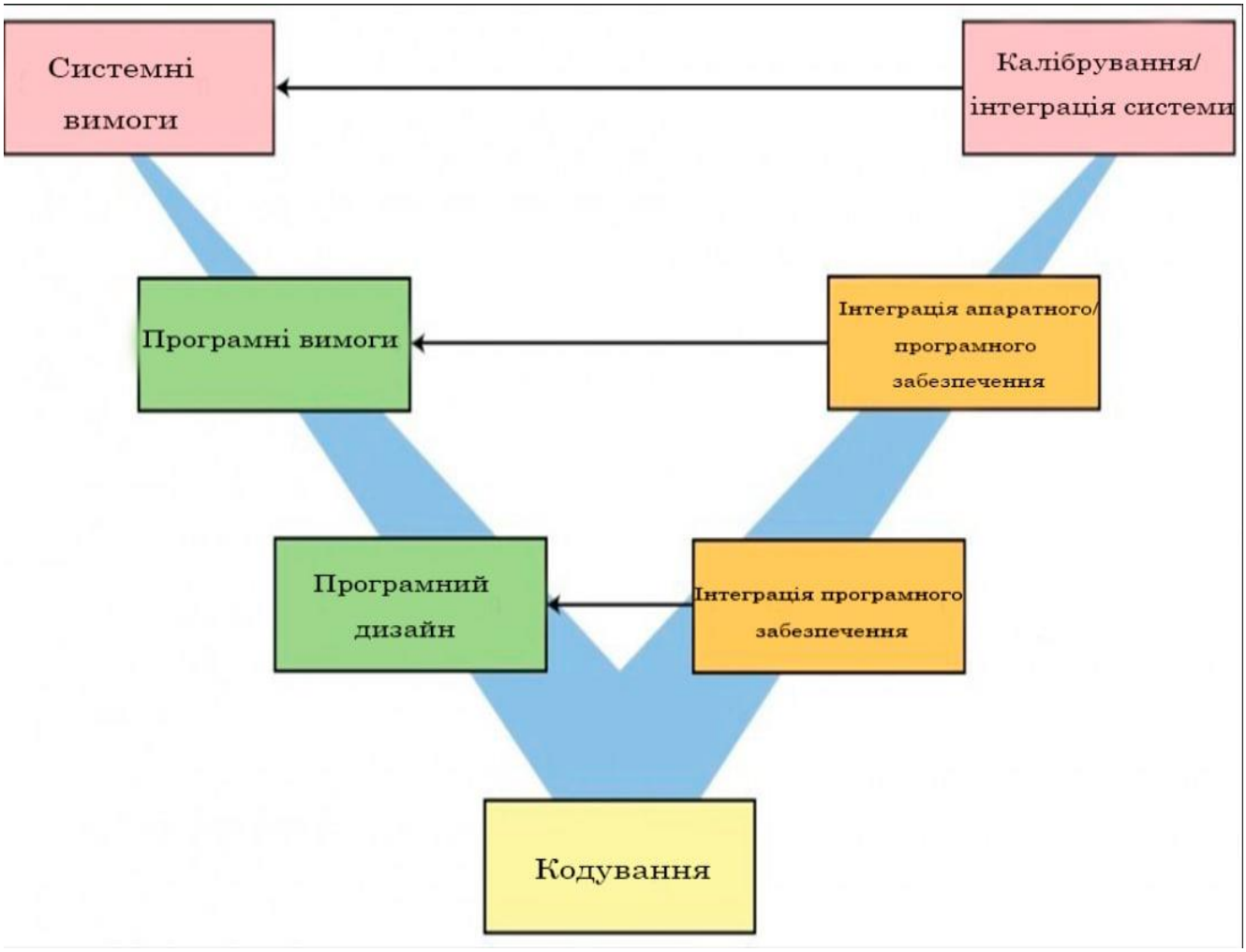


Рисунок 1.7 - модель життєвого циклу V-Shaped

У свою чергу процес розробки програмного забезпечення Spiral поєднує фокус V-Shaped процесу на тестуванні та оцінці ризику від Incremental і Iterative процесів(рисунок 1.7).

У вище зазначеному процесі після складання плану конкретної ітерації або етапу, наступним кроком є проведення поглибленого аналізу ризику для виявлення помилок або сфер надмірного ризику. Наприклад, як частину свого плану була придумана функція, яка не була перевірена клієнтами. Замість того, щоб просто додати його до поточного етапу, буде можливість створити прототип для тестування з користувачами, перш ніж переходити до повної фази розробки.

Після завершення кожного етапу сфера розширюється далі як спіраль, і ви починаєте з планування та іншої оцінки ризику.

Хоча теоретично фантастичний, спіральний процес розробки ПЗ рідко застосовується на практиці через час та витрати. Натомість його використовують як приклад як критично думати про ітераційний підхід до розвитку.

Зрештою, який процес розробки не був би вибраний, все зводитиметься до цілей, розміру проекту, взаємодії між різними командами при розробці. Адже при розробці ПЗ чи веб-застосування задіюються команди з різними ролями. Серед найбільш поширених:

1) розробники;
2) менеджмент;
3) фахівці з тестування;
4) аналітики;
5) адміністрування;
6) маркетинг.

Важко заперечувати, що саме тестування відіграє життєво важливу роль у розробці. У кожній компанії тестування є важливим і цінним етапом життєвого циклу розробки програмного забезпечення чи веб-додатку. Методи, що використовуються для тестування, відрізняються від компанії до фірми. Тестування одне із самих складних завдань. Кожного дня виникатимуть труднощі як у кодуванні, так і в декодуванні.

Зараз тестування програмного забезпечення стало частиною програмування, завдяки чому розробники можуть виправляти помилки з самого початку. Це етап, коли розробники знаходять помилки в програмному забезпеченні та роблять програму без помилок.

Роль тестування в розробці веб-додатку та ПЗ починається з підвищення надійності, якості та продуктивності. Це допомагає розробнику перевірити, чи працює програмне забезпечення правильно, і переконатись, що розроблюваний проект не виконує те, що не повинен робити.

Коли тестування впроваджується на ранній стадії розробки, вартість виправлення помилки значно менша. Виправлення помилки росте у ціні, коли помилку не вдається знайти в потрібний час.

Якість ПЗ чи веб-додатку відіграє життєво важливу роль у сьогоднішньому конкурентному світі. Отже, QA engineer відіграє ключову роль у життєвому циклі розробки для виробництва якісних продуктів. Тестування є невід'ємною частиною життєвого циклу розробки для підвищення узгодженості та продуктивності програмного забезпечення. Етапи тестування мають велике значення в життєвому циклі розробки, оскільки вони відіграють велику роль у виконанні та усуненні несправностей.

Послуги тестування програмного забезпечення допомагають розробнику дізнатися різницю між фактичним та очікуваним результатом, що сприяє поліпшенню якості продукту.Також без належного тестування новостворений продукт може бути небезпечним для користувачів. Вони будуть розгублені та втратять довіру до цього конкретного веб-додатку.

Хороше та ефективне тестування програмного забезпечення допомагає покращити безпеку. Аутентифікація та перевірка - головна мета служб тестування програмного забезпечення. В основному тестування програмного забезпечення допомагає не тільки виявити дефекти програмного забезпечення, але й його конфігурацію. За допомогою тестування розробники можуть визначити надійність програмного забезпечення.

Будь-який продукт можна перетворити на солідний і послідовний продукт за допомогою тестування при розробці веб-застосування або програмного забезпечення. У сучасному конкурентному світі кожному бізнесу важливо робити надзвичайно важливо створювати успішні проекти. У таких випадках тестування, одна з головних причин яка знижує ризики при розробці та збільшує вірогідність успіху у кінці.

1.3 Визначення вимог до системи автоматизації та розробка технічного завдання

Ретельне тестування має вирішальне значення для успіху програмного продукту. Якщо веб-додаток або програмне забезпечення не працює належним чином, велика ймовірність того, що більшість людей не купуватимуть його або використовувати, принаймні ненадовго. Але тестування на виявлення дефектів чи помилок - трудомістке, дороге, часто повторюване і спричинене людською помилкою. Автоматизоване тестування, за допомогою якого команди забезпечення якості використовують різні програмні інструменти для автоматичного запуску детальних, повторюваних та інтенсивних тестів, допомагає командам поліпшити якість продукту та максимально використати свої обмежені ресурси тестування. Інструменти автоматизації тестів, такі як довідкові команди TestComplete, тестують швидше, дозволяють протестувати значно більше коду, покращують точність тесту та звільняють інженерів з контролю якості, щоб вони могли зосередитися на тестах, які вимагають ручної тестування і які виконються лише за рахунок унікальних людських навичок.

Перевага автоматизованого тестування пов’язана з тим, що один і той самий тест можна повторювати необмежену кількість разів. Тести, які проводяться лише кілька разів, краще залишити для ручного тестування. Хороший Test case для автоматизації є той, який запускають часто і вимагають великих обсягів даних для виконання однакових дій.

Успіх у автоматизації тестів вимагає ретельного планування та проектування. У такій ситуації початком є створення плану автоматизації. Це дозволяє визначити початковий набір тестів для автоматизації та служити орієнтиром для майбутніх тестів. По-перше, слід визначити мету для автоматизованого тестування, а також визначити які типи тестів автоматизувати. Існує кілька різних типів тестування, і кожен з них має своє місце в процесі тестування. Наприклад, модульне тестування використовується для тестування невеликої частини передбачуваного додатка. Щоб протестувати певний фрагмент

користувацького інтерфейсу програми, буде використоване функціональне або графічне тестування.

Після визначення мети та типів тестів для автоматизації, наступний крок це вирішення, які саме дії будуть виконувати автоматизовані тести. Головне правило не створювати тестові кроки, які одночасно перевіряють різні аспекти поведінки програми. Великі, складні автоматизовані тести важко редагувати та налагоджувати. У такому випадку найкраще розділити тести на кілька логічних, менших тестів. Це зробить тестову систему автоматизації більш послідовною та керованою, крім цього дозволить обмінюватися тестовим кодом, тестовими даними та процесами. Результатом такої дії буде отримання більшої кількості можливостей для оновлення автоматизованих тестів, просто додавши невеликі тести, що стосуються нових функціональних можливостей. Ще одною вимогою до системи автоматизації є перевірка функціональностісті програми під час її додавання, а не очікувати, поки вся функція буде реалізована.

Наступна вимога це створення тестів, такими щоб вони були маленькими та зосередженими на одній меті. Наприклад, окремі тести лише для читання порівняно з тестами читання або запису. Це дозволяє використовувати ці окремі тести неодноразово, не включаючи їх у кожен автоматизований тест.

Після створення декількох простих автоматизованих тестів з'явиться можливість згрупувати тести в один, більший автоматизований тест, а після у єдину систему. Корисним буде і організація автоматизованих тестів за функціональною областю програми, основним і другорядним розділом програми, загальними функціями або базовим набором даних тесту. Якщо автоматизований тест буде посилається на інші тести, то буде необхідність в створенні дерева тестів, де можна запускати тести в певному порядку.

Також якщо надати унікальні імена для елементів керування, це зробить автоматизовані тести стійкими до цих змін інтерфейсу користувача та гарантує, що ваші автоматизовані тести працюють без необхідності вносити зміни в сам тест.

1.4 Висновки

Проведено аналіз існуючих програмно-технічних засобів предметної області, в результаті якого визначено було визначено різні моделі життєвого циклу розробки веб-додатку. Необхідні вимоги, а також рекомандації при створенні інформаційна системи для автоматизації тестування. Також було проаналізовано важливість створення такої системи і рівню автоматизації процесу тестування - автоматизоване тестування в цілому. Крім цього було проаналізовано роль тестування на різних етапах розробки веб-застосування.

2 ІНСТРУМЕНТИ ТА СПОСОБИ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ WEB-ЗАСТОСУВАНЬ

2.1 Вибір мови програмування для автоматизації

Завжди існує багато невизначеності, яку мову програмування використовувати під час запуску нового проекту автоматизації тестування. Чи варто йти з тією ж мовою, якою користується команда розробників? Або повинна вибратись мова, яка має велику кількість підтримки серед користувачів, щоб можливо було легко отримати допомогу, коли виникнуть проблеми при розробці? Це критичні фактори, на які звертають увагу при виборі мови програмування.

May 2021	May 2020	Change	Programming Language	Ratings	Change
1	1		C	13.38%	-3.68%
2	3	↑	Python	11.87%	+2.75%
3	2	↓	Java	11.74%	-4.54%
4	4		C++	7.81%	+1.69%
5	5		C#	4.41%	+0.12%
6	6		Visual Basic	4.02%	-0.16%
7	7		JavaScript	2.45%	-0.23%
8	14	⇈	Assembly language	2.43%	+1.31%
9	8	↓	PHP	1.86%	-0.63%
10	9	↓	SQL	1.71%	-0.38%
11	15	⇈	Ruby	1.50%	+0.48%
12	17	⇈	Classic Visual Basic	1.41%	+0.53%
13	10	↓	R	1.38%	-0.46%
14	38	⇈	Groovy	1.25%	+0.96%
15	13	↓	MATLAB	1.23%	+0.06%
16	12	⇊	Go	1.22%	-0.05%
17	23	⇈	Delphi/Object Pascal	1.21%	+0.60%
18	11	⇊	Swift	1.14%	-0.65%
19	18	↓	Perl	1.04%	+0.16%
20	34	⇈	Fortran	0.83%	+0.51%

Рисунок 2.1 - Список популярності мов програмування в 2021 році [1]

Існує буквально сотні мов програмування - від найпопулярніших і часто використовуваних(рисунок 2.1) до тих, які вже не використовуються. Наприклад, індекс TIOBE перелічує цілих 256 з них, починаючи від FoxPro і закінчуючи Z Shell. На рисунку 2.2 наведений графік популярності мов програмувань.

Вони варіюються від мов високого рівня до низькорівневих: останні є зручними для синтаксису машин, а перші легше розуміти та компілювати для людей. Java, JavaScript, Python є прикладами мов програмування високого рівня, тоді як C, C ++, без автоматичного управління пам'яттю та відсутністю надійних фреймворків, зараз вважаються мовами низького рівня.

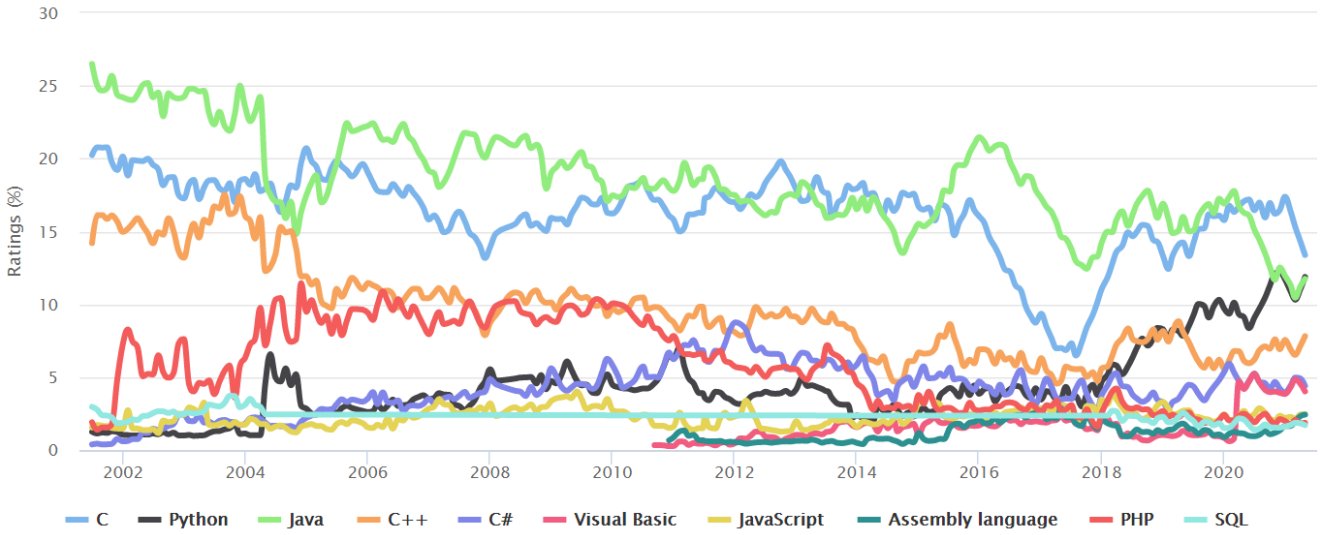


Рисунок 2.2 – Графік зміни популярності мов програмування в останні десятиліття [1]

Ці мови вирішують різні цілі. Наприклад, Python використовується для видобутку даних та візуалізації даних; його бібліотеки NumPy, SciPy, TensorFlow використовуються для розробки алгоритмів машинного навчання; для створення веб-додатків тощо.

Серед інших способів використання мов програмування є їх застосування в автоматизації тестів, підмножині служб контролю якості. Це дозволяє тестерам прискорити перевірку програмного забезпечення та розширити охоплення тестуванням.

З усіх мов програмування найпопулярнішими для автоматизації тестів виділяють 7 найкращих. Серед них:

1) Java;
2) Python;
3) JavaScript;
4) C#;
5) Rudy;
6) PHP;
7) SmashTest.

Загальні характеристики вище перечислених наведено в таблиці 2.1:

Таблиця 2.1 – характеристики мов програмування для автоматизації

Мова програмування	Java	Python	JavaScript	C#	Rudy	PHP	SmashTest
Простота навчання	1	2	6	3	4	7	5
Зручність використання	6	5	4	7	3	1	1
Підтримка користувачів	1	3	5	2	4	7	7
Популярність на ринку	1	6	5	2	3	7	7
Інструменти	2	4	5	1	3	6	7
Кількість робочих місць	1	3	4	2	5	7	6
Зарплата	1	3	6	2	4	7	5
Місце в рейтингу	1	3	4	2	5	4	6

Java - найпоширеніша мова програмування, що використовується для автоматизації тестів, яка належить корпорації Oracle. побудована на принципах об'єктно-орієнтованого програмування. Мова дотримується принципу WORA (Пиши один раз - запускай де завгодно), що приносить багато переваг між платформами. За статистикою більшість тестувальників, а саме 44%

використовують Java для своїх автоматизованих перевірок. Існує безліч легко доступних фреймворків, плагінів та освітніх ресурсів, які підтримують Java для автоматизації тестів – а за цим слідує те, що підтримка спільноти є рушійним фактором при виборі мови програмування для автоматизації тестів. Але також, згідно з опитуванням розробників за 2018 рік, за останній рік використання Java професійними розробниками зросло на 18%. Якщо команди узгоджують свої засоби автоматизації тестів із інструментами їх розробки продуктів, це також може пояснити популярність Java для перевірок інтерфейсу користувача.

Багато великих корпорацій використовують Java для обслуговування своїх внутрішніх систем. Існує більше 3 мільярдів пристроїв, на яких запущено програми, побудовані за допомогою Java. Незважаючи на те, що JUnit є популярною структурою модульного тестування, низка платформ тестування автоматизації з відкритим кодом розроблено за допомогою Java. Автоматизоване тестування браузера для веб-додатку можна виконати за допомогою JUnit разом із Selenium WebDriver. Широка платформа тестових платформ, пакетів та освітніх ресурсів Java робить її однією з найкращих мов програмування для автоматизації тестів. Він також забезпечує потужний командний рядок, просту інтеграцію збірки, підтримку IDE, серед іншого.

Python - це мова програмування з відкритим кодом для автоматизації тестів, машинного навчання тощо. Остання версія Python - 3.8.1. Головною перевагою Python перед іншими мовами програмування для автоматизації тестів є крива простоти вивчення завдяки читабельності мови.

Відповідно до опитування розробників яке відбулось у 2019, більшість, а саме 73,1% проголосували за Python як найбільш бажану мову програмування, що свідчить про популярність мови Python.

Нижче наведено деякі основні причини щодо популярності Python:

1) існує ряд бібліотек для мови програмування Python, і ці бібліотеки полегшують роботу розробника, оскільки вони допомагають виконувати дії, не використовуючи багато коду;

2) python набагато портативніший у порівнянні з іншими мовами програмування;

3) синтаксис Python легко засвоїти, що робить його ідеальною мовою програмування навіть для початківців;

4) велика спільнота яка використовує Python.

Бібліотеки Selenium та Appium для Python полегшують роботу з автоматизації та крос-браузерного тестування у настільних та мобільних пристроях. PyUnit та Pytest - це найпопулярніші фреймворки для тестування Python, які використовуються для тестування автоматизації Selenium для автоматизованого крос-браузерного тестування.

Хоча багато хто стверджує, що Python є чудовою першою мовою для тих, хто новачок у IT-сфері, і тому чудово підходить для ручних тестувальників, які наважуються на автоматизацію тестів, дані показують, що лише 8% тестувальників розробляють свої автоматизовані перевірки інтерфейсу користувача на цій мові.

Також об'єктно-орієнтованість і функціональність, дозволяє розробникам з'ясувати які саме функції або класи краще для даного завдання. Це корисно для автоматизації тестів, оскільки функції без стану запобігають побічним ефектам, а простий синтаксис робить їх читабельними.

JavaScript також є чудовою мовою програмування для автоматизації тестів, яка переважно використовується для інтерфейсної розробки. Багато великих веб-сайтів використовують JavaScript для інтерфейсної розробки, і він не менш популярний для тестування автоматизації.

Однією з основних причин його домінування в тестовій автоматизації може бути ширше впровадження методів shift-left testing, де розробники також беруть участь у розробці тестового коду. У методології shift-left команда QA працює у тісній співпраці з командою розробників, щоб запропонувати ефективну автоматизацію тестування.

Розробники також зазвичай використовують JavaScript разом із Selenium для тестових сценаріїв, пов'язаних з автоматизованим тестуванням браузера. Він також може використовуватися з віддаленою сіленевою сіткою, такою як LambdaTest, без значних змін у вихідному коді.

Наявність широкого спектру тестових фреймворків для модульного тестування та тестування E2E (End-to-End) робить JavaScript найкращою мовою програмування для автоматизації тестування. Деякі з найкращих платформ автоматизації тестів JavaScript:

1) Jest;
2) Mocha;
3) Jasmine;
4) Nightwatch JS.

Доступність цих фреймворків для створення та побудови тестових кейсів для наскрізного тестування та модульного тестування робить JavaScript необхідною мовою для автоматизації. Наприклад, фреймворк Jasmine надає перевагу тестуванню з використанням інтерфейсу, тоді як Mocha є кращим як для інтерфейсного, так і для тестування.

Крім цього JavaScript разом із Selenium WebDriver може виконувати автоматизоване тестування інтерфейсу користувачів програм. Також ця мова забезпечує ефективність завдяки своїм структурованим функціям і шаблонам, роблячи тестовий сценарій компактним.

Створений Microsoft, C# також є популярною мовою програмування для автоматизації тестів. Він розроблений на концепціях об'єктно-орієнтованого програмування. Це одна з найпопулярніших мов, що використовують фреймворк .NET. У 2019 році близько 67% респондентів в опитуванні розробників вважають, що C# є найулюбленішою мовою програмування для автоматизації тестів, веб-розробки тощо. C# як мова програмування для автоматизації тестів добре підходить для програм, які базуються на платформах Android, Windows та iOS. Остання версія C# - 8.0.

Ця мова програмування повільно та неухильно набирає обертів в області автоматизації тестів. Завдяки потужним особливостям мови та сумісності з Selenium WebDriver, багато тестувальників автоматизації схильні використовувати саме C# для автоматизації та крос-браузерного тестування. Використовуючи шаблон дизайну Page Object Model, тестувальники можуть запропонувати ефективний та ремонтопридатний тестовий код.

У C# є ряд систем тестування автоматизації, які допомагають у тестуванні автоматизації Selenium або автоматизованому тестуванні браузера. Через наявність широкого спектра тестових платформ, багато розробників розглядають C# для розробки тестових кейсів, пов'язаних із перехресним тестуванням браузера. Нижче наведені найбільш часто використовувані фреймворки автоматизації тестів на C#:

1) NUnit;
2) MSTest;
3) xUnit.Net.

Ruby - це ще одна мова програмування для автоматизації тестів, яка набирає обертів на арені автоматизації тестів та автоматизованого тестування браузера. Він має відкритий код і орієнтований на простоту та продуктивність. Як і Python, Ruby також легко вивчити та реалізувати. Зручний для людини синтаксис та гнучка об'єктно-орієнтована архітектура роблять Ruby потужною мовою програмування.

Ще одним цікавим аспектом є те, що Ruby - це підтримка зростаючого спільноти користувачів Ruby, яка вважається одним з важливішою властивістю мови. Поступово вона стає бажаною мовою програмування для створення веб-додатків. Розробники можуть створювати корисні програми на Ruby, використовуючи значно менші рядки коду.

Framework Selenium також працює з мовою Ruby, отже, його можна використовувати для тестування автоматизації Selenium. Початок роботи з Ruby та Selenium не є складним завданням, і ви можете виконати перший крос-браузерний тест із Selenium WebDriver & Ruby за допогою легких рядків коду.

В свою чергу PHP - це мова сценаріїв на стороні сервера, що використовується для веб-розробки. Однак він також добре використовується як мова програмування для автоматизації тестів. Якщо ви починаєте з програмування, вам неодмінно слід пройти PHP. Рівень складності PHP низький порівняно з іншими мовами програмування, такими як Python та Java.

PHP має велику підтримку серед спільноти та зростаючу екосистему. Він пропонує XDebug, потужний інструмент налагодження та профілювання, який

має потужні можливості. Він підтримує велику кількість систем автоматизації тестів, таких як:

1) Laravel Dusk;
2) Codeception;
3) PHPUnit;
4) BeHat.

BeHat представляє собою інструмент тестування з відкритим кодом, дозволяє видалити логіку з тесту. Codeception має високу швидкість виконання, в той час як вона використовується для приймального тестування, модульного тестування та функціонального тестування веб-додатків. У свою чергу PHPUnit це також інструмент тестування з відкритим кодом, який розширює тестовий функціонал. Laravel Dusk дозволяє програмно тестувати програми, включаючи тестування, якщо програма працює за бажанням із браузером.

Ще одною з мов програмування для автоматизації тестів, яка використовує NodeJS це – SmashTest. Це інструмент із відкритим кодом, що дозволяє швидко генерувати тести. Мова в 10 разів швидша за інші, але недоліком є документація цієї мови, яка є одною з самих складних в цьому списку. Але витративши час на досконале вивчення цієї мови, то вона стає простою для розуміння, яка має зручні для читання кроки та потужні функції звітування, які роблять її ідеальною мовою для автоматизованого тестування.

Після співставлення переваг та недоліків різних мов програмування, для інформаційної системи для автоматизації тестування Web-застосування було обрано об'єктно-орієнтовану мова програмування, а саме Java.

2.2 Порівняння інструментів та Java фреймворків для автоматизованого тестування

Коли йдеться про програмування в сучасному технічному світі, миттєво спадає на думку Java. Зрештою, вона вважається однією з найбільш універсальних мов програмування. Більшість розробників користуються Java завдяки незалежності платформи, безпеці, простоті використання, різноманітності

доступних ресурсів та ряду інших важливих функцій. Ці риси помітно сприяли популярності Java як мови програмування.

Якщо планується створення власної системи автоматизації тестів, потрібно обдумати вибір усіх інструментів та фреймворків. У більшості випадків краще розглянути один або кілька доступних варіантів з відкритим кодом.

Це тому, що загалом фреймворк - це набір найкращих практик, припущень, загальних інструментів та бібліотек, якими можна користуватися між командами. Одне з головних правил це не потрібно будувати такий, який буде унікальним для середовища розробників. Фреймворк допоможе зробити код автоматизації тестів багаторазовим, ремонтопридатним та стабільним, а також допоможе запобігти проект від дорогих витрат на виправлення дефектів. Зрештою, навіть незначні помилки можуть призвести до великих проблем.

Команди, які застосовують ці переваги, будуючи власні розроблені системи автоматизації з нуля, виконують зайву роботу. Це тому, що вони могли легко використати існуючі інструменти та бібліотеки з відкритим кодом, які б задовольнили їхні потреби без написання коду - і, в більшості випадків, з кращими результатами.

Хоча використання інструментів з відкритим кодом, як правило, кращий варіант, ніж створення власного фреймворку з нуля, але й не можна припускати, що інструменти автоматизації тестів з відкритим кодом - це все, що потрібно. Після того, як стане зрозуміло, як правильно вибрати інструмент автоматизації тестів для кожної ролі у організації, стане можливим отримати комбінацію комерційних та відкритих джерел.

Крім цього фреймворки модульного тестування Java надають програмістам стандартизовані, складні та розширювані засоби для створення веб-додатків або будь-якого іншого програмного забезпечення. Він складається з масивної колекції пакетів, що постачають заздалегідь написаний код. Залежно від вибору фреймворків для тестування на мові Java, вони будуть включати різні бібліотеки, компілятори, інструменти та API.

Більше того, розширені модульні тестові програми Java завжди забезпечують захист веб-додатку. Тож у випадку порушення в безпеці можна

вирішити це на веб-сайті без жодних проблем. Вони навіть пропонують вбудовані функції та модулі, щоб забезпечити ефективність та швидкість для розробників та тестувальників.

Серед переваг використання фреймворків для тестування Java також відзначають надійну підтримку. Фреймворки пропонують широку підтримку спільноти, де є можливість знайти рішення проблем та сумнівів протягом короткого періоду. Також серед переваг є витрати, а саме вартість обслуговування платформ Java є відносно низькою. Крім цього сам бюджет на розробку значно скорочуються.

На даний момент існує безліч платформ тестування Java, доступних для тестувальників якості. Зрозуміло, що чим більше вибір, тим складнішим є вибір найкращого. Виділимо 10 популярних фреймворків тестування на Java:

1) JUnit;
2) Selenium;
3) REST Assured;
4) TestNG;
5) Mockito;
6) Spock Framework;
7) Cucumber;
8) Spring Test;
9) DBUnit;
10) Google’s Robot Framework.

З усіх представлених вище фреймворків найбільшою популярністю до сих пір користується троє, а саме Junit, Selenium та TestNG. Тому саме їх буде взято для більш детального огляду.

Одна з популярних фреймворків для модульного тестування на мові Java залишається JUnit. Модульний тест використовується при виконанні декількох кроків сценарія або для тестування невеликого фрагмента коду. Він відіграє життєво важливу роль у тестовій розробці та є частиною колективних модульних тестових систем, що називаються xUnit. JUnit підсилює ініціативу «починати з тестування, а не з кодування», що наголошує на налаштуванні даних тестування

для частини коду, яку слід протестувати перед виконанням. Це підвищує продуктивність програміста та стабільність програмного коду, що, у свою чергу, зменшує тиск на програміста та мінімізує час, витрачений на налагодження.

Анотації JUnit 5.0 додали більшу потужність та зручність у модульному тестуванню. Анотації полегшують процес написання модульних тестів для вивчення винятків. Експерти-розробники, які дотримуються тестового підходу, повинні спочатку написати та запустити модульне тестування перед написанням подальшого коду. Після написання коду необхідно провести все тестування та перевірити відповідні результати. Після додавання нового тестового коду слід повторно виконати всі сценарії тесту, щоб переконатися, що все в порядку.

Хоча це примітивний спосіб тестування проектів на основі Java, він пропонує декілька переваг командам серед них ранній пошук помилок тобто фреймворк може легко знаходити помилки на ранній стадії порівняно з іншими системами автоматизації тестування. Коли помилка виявляється, вона вказується в окремому розділі, доки її не виправлять. Це допомагає зосередитись на налагодженні проекту. Крім цього JUnit це безкоштовна система тестування з відкритим кодом.

У свою чергу Selenium - це автоматизована платформа для тестування додатків із відкритим кодом, яка використовується для тестування між браузерами. Selenium досить потужна система управління та керування веб-браузерами за допомогою програми. Він функціональний майже для кожного браузера, працює в більшості популярних операційних систем, а його скрипти можуть бути написані такими популярними мовами програмування, як C #, Java, Python, PHP та інші.

Інтегроване середовище розробки Selenium надає можливість запису та відтворення для тестування та створення сценаріїв Selenium для подальшого використання. Підтримує паралельне виконання тесту, що підвищує ефективність та скорочує час виконання тесту. Його можна інтегрувати з такими фреймворками, як Ant, Maven тощо.

Це надійний фреймворк для управління веб-браузерами за допомогою коду автоматизації тестування. Ось що робить Selenium однією з найпотужніших платформ тестування:

Багаторазове використання та інтеграція - сітчастий LambdaTest на основі хмарної автоматизації Selenium дозволяє проводити тести на Selenium масштабно. Це допомагає зменшити інвестиції, необхідні для створення власної інфраструктури Selenium Grid. Селен можна легко інтегрувати з такими популярними системами автоматизації тестів, як JUnit, TestNG та інші.

Серед основних перевах використання саме цього фреймворку є те що перш за все, він робить ваші тести точними та стабільними, вирішуючи всі проблеми Ajax та синхронізації. Короткий процес написання тестових кейсів, який зменшує час очікування. Підтримує тести додатків, розроблені за допомогою AngularJS. Скорочує велику кількість команд, які раніше використовувались іншими традиційними інструментами Selenium.

Ще одною з популярних та потужних платформ тестування Java є TestNG, що використовується для інтеграційного, функціонального та модульного тестування. Він був створений Седріком Беустом у 2004 році, а тепер оновлений до версії 7. Це головний конкурент таких фреймворків, як JUnit та Selenium. TestNG схожий на JUnit, проте він налаштований з надзвичайними анотаціями та чудовими функціоналами які не підтримується JUnit.

Основними перевагами є це те що він дозволяє створювати та запускати паралельне тестування на декількох фрагментах коду. Під час реалізації тестового випадку ви можете створити звіт HTML. Тестові кейси можна легко організувати та згрупувати відповідно до пріоритетів. Виконувати тести набагато простіше, просто поставивши фреймворк для запуску інтерфейсного тестування або тестування баз даних. Пріоритети можна легко встановити за допомогою параметризації даних та використання анотацій.

JUnit вражає своїми можливостями, коли тест проводиться ізольовано однак, коли існує якась залежність, тоді не буде отримано ніякого контролю над тим, яке тестування проводилося раніше. Структури TestNG для Java допомагає тим, що дозволяє виконувати тестові випадки так, як того потребує тестувальник.

Крім того, надається додаткова підтримка параметризації в TestNG. Незважаючи на те, що ця особливість вже є в JUnit 4.5, рамка TestNG є більш корисною. Якщо будь-який проект є складним, і може знадобитися 100 або більше тестових кейсів, краще витратити трохи часу та налаштувати фреймворк TestNG, а не повністю залежати від інших фреймворків, таких як JUnit.

2.3 Взаємодія Java і Selenium при створенні автоматизованих тестів для веб-застосування

Кілька інструментів можуть керувати веб-браузером так, як це робить справжній користувач. До прикладу, навігація по різних сторінках, взаємодія з елементами сторінки або збір деяких даних. Цей процес називається автоматизацією веб-браузера. Те, що можна зробити з автоматизацією веб-браузера, повністю залежить від уявлень тестувалька та потреб на різних етапах розробки проекта.

Серед завдань які виконує автоматизація веб-браузера можуть бути:

1) автоматизація ручних тестів веб-додатків;

2) автоматизація повторюваних завдань, таких як скасування інформації з веб-сайтів;

3) заповнення HTML-форм, виконання адміністративних завдань тощо.

Один з найпопулярніших інструментів автоматизації веб-браузера - Selenium. Тому потрібно більше дізнатися про його функції, API та як саме його можливо використовувати його з мовою Java для автоматизації будь-якого веб-сайту.

Selenium собою представляє колекцію інструментів, що включає Selenium IDE, Selenium RC та Selenium WebDriver.

Selenium IDE - це по суті інструмент відтворення записів, який поставляється як плагін Firefox та розширення Chrome. Selenium RC був застарілим інструментом, який зараз знецінений. Selenium WebDriver - це найновіший і широко використовуваний інструмент.

Тут важливо зазначити, що Selenium створений для взаємодії лише з веб-компонентами. Отже, якщо ви зустрічаєте будь-які компоненти на робочому столі, такі як діалогове вікно Windows, Selenium сам по собі не може взаємодіяти з ними. Існують інші типи інструментів, такі як AutoIt або Robot Framework, які можна інтегрувати з Selenium для цих цілей.

Крім цього популярність Selenium при автоматизації браузера можна пояснити тим, що він не залежить від конкретної мови програмування і підтримує Java, Python, C #, Ruby, PHP, Perl тощо. Також присутня можливість написати свою реалізацію для цієї мови, якщо вона ще не підтримується. На успіх Selenium також вплинуло те, що специфікації WebDriver стали рекомендацією W3C для браузерів.

Перш ніж розпочати автоматизацію веб-браузера, корисно зрозуміти концепцій, яка породжує плутанину серед початківців. WebDriver - це не клас, це інтерфейс. Усі драйвери, що залежать від браузера, такі як ChromeDriver, FirefoxDriver, InternetExplorerDriver - це класи Java, що реалізують інтерфейс WebDriver. Ця інформація важлива, оскільки при запуску програми в іншому браузері, не потрібно буде міняти купу коду, щоб вона працювала, просто знадобиться замінити WebDriver на будь-який браузер, який ви хочете.

Спочатку визначимо шлях, тобто розташування драйвера браузера у файловій системі. Далі ми створимо “правильний драйвер” для цього браузера, ChromeDriver, у нашому випадку:

```
System.setProperty("webdriver.chrome.driver",
"C:\\QA\\Browser\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
```

Як видно в цій частині коду, драйвер містить посилання на ChromeDriver і тому може використовуватися для керування браузером. Коли виконується наведене вище твердження, у системі слід відкрити нове вікно браузера. Але браузер ще не відкрив жодного веб-сайту і саме цю роботу потрібно доручити

браузеру. Щоб використовувати інший WebDriver, потрібно вказати шлях драйвера у файловій системі, а потім створити його екземпляр.

Як зазначалося вище, спочатку потрібно перейти на цільовий веб-сайт. Для цього просто потрібно надіслати запит GET на URL-адресу веб-сайту:

```
driver.get("https://amazon.com");
```

Також варто візначити, що браузер, а у ньому сайт відкниється у зменшеному вікні. Тому для подальшої зручності використання варто використати команду maximize аби змініти розмір поточного вікна до повноекранного режиму:

```
driver.manage().window().maximize();
```

Першим кроком в автоматизації веб-браузера є пошук елементів на веб-сторінці, з якими є необхідність взаємодіяти, таких як кнопка, поле введення інформації, випадаючий список тощо.

Представленням Selenium таких елементів HTML є WebElement. Як і WebDriver, WebElement також є інтерфейсом Java. Отримавши WebElement, з'явиться можливість виконати будь-яку операцію над ним, яку в теорії міг виконати зробити кінцевий користувач, наприклад, клацання, друк, вибір тощо.

Очевидно, що спроба виконати недійсні операції, як-от спроба ввести текст в елемент кнопки, призведе до винятку.

Крім цього важливою функцією є використовування атрибутів HTML такого елемента, як id, class та name, щоб знайти елемент. Якщо таких атрибутів немає, ми можемо скористатися деякими вдосконаленими методами пошуку, такими як CSS Selectors та XPath.

Перевірити атрибути HTML будь-якого елемента, можливо після відкриття веб-сайту у браузері Chrome або іншому браузері який підтримує дану можливість. Після чого потрібно навести курсор миші на об’єкт атрибути якого

потрібно дізнатись та клацнути правою кнопкою миші та у списку, що з’явився обрати “Inspect”.

Рисунок 2.4 – Пошук атрибутів поля “Search”

На рисунку 2.4 продемонстровано, що елемент має тег <input> і кілька атрибутів, таких як id, клас тощо. Загалом WebDriver підтримує 8 різних локаторів для пошуку елементів:

1) id;
2) className;
3) name;
4) tagName;
5) linkText;
6) partialLinkText;
7) cssSelector;
8) xpath.

Для прикладу візьмемо код для пошуку елементів по атрибуту id. Якщо перевірити поле для пошуку на цільовому веб-сайті, то можна виявити, що він має атрибут id:

```
<input type="text" id="twotabsearchtextbox" value=""
name="field-keywords" autocomplete="off" placeholder=""
class="nav-input nav-progressive-attribute" dir="auto"
tabindex="0" aria-label="Search">
```

Для знайдення елементів на веб-сторінці і подальшої взаємодії з цим елементом цей елемент, використовуючи локатор ідентифікаторів:

```
WebElement newsletterEmail = driver.findElement (By.id
("електронна пошта"));
```

Після того як було знайдено елементи HTML на сторінці, і є отримана можливість отримати відповідний WebElement. Однак взаємодії з цими елементами ще було, як би це робив би кінцевий користувач, а саме клацання, набір тексту, вибір тощо. Тому розглянемо самі основні та прості дії з різними елементами сайту.

До прикладу, щоб виконати операцію клацання потрібно використати метод click (). Використовувати це можна на будь-якому WebElement, якщо його можна натиснути. Якщо ні, це викличе виняток.

```
WebElement element =
driver.findElement(By.id("searchbutton"));
element.click();
```

Оскільки це насправді є натисканням на сторінку, то веб-браузер перейде за посиланням, на яке було натиснуто програмно.

У свою чергу щоб ввести інформацію у поле необхіодно викристати команду sendkeys, наприклад:

```
WebElement element = driver.findElement(By.id("password"));
element.sendkeys("whoareyou");
```

Не рідкість ситуації коли може знадобитися навести курсор на пункт меню, який робить пункт підменю. Тому важливість взаємодії мишки і клавіатури вкрай важлива. Для цього більшість використовує команду moveToElement, також необхідно буде використати породжуючий патерн проектування - builder. Для прикладу:

```
WebElement elem = driver.findElement(By.id("button"));
builder.moveToElement(elem).build().perform();
```

Також оскільки багато веб-сайтів використовують файли cookie для зберігання стану користувача чи інших даних, може бути корисним програмний доступ до них за допомогою Selenium. Деякі загальні операції з файлами cookie описані нижче.

```
driver.manage().getCookies();
driver.manage().addCookie(mySavedCookie);
driver.manage().deleteCookie(targetCookie);
```

2.4 Висновки

В даному розділі було продемонстровано та проаналізовано різні інтрументи, з чого можна зробити підсумок, що Selenium WebDriver має безліч корисних методів для моделювання майже всіх взаємодій користувачів. Варто зазначити, що сучасні веб-додатки дійсно розумні. Якщо вони хочуть обмежити їх автоматичне використання, існують різні способи зробити це, наприклад, використання капчі. На жаль, Selenium не може обійти капчу. Тому при розробці веб-додатку чи сайту, на етапі тестування за допомогою автоматизованих тестів капчу деактивують, а вже коли проект протестований її добавляють. Ще одним недоліком є те що він не може взаємодіяти напряму з файловою системою ОС, але тут на допомогу приходить Java та Robot Framework. Отримуємо у результаті чудову комбінацію мови та інструмента для автоматизованого тестування які доповнюють друг друга.

3 ПРОГРАМНО-АПАРАТНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ

3.1 Реалізовані функціональні особливості

У результаті проектування було реалізовано інформаційну систему, що реалізує принципи та методи ручного через впровадження автоматизації Web-застосування. Окрім базових функції при автоматизації тестування було також задіяно обробку винятків, а саме конструкція try-catch. Вона намагається виконати інструкції в блоці try, і, у разі помилок, виконувати блок catch. Ось приклад коду, що застосовує обробку винятків:

```
try {
WebElement saveButton =
driver.findElement(By.xpath("//*[@id=\"privacy\"]/button"))
;
 builder.moveToElement(saveButton).click().perform();
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Save button is not
found");
        }
```

До прикладу, ось як буде виглядати результат, якщо станеться випадок якого не мало бути:

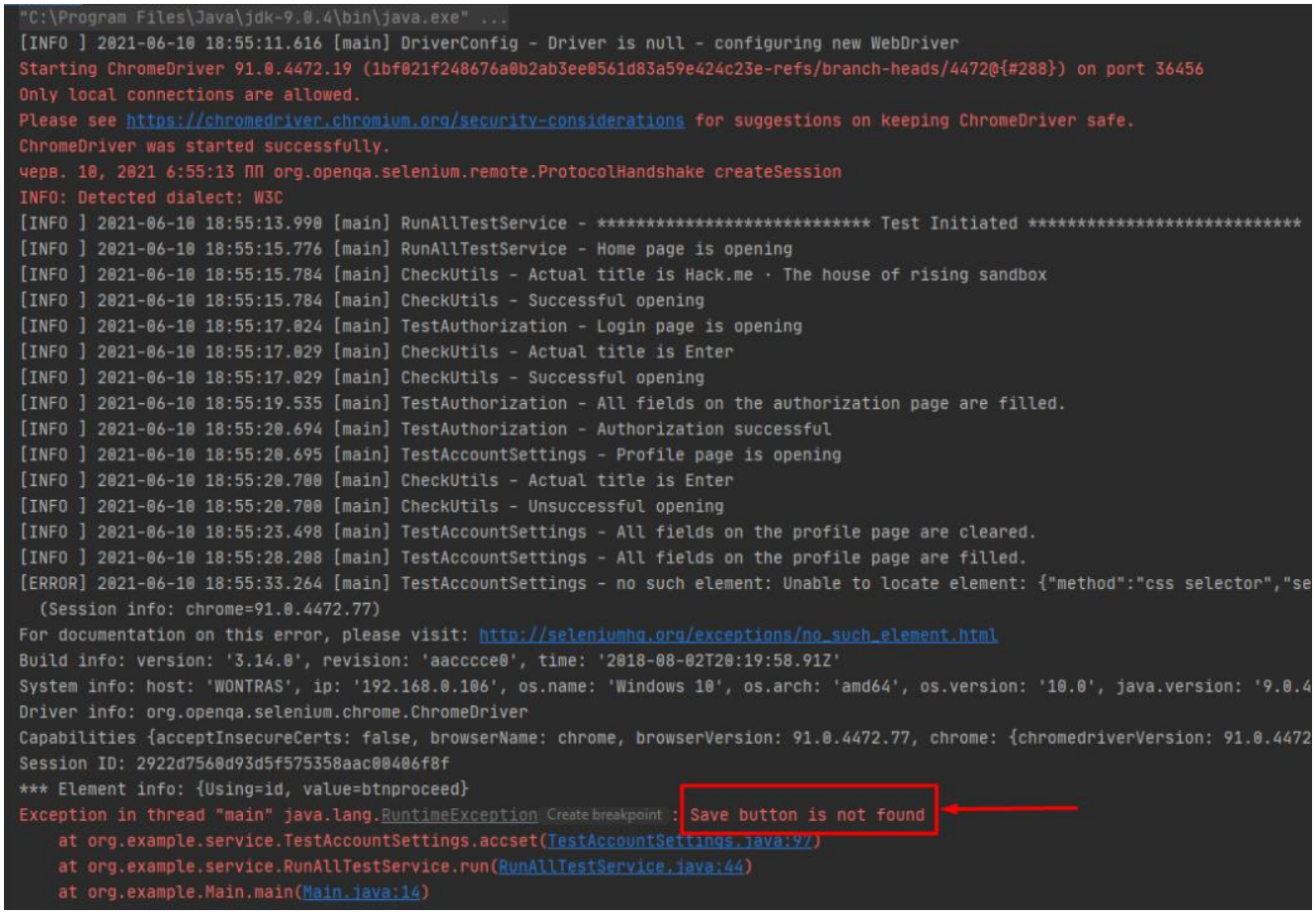


Рисунок 3.1 –Виникнення помилки та початок виконання блоку catch

Як видно на рисунку 3.1 після того як елемент сторінки saveButton не був знайдений за атрибутом HTML – xpath. І одразу після цього робота ПЗ зупиняється та відображається за допомогою логування інформація про час коли помилка сталася, який саме тестовий сценарій провалився, а також що саме сталось. У цьому випадку кнопка «Save» не була знайдена у веб-додатку. Тобто в основі всеодно лежить принцип try-catch при якому якщо виникає помилка, то виконання try на ній переривається, і управління стрибає в початок блоку catch (error). При цьому змінна error буде містити об'єкт помилки з докладною інформацією про те, що сталося.

Як вказувалось раніше, окрім обробки винятків, було також використано логування, а саме бібліотека журналювання Java програм - Log4j. Основна функція логування не використовуючи терміни вікіпедії полягає у тому, що це можливість стежити за процесом виконання бізнес-логіки проекту. Також варто зазначити це досить надійна, швидка та гнучка середа журналів (API), яка була

написана на Java. Він розглядає процес ведення журналу з точки зору рівнів пріоритетів та пропонує механізми для створення інформації про реєстрацію в самих різних пунктах призначення, таких як база даних, файл, консоль, системний журнал UNIX та інші. Перед початком його використання потрібно додати в pom.xml залежність:

```
<dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-core</artifactId>
            <version>2.7</version>
        </dependency>
        <dependency>
            <groupId>org.apache.logging.log4j</groupId>
            <artifactId>log4j-slf4j-impl</artifactId>
            <version>2.7</version>
        </dependency>
```

Крім цього щоб гнучко керувати логуванням варто створити в resources файл log4j.xml (рисунок 3.2):

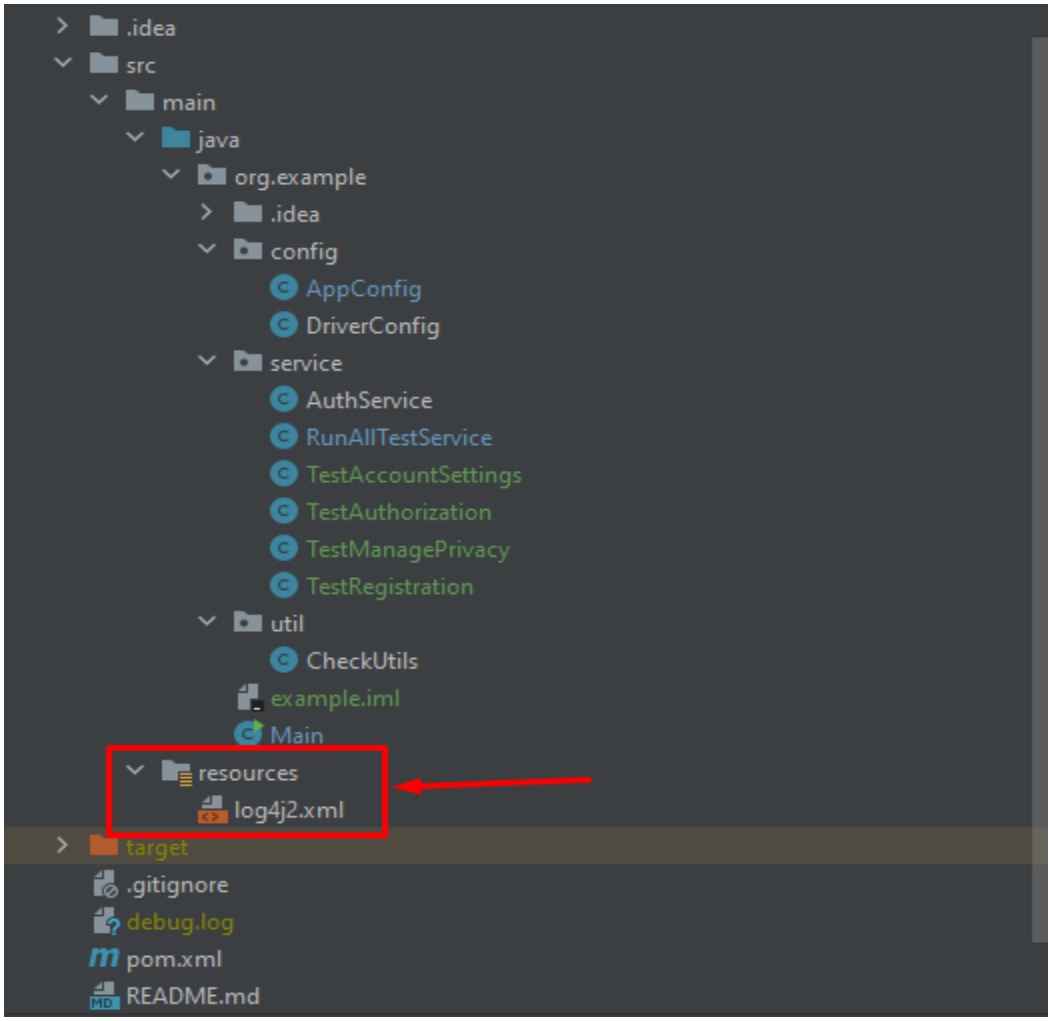


Рисунок 3.2 – Розміщення файлу log4j2.xml у папці resources

Ще один з інструментів який був використаний це Robot Framework. Який собою представляє систему автоматизації з відкритим кодом. Він може бути використаний для автоматизації випробувань та роботизованої автоматизації процесів. Robot Framework відкритий та розширюваний і може бути інтегрований практично з будь-яким іншим інструментом для створення потужних та гнучких рішень для автоматизації. Будучи відкритим кодом також означає, що Robot Framework можна використовувати безкоштовно без ліцензійних витрат. Один з прикладів його використання це встановлення взаємодії між файловою системою ОС, а також Selenium. Ще один зі способів це використання гарячих клавіш клавіатури для збільшення функцій взаємодії з веб-додатком. Для більш зручного використання Robot Framework буде доречно додати ключове слово Java – this. Щоб використати його для посилання на поточний екземпляр методу, в якому він використовується:

```
try {
            this.robot = new Robot();
        } catch (AWTException e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Robot framework is
not working.");
        }
```

Наступний приклад частини кода показує як саме один зі способів, а саме використання гарячих клавіш для очистки текстовго поля:

```
try {
            WebElement firstNameField =
driver.findElement(By.id("firstname"));

builder.moveToElement(firstNameField).clickAndHold().perfor
m();
            builder.pause(Duration.ofSeconds(1));
            robot.keyPress(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_DELETE);
            robot.keyPress(KeyEvent.VK_TAB);
            robot.keyPress(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_DELETE);
            robot.keyPress(KeyEvent.VK_TAB);
```

```
            robot.keyPress(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_DELETE);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Clearing text
fields failed.");
        }
```

3.2 Реалізація класів для тестових сценаріїв інформаційної системи

Під час проектування було визначено, що для прикладу створення інформаційної системи для автоматизації тестування буде реалізовано для Web-застосування https://hack.me/(рисунок 3.3)

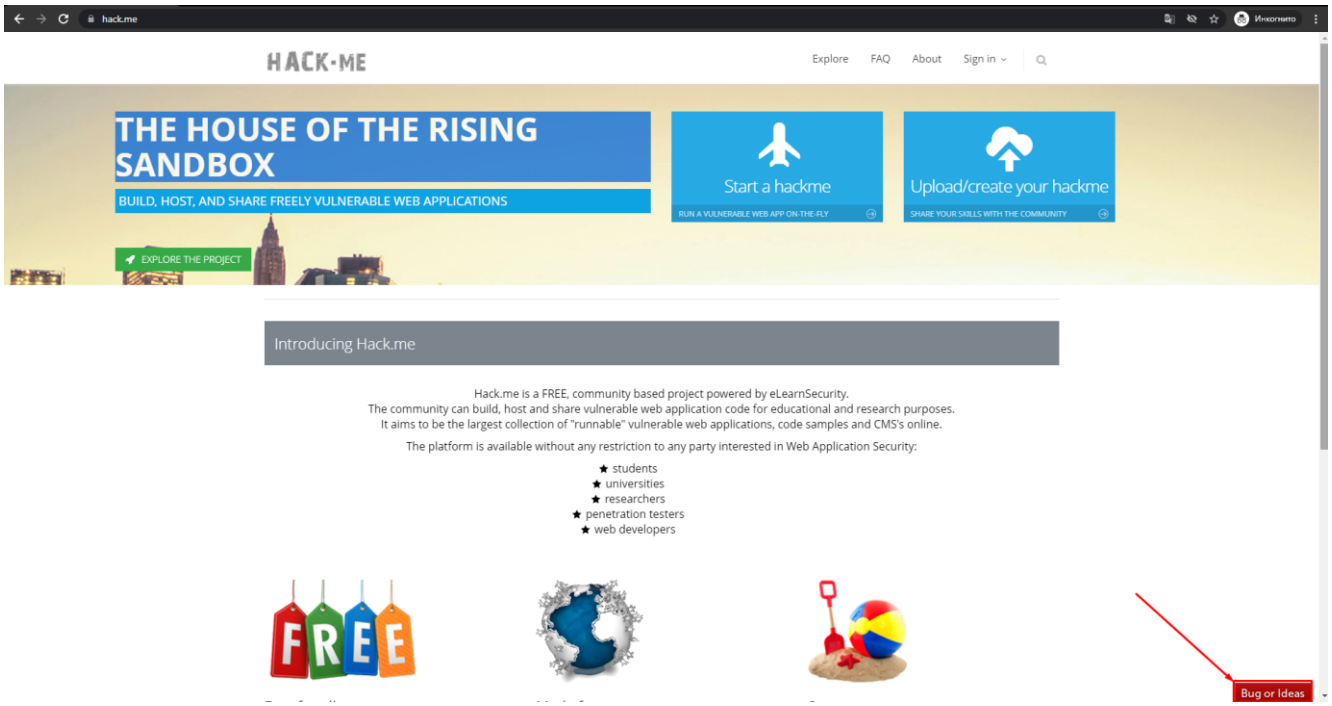


Рисунок 3.3 – Головна сторінка Hack.me

До того ж самі розробники не проти якщо їх повідомлять про помилки які будуть знайдень у результаті випробувань на їхньому веб-додатку. На ньому доступно декілька сторінок, у ході створення інформаційної системи було створено тестові сценарії для сторінок:

1) реєстрації;
2) авторизації;
3) настройка облікового запису.

Отож, було розроблено декілька класів для кожного тестового сценарію. Крім цього було створено окремий клас RunAllTestService який дозволяє визначати порядок у якому будуть відтворюватися всі тестові сценарії, а також добавляти чи видаляти з черги. Але незважаючи на це точкою входу в програму всеодно залишається метод main (). Підпис методу завжди залишається:

```
public static void main (String [] args)
```

Аргументи командного рядка передаються через параметр args, який є масивом String s.

Для того щоб впровадити зручність та не дублювати код до пакету config було додано два класи, а саме DriverConfig та AppConfig. DriverConfig був створений щоб при кожному запуску усіх тестів, браузер запускався зі сталими параметрами. Нижче приведено частина коду з цього класу:

```
public static WebDriver getDriver() {
        if (driver == null) {
            LOGGER.info("Driver is null - configuring new
WebDriver");
            ChromeOptions options = new ChromeOptions();
            options.addArguments("user-data-dir=" +
PATH_TO_CHROME_PROFILE);

            //PATH TO CHROMEDRIVER
```

```
            System.setProperty(USE_DRIVER,
PATH_TO_CHROMEDRIVER);
            driver = new ChromeDriver(options);
            driver.manage().window().maximize();
            driver.manage().timeouts().pageLoadTimeout(5,
TimeUnit.SECONDS);
            driver.manage().timeouts().implicitlyWait(5,
TimeUnit.SECONDS);
        }
        return driver;

    }
```

Необхідно звернути увагу на те, що було зроблено копію шляху у файловій системі до даних профіля Chrome. Це було зроблено задля того, щоб відкрити сеанс перегляду Chrome з певними параметрами закладок, розширень, теми, файлів cookie тощо. Крім цього за замовчуванням Selenium відкриває браузер у вікні малого розміру. Тому був доданий постійний параметр який відкриває браузер у повноекранному режимі.

У свою чергу у класі AppConfig розміщено статичні змінні з рядковим типом даних. Що дозволяє зручно використовувати різні дані під час створення різних тестових сценарій. Нижче наведений код класу AppConfig:

```
public class AppConfig {
    public static final String USER_FIRST_NAME = "Sandro";
    public static final String USER_lAST_NAME =
"Botticelli";
    public static final String USER_ORGANIZATION =
"University";
    public static final String NEW_USER_FIRST_NAME =
"Alex";
```

```
    public static final String NEW_USER_lAST_NAME =
"Williams";
    public static final String NEW_USER_ORGANIZATION =
"Office";
    public static final String USER_EMAIL =
"bottichelli1445@gmail.com";
    public static final String USER_LOGIN = "Sandro";
    public static final String USER_PASSWORD =
"Password1!";
    public static final String PATH_TO_CHROME_PROFILE =
"C:\\Users\\Lenovo
L340\\AppData\\Local\\Google\\Chrome\\UserDataCopy";
    public static final String PATH_TO_CHROMEDRIVER =
"D:\\IDEA projects\\chromedriver_91\\chromedriver.exe";
    public static final String USE_DRIVER =
"webdriver.chrome.driver";
    public static final String HOME_PAGE_URL =
"https://hack.me/";
    public static final String REGISTRATION_PAGE_URL =
"https://me.hack.me/signup";
    public static final String LOGIN_PAGE_URL =
"https://me.hack.me/login";
    public static final String HACKME_REGIST_TITLE = "Sign
Up for Hack.me";
    public static final String HACKME_MAIN_TITLE = "Hack.me
· The house of rising sandbox";
    public static final String HACKME_LOGIN_TITLE =
"Enter";
    public static final String HACKME_PROFILE_TITLE =
"Members Area";
    public static final String START = "777";
```

```
}
```

Як видно з коду приведеного вище було використано ключове слово final. Яке може застосовуватися до класів, методам, змінним в тому числі аргументів методів. Для змінних рядкового типу це означає, що одного разу присвоєне значення не може бути змінено.

Окрім вище реалізованих рішень також був створений пакет util. В ньому розміщений клас CheckUtils який виконує декілька функцій. Одна з них це метод openWebsite який дозволяє його швидко та зручно використовувати його у різних тестовах сценаріях. Нище наведено його код:

```
public static void openWebsite(WebDriver driver, String
openURL) {
        driver.get(openURL);
    }
```

Ще один з методів який був доданий це – verifyTitle. Його необхідність в системі полягає в тому, що він захопилює поточний заголовок сторінки, а згодом він буде збережений у змінну, а потім буде опублікований у консолі. У нашому методі він же використовується для перевірки коректності відкриття сторінок. І в залежності від результату в логах буде відображено чи відкрилась сторінка яка повинна була чи ні. До прикладу, реалізація цього методу виглядає наступним чином:

```
public static void verifyTitle(WebDriver driver, String
expectedTitle) {
        String actualTitle = driver.getTitle();
        LOGGER.info("Actual title is {}", actualTitle);
        if (expectedTitle.equals(actualTitle)) {
            LOGGER.info("Successful opening");
        } else {
```

```
            LOGGER.info("Unsuccessful opening");
        }

    }
```

3.3 Розробка програмного забезпечення на базі реалізованих рішень та проведення тестування системи

В процесі розробки було прийнято рішення протестувати роботу готового рішення. Інформаційна система поетапно повинна була відтворювати тестові випадки на сторінці реєстрації, авторизації та настройки облікового запуску.

Отож на сторінці реєстрації покроково мають відворитися наступні кроки(рисунок 3.4):

1) заповнення текстовго поля First Name;
2) заповнення текстовго поля Last name;
3) заповнення текстовго поля Organization;
4) заповнення текстовго поля Email address;
5) заповнення текстовго поля Confirm email address;
6) заповнення текстовго поля Username;
7) заповнення текстовго поля Password;
8) заповнення текстовго поля Confirm password;
9) натискання на кнопку Proceed.

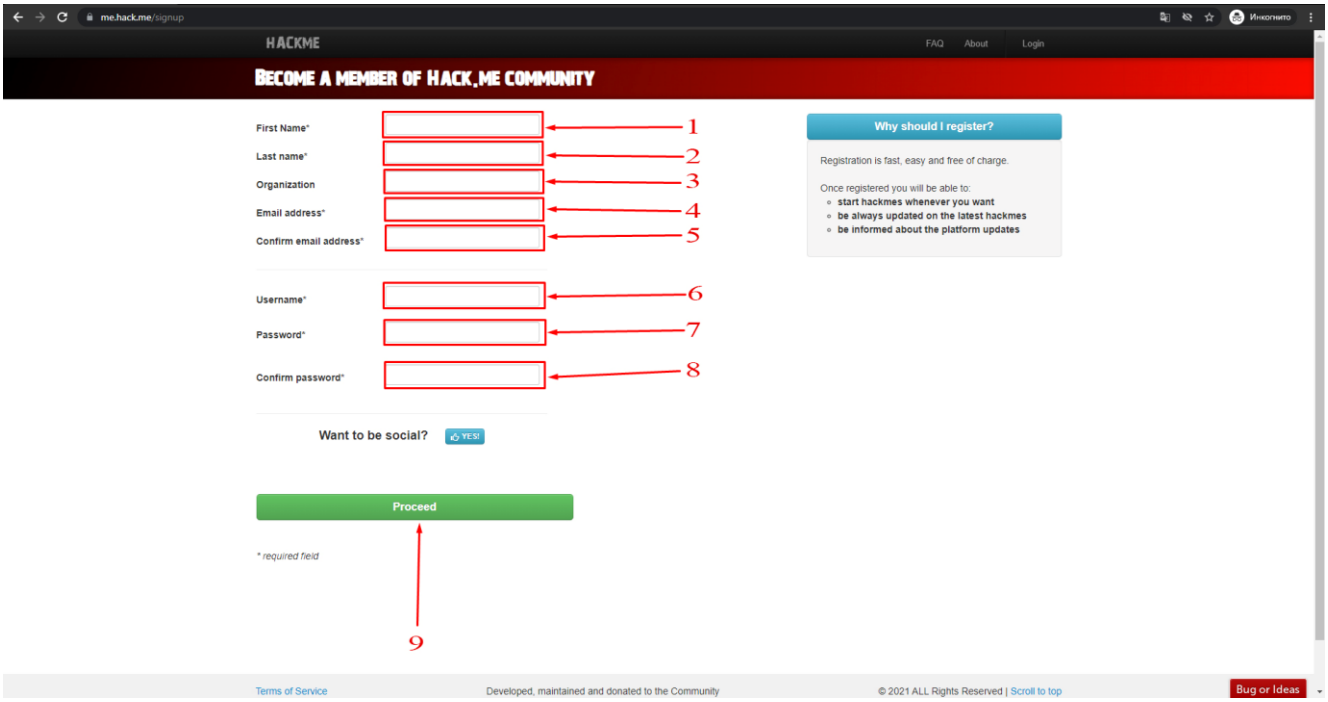


Рисунок 3.4 – Кроки тестового сценарію на сторінці реєстрації Hack.me

У нижче наведеному коді продемонстровано як саме реалізовано заповнення текстового поля та натискання на кнопку для веб-додатку:

```
try {
            WebElement firstNameField =
driver.findElement(By.id("name"));

builder.moveToElement(firstNameField).build().perform();

driver.findElement(By.id("name")).sendKeys(USER_FIRST_NAME)
;
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("First name field is
not found");
        }

try {
```

```
            WebElement proceedButton =
driver.findElement(By.id("btnproceed"));

builder.moveToElement(proceedButton).click().perform();
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Proceed button is
not found");
        }
```

Результати роботи виводились в консоль.

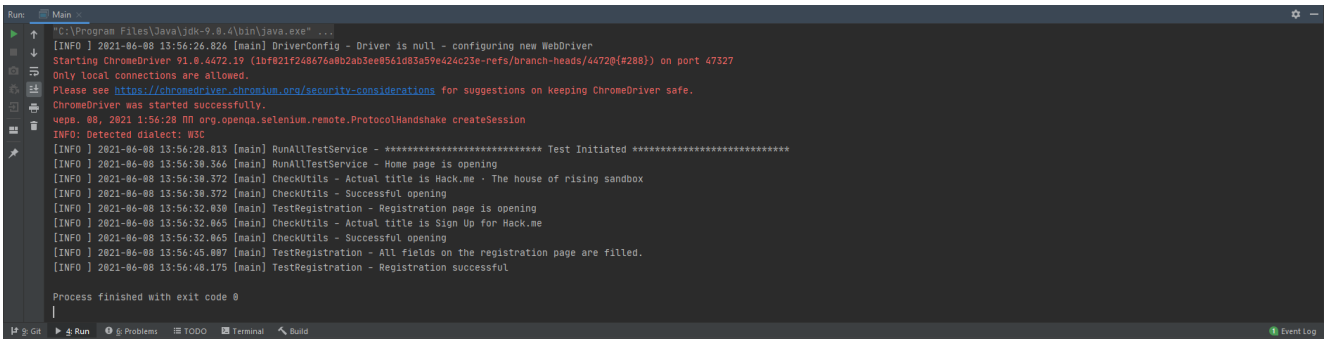


Рисунок 3.5 – Результат роботи класу TestRegistration

Як видно на скріншоті(рисунку 3.5), спочатку був успішно відкритий браузер із заданими параметрами, після чого була відкрита головна сторінка Hack.me. Наступним етапом був перехід на сторінку реєстрації та одразу відбулась перевірка методом verifyTitle. І кінцевим етапом стало заповнення кожного текстового поля та пітвердження реєстрації натисканням на кнопку Proceed.

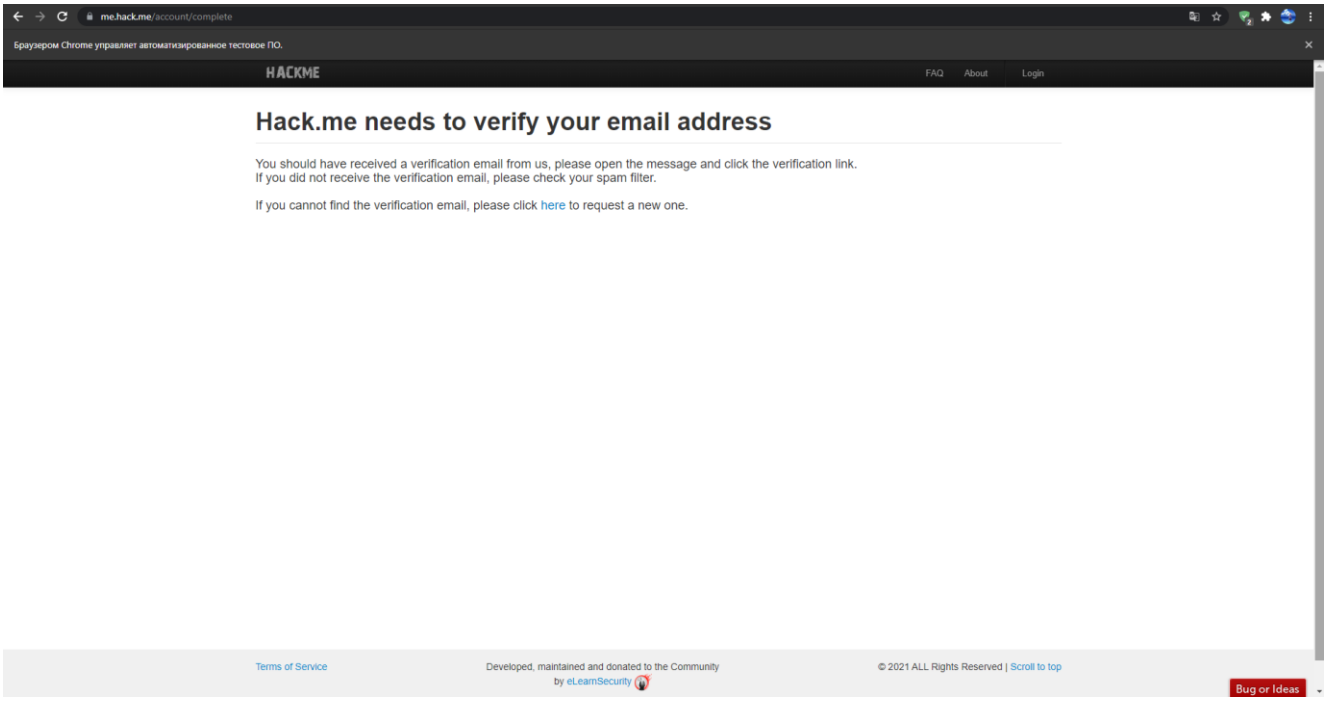


Рисунок 3.6 – Результат роботи класу TestRegistration у веб-додатку

Як показано на рисунку 3.6 у результаті роботи тестового сценарію відбувся перехід на сторінку complete. Нижче на рисунку 3.7 наведено блок-схему виконання класу TestRegistration:

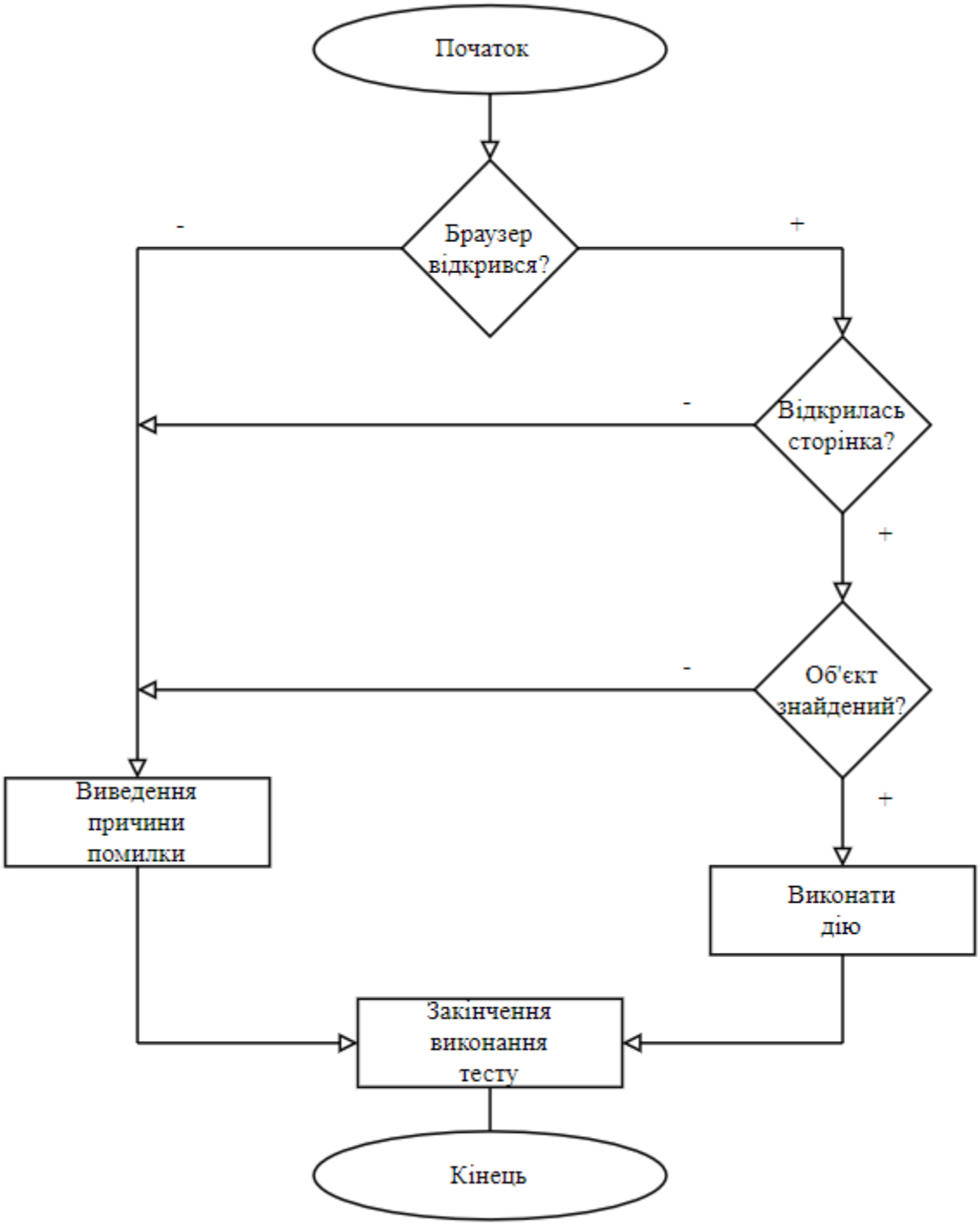


Рисунок 3.7 – Блок-схема роботи класу TestRegistration

Наступним тестовим сценарієм після успішної реєстрації відбувається авторизація облікового запису. Де першим кроком відбувається перехід на сторінку реєтрації. Після цього у поле Username та Password вводиться потрібна інформація, і натискається кнопка Login. Результатом виконання тестового

сценарію TestAuthorization стане успішний вхід користувача у свій профіль(рисунок 3.8).

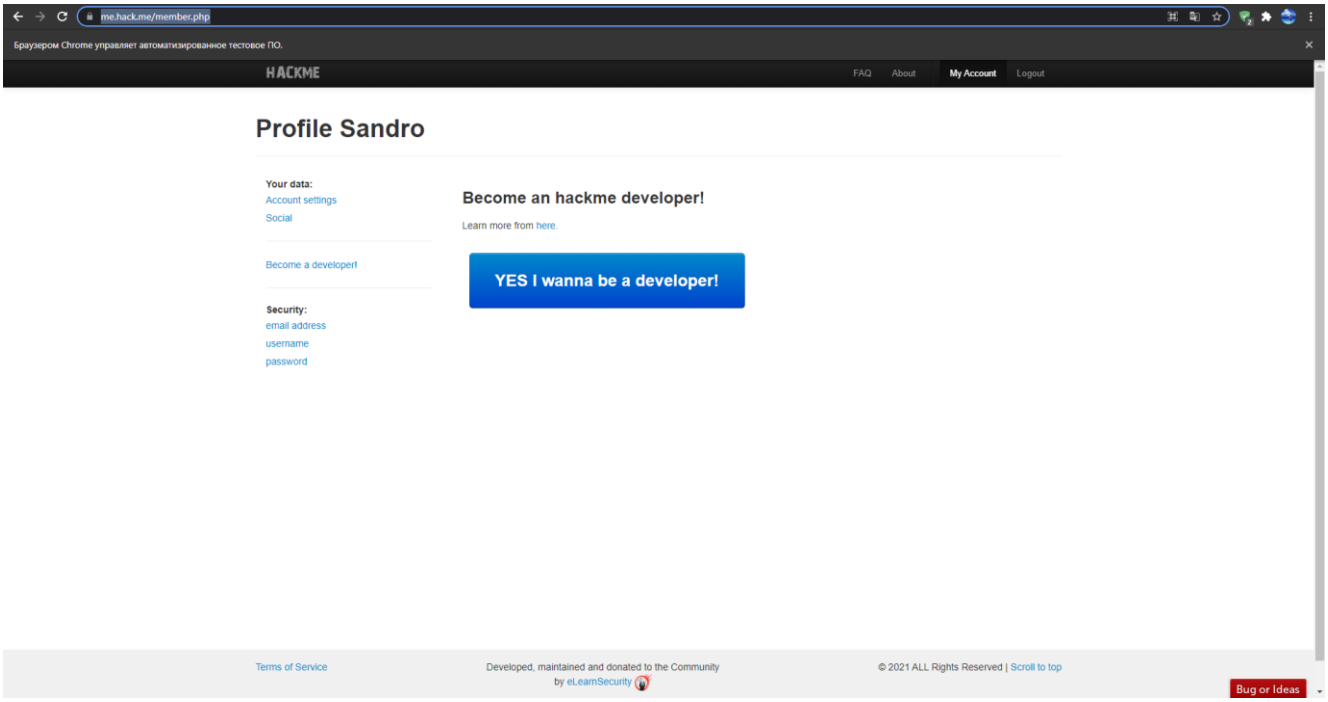


Рисунок 3.8 – Результат роботи класу TestAuthorization у веб-додатку

У свою чергу у консолі буде вказана наступна інформація(рисунок 3.9).

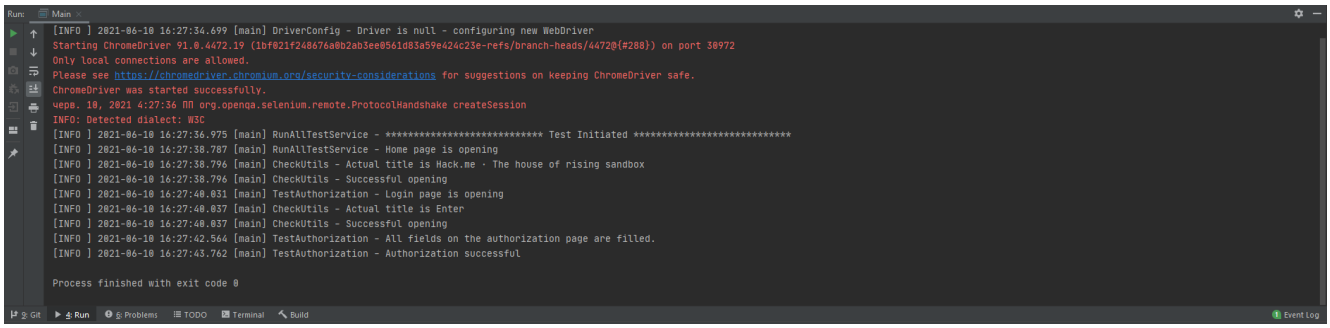


Рисунок 3.9 – Результат роботи класу TestAuthorization

На черзі клас TestAccountSettings завдання якого на сторінці AccountSettings змінити дані у текстових полях First name, Last name і Organization, після чого чого зберегти зміни. Саме у цьому класі був залучений Robot Framework. Його завдання у цьому тестовому сценарію за допомогою гаряших клавіш клавіатури

максимально швидко провести очистку текстових полів. Після запуску тестового сценарію у консолі буде відображена наступна інформація:

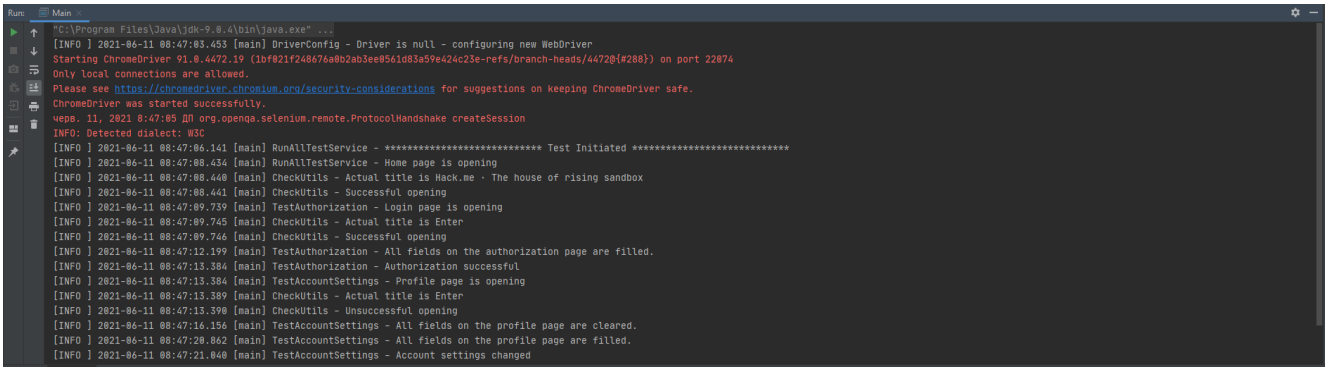


Рисунок 3.10 – Результат роботи класу TestAccountSettings

Як видно на рисунку 3.10 у консолі після успішного виконання класу TestAuthorization одразу відбулось виконання TestAccountSettings. У ході якого було перевірено чи відбулось успішне відкриття сторінки, після чого була зроблена перевірка чи всі текстові поля очищені, а також чи коректно було введено нову інформацію. Кінцева перевірка чи була збережена нова інформація в обліковому записі(рисунок 3.11).

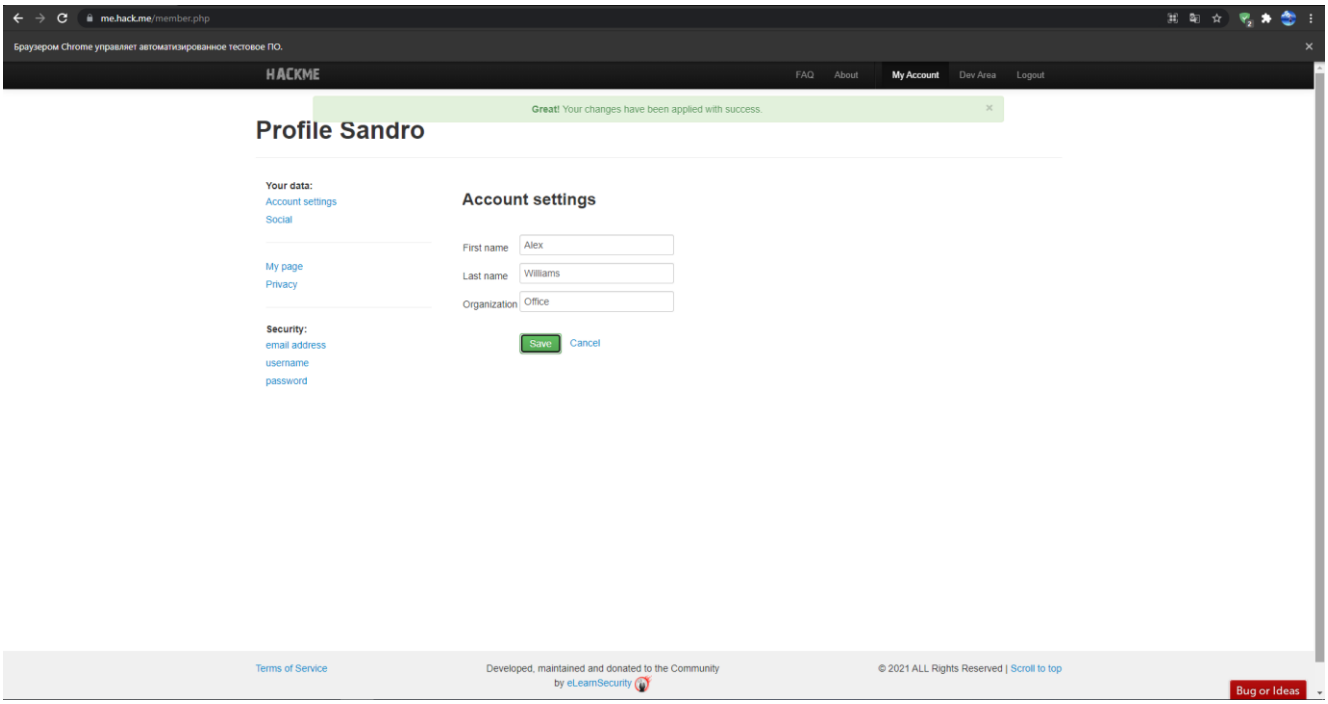


Рисунок 3.11 – Результат роботи класу TestAccountSettings у веб-додатку

Останній етап який відтворить інформаційна система це буде зміна рівня доступа до деякої інформації користувача на сторінці Manage your privacy. А саме зміна рівня для імені та прізвища з приватного на загальний. Крім цього рівень гіперпосилань на Twitter та Facebook буде змінений на приватний(рисунок 3.12).

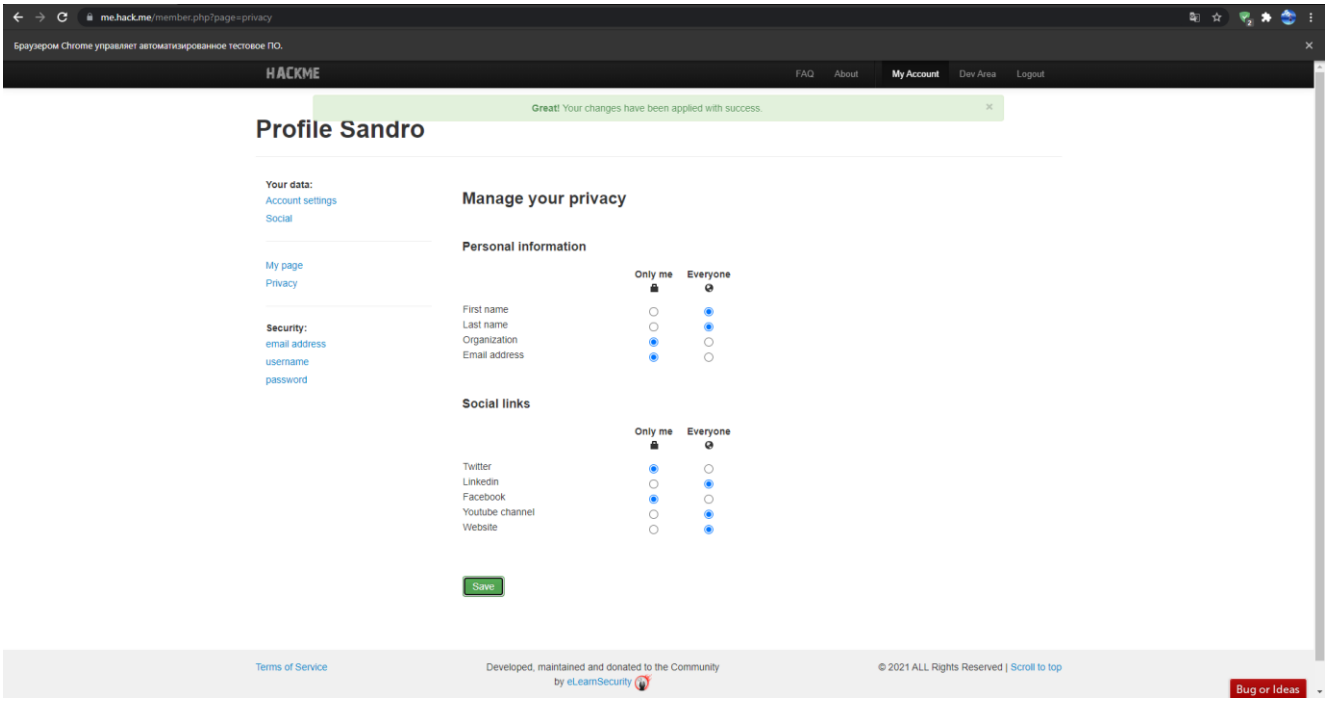


Рисунок 3.12 – Результат роботи класу TestManagePrivacy у веб-додатку

Успішна зміна рівней доступу на сторінці Manage your privacy завершує поетапне автоматизоване тестування веб-додатку Hack.me. З черги можна додавати чи видаляти тестові сценарії, а також змінювати порядок відтворення.

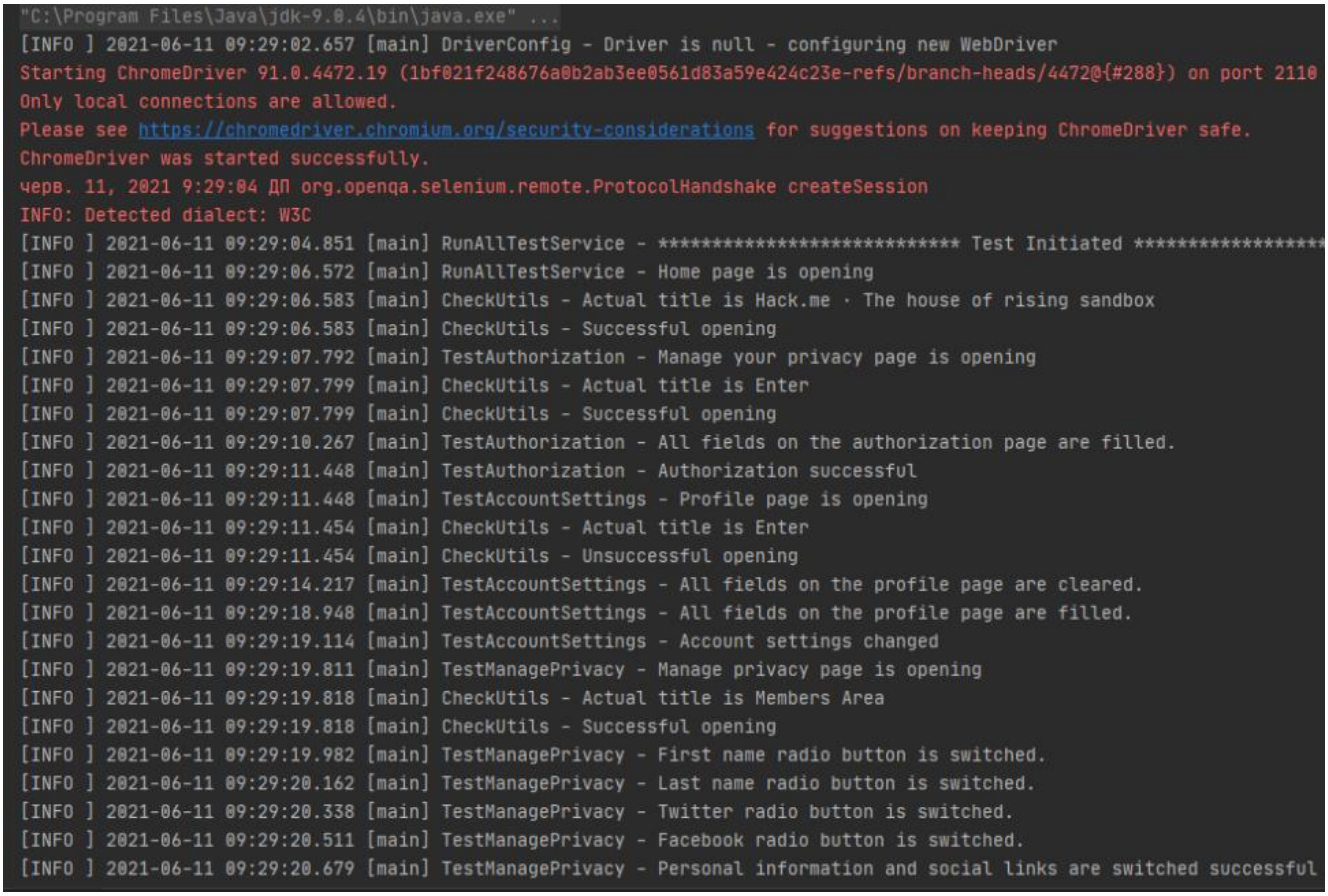


Рисунок 3.13 – Результат роботи виконання усіх тестових сценаріїв

Як видно на рисунку 3.13, в консолі в порядку виконання тестових сценаріїв з’явились інформація про успішні дії, що в свою чергу означає, що на момент проведення тестування обрані сторінки та функціонал на них пряцює коректно. На рисунку 3.14 наведена блок-схема роботи усіх тестових сценаріїв. Створена інформаційна система реалізована згідно найкращих практик і з застосуванням відомих шаблонів проектування, тому її функціонал є можливість легко і зручно можна розширити.

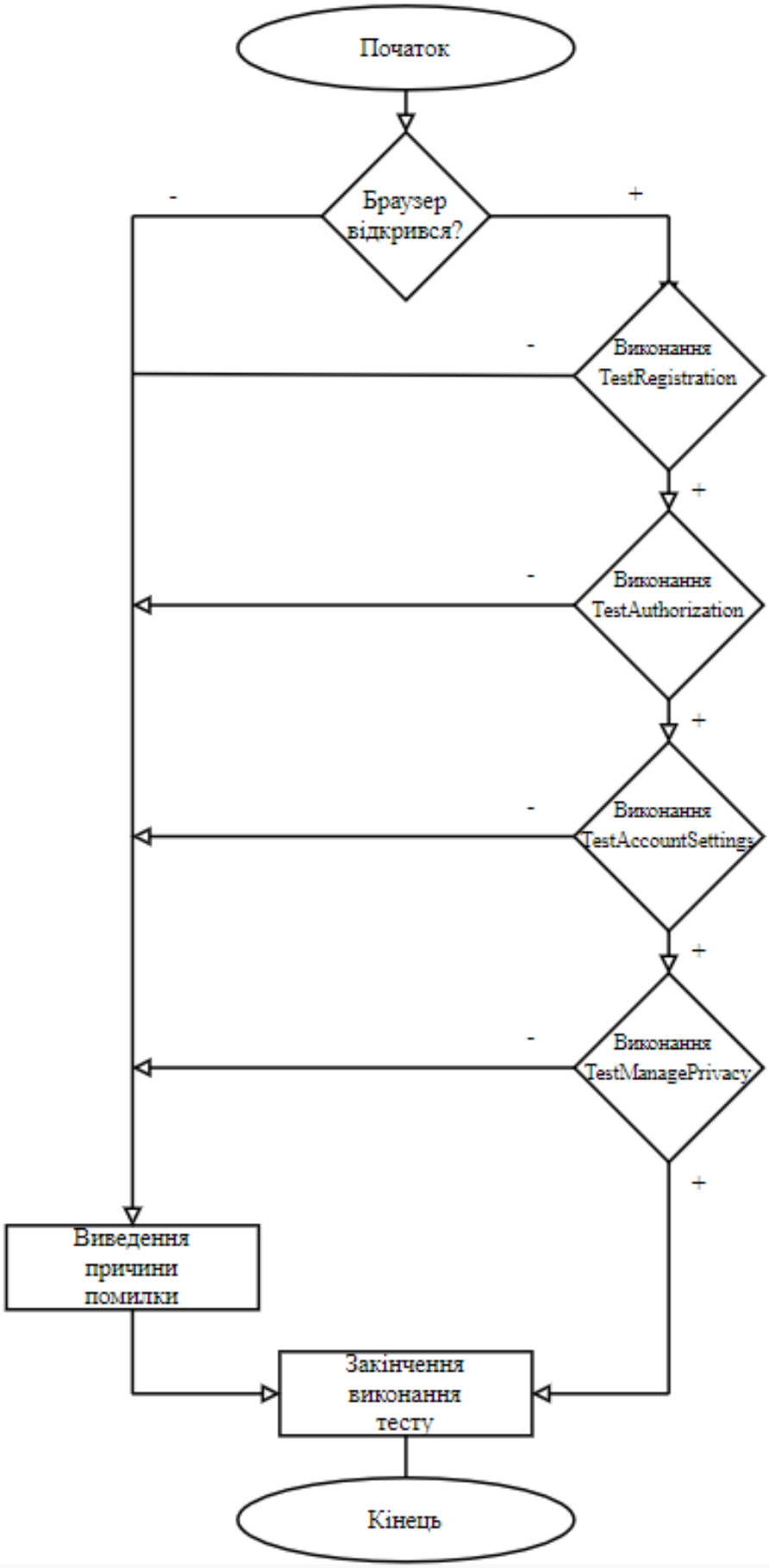


Рисунок 3.14 – Блок-схема роботи усіх тестових сценаріїв

3.4 Висновки

В даному розділі було проведено процес розробки інформаційної системи, що реалізує автоматизоване тестування веб-додатку, а також контроль якості на ньому. Було розглянуто додаткові шляхи покращення системи

Був доданий журнал помилок який надає велику кількість інформації при виконанні тестових сценаріїв. Також була описана реалізація основних компонентів, що були спроектовані в другому розділі, особливості їх роботи, та наведені приклади коду.

Розробка відбувалась в середовищі IntelliJ IDEA, в процесі використовувались наступні технології:

1. Java 11;
2. Selenium 3.143.59;
3. Log4j;
4. Maven.

ВИСНОВКИ

Під час практичних та теоретичних досліджень було визначено, що надійність веб-додатків є необхідністю в часи розвитку інтернет технологій. Також було приділено увагу тому що слід приділяти достатню кількість часу тестуванню надійності та більш швидкому пошуку помилок що виникають при розробці веб-додатків.

В першому розділі був зроблений аналіз основних видів веб-додатків, їх призначення та класифікацію. Було виявлені вразливі місця кожно з них відносно класифікації їх побудови. Також було розглянуто етапи розробки веб-додатків, і роль процесу тестування у них.

В другому розділі було було розглянуто популярні мови програмування для створення автоматизованих тестів на сьогоднішній день. Також було зроблено аналіз наявних інструментів та фреймворків для тестування веб-додатків. Під кінець був зроблений аналіз взаємодії різних мов та інструментів та обрано найкращу комбінацію.

В третьому розділі було розроблено інформаційну систему, що реалізує автоматизоване тестуванння веб-додатку, а також контроль якості на ньому. В процесі розробки були використані інструменти на фреймворки визначені в другому розділі, а також застосовані популярні шаблони проектування.

Розроблена система на мові Java та з використанням інструмента Selenium, була протестована у найновішій версії браузера Chrome. Також був створений окремий пакет який дозволить проводити тестові сценарії і у інших браузерах тим самим здійснюючи крос-браузерне тестування. Окрім цього, були реалізовані можливості, для швидкої зміни порядку відтворення тестів, а також додавання чи видалення з черги. Важливою є реалізація журнала логування log4j який надає додаткову інформацію на консоль для зневадження тестів і просто з метою стеження за їх виконанням.

В результаті роботи було глибше досліджено особливості автоматизованого тестування при розробці веб-додатку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. The TIOBE Programming Community index is an indicator of the popularity of programming languages. URL: https://www.tiobe.com/tiobe-index/
2. Selenium documentation. URL: https://www.selenium.dev/documentation/en/
3. Stresnjak S., Hocenski Z. Usage of Robot Framework in Automation of Functional Test Regression. 2011. 31 с.
4. Dale H. Emery. Writing Maintainable Automated Acceptance Tests. 2009. 5 с.
5. Hendrickson E. Driving Development with Tests: ATDD and TDD. 2008. 8 с.
6. Crispin L. An Introduction to Test Automation Design. 2010. 2 с.
7. Schild H. Java: The Complete Reference, Tenth Edition. 2017. 1200 с.
8. Components of the Selenium Automation Tool. URL: https://dzone.com/articles/components-of-selenium-automation-tool
9. Colantonio J. The UFT API Testing Manifesto: A step-by-step, hands-on testing guide for the masses 2nd Edition. 2014. 226 с.
10. Selenium IDE Market Overview. URL: https://ui.vision/blog/selenium-ide-2018/
11. What is Selenium. URL: https://sparkdatabox.com/blog/selenium/
12. Java 10 Released, First in the New Faster Cadence. URL: https://adtmag.com/articles/2018/03/21/java-10.aspx
13. 6 Different Types of Web Application Development URL: https://www.clustox.com/6-different-types-of-web-application-development
14. Automation Testing Life Cycle URL: https://www.lambdatest.com/blog/all-you-have-to-know-about-automationtesting-life-cycle
15. List of Testing Tools. URL: http://www.guru99.com/list-of-testing-tools.html
16. Naftalin M., Wadler P. Java Generics and Collections. 2006. 294 с.

17. Loy M., Niemeyer P., Leuck D. Learning Java, 4th Edition. 2020. 520 c.

18. Apache Log4j 2. URL: https://logging.apache.org/log4j/2.x/index.html

19. Jeffrey L., Lonnie D., Bentley, Kevin C., Dittman. Systems Analysis and Design Methods. 6th edition. 2003. 22 c.

20. Kolawa A., Huizinga D. Automated Defect Prevention: Best Practices in Software Management. 2007. 54 c.

21. Kenefick S. Produce Better Software by Using a Layered Testing Strategy. 2014. 9 c.

22. Wayne A. DevOps: Are You Pushing Bugs to Your Clients Faster?. 2015. 3 c.

23. Hinz J. Fifth Generation Scriptless and Advanced Test Automation Technologies. 2007. 9 c.

24. Blokdyk G. Software QA Complete Self-Assessment Guide. 2019. 43 c.

25. Filipova O., Vilão Rui Software Development From A to Z. 2018. 67 c.

26. Maksymenko D. QA Engineer I Make Developers Cry. 2019. 22 c.

27. Babu Munta J. Software Quality and Java Automation Engineer Survival Guide. 2016.

28. Iancu L. QA Quality Assurance & Software Testing Fundamentals. 2019.

29. Hung Q. Nguyen. Testing Applications on the Web. 2000.

30. Sliger M., Broderick S. Software Project Manager's Bridge to Agility, The. 2008.

31. Savenkov R. How to Become a Software Tester. 2008.

32. Antonimuthu R., Rajamanickam A. Software Testing and QTP Automation. 2013.

33. Linz T. Testing in Scrum: A Guide for Software Quality Assurance in the Agile World. 2014.

34. Dustin E. Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality 1st Edition. 2009.

35. Pedron L. Software Test Automation: Getting Started Guide for QA Managers, Quality Engineers and Project Managers. 2015.

36. Naik K. Software Testing and Quality Assurance: Theory and Practice. 2008.

37. Suryn W. Software Quality Engineering: A Practitioner's Approach 1st Edition. 2014.

38. Khare S. A Guide to Software Testing Lifecycle (STLC): Software Testing Lifecycle is a standard procedure divided into different phases, followed by the QA Team to complete all testing activities. 2019.

39. Gao J. Testing and Quality Assurance for Component-Based Software. 2003.

40. Link J. Unit Testing in Java: How Tests Drive the Code (The Morgan Kaufmann Series in Software Engineering and Programming). 2003.

41. Becker P. Facts and Fallacies of Software Engineering 1st Edition. 2002.

42. Horstmann Cay S. Core Java Volume I-Fundamentals (10th Edition). 2016. 88 с.

43. Patton R. Software Testing, 2nd Edition. 2005.

44. C. Jorgensen P. Software Testing: A Craftsman’s Approach, Fourth Edition. 1995.

45. Whittaker J. How to Break Software: A Practical Guide to Testing.

Додаток А

(обов’язковий)

Лістинг програмного коду

Файл AppConfig.java

```
package org.example.config;

public class AppConfig {

    public static final String USER_FIRST_NAME = "Sandro";

    public static final String USER_lAST_NAME = "Botticelli";

    public static final String USER_ORGANIZATION = "University";

    public static final String NEW_USER_FIRST_NAME = "Alex";

    public static final String NEW_USER_lAST_NAME = "Williams";

    public static final String NEW_USER_ORGANIZATION = "Office";

    public static final String USER_EMAIL = "bottichelli1445@gmail.com";

    public static final String USER_LOGIN = "Sandro";

    public static final String USER_PASSWORD = "Password1!";

    public static final String PATH_TO_CHROME_PROFILE = "C:\\Users\\Lenovo
L340\\AppData\\Local\\Google\\Chrome\\UserDataCopy";

    public static final String PATH_TO_CHROMEDRIVER = "D:\\IDEA
projects\\chromedriver_91\\chromedriver.exe";

    public static final String USE_DRIVER = "webdriver.chrome.driver";

    public static final String HOME_PAGE_URL = "https://hack.me/";

    public static final String REGISTRATION_PAGE_URL =
"https://me.hack.me/signup";

    public static final String LOGIN_PAGE_URL = "https://me.hack.me/login";

    public static final String HACKME_REGIST_TITLE = "Sign Up for Hack.me";

    public static final String HACKME_MAIN_TITLE = "Hack.me · The house of rising
sandbox";

    public static final String HACKME_LOGIN_TITLE = "Enter";

    public static final String HACKME_PROFILE_TITLE = "Members Area";

    public static final String START = "777";

}
```

Файл DriverConfig.java

```
package org.example.config; import org.openqa.selenium.WebDriver;
```

```
import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.chrome.ChromeOptions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.concurrent.TimeUnit;

import static org.example.config.AppConfig.*;

public class DriverConfig {

    private static final Logger LOGGER =
LoggerFactory.getLogger(DriverConfig.class);

    private static WebDriver driver;

    public static WebDriver getDriver() {

        if (driver == null) {

            LOGGER.info("Driver is null - configuring new WebDriver");

            ChromeOptions options = new ChromeOptions();

            options.addArguments("user-data-dir=" + PATH_TO_CHROME_PROFILE);

            //PATH TO CHROMEDRIVER

            System.setProperty(USE_DRIVER, PATH_TO_CHROMEDRIVER);

            driver = new ChromeDriver(options);

            driver.manage().window().maximize();

            driver.manage().timeouts().pageLoadTimeout(5, TimeUnit.SECONDS);

            driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);

        }

        return driver;

    }

}
```

Клас CheckUtils.java

```
package org.example.util;

import org.openqa.selenium.WebDriver;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class CheckUtils {

    private static final Logger LOGGER =
LoggerFactory.getLogger(CheckUtils.class);

    public static void openWebsite(WebDriver driver, String openURL) {
driver.get(openURL);

    }
```

```
    public static void verifyTitle(WebDriver driver, String expectedTitle) {
        String actualTitle = driver.getTitle();
        LOGGER.info("Actual title is {}", actualTitle);
        if (expectedTitle.equals(actualTitle)) {
            LOGGER.info("Successful opening");
        } else {
            LOGGER.info("Unsuccessful opening");
        }
    }
}
```

Файл Main.java

```
package org.example;

import org.example.config.DriverConfig;

import org.example.service.*;

import org.openqa.selenium.WebDriver;

import static org.example.config.AppConfig.START;

public class Main {

    private static final WebDriver driver = DriverConfig.getDriver();

    public static void main(String[] args) {

        RunAllTestService workflowService = new RunAllTestService(driver, new
TestAuthorization(driver), new TestAccountSettings(driver), new
TestManagePrivacy(driver));

        workflowService.run(START);

    }

}
```

Файл RunAllTestService.java

```
package org.example.service;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import static org.example.config.AppConfig.*;

import static org.example.util.CheckUtils.openWebsite;

import static org.example.util.CheckUtils.verifyTitle;
```

```
public class RunAllTestService {

    private final WebDriver driver;

    private static final Logger LOGGER =
LoggerFactory.getLogger(RunAllTestService.class);

    private final TestAuthorization testAuthorization;

    private final TestAccountSettings testAccountSettings;

    private final TestManagePrivacy testManagePrivacy;

    private final Actions builder;

    private final WebDriverWait wait;

    public RunAllTestService(WebDriver driver, TestAuthorization
testAuthorization, TestAccountSettings testAccountSettings, TestManagePrivacy
testManagePrivacy) {

        this.driver = driver;

        this.testAuthorization = testAuthorization;

        this.testAccountSettings = testAccountSettings;

        this.testManagePrivacy = testManagePrivacy;

        this.builder = new Actions(driver);

        this.wait = new WebDriverWait(driver, 10);

    }

    public void run(String START) {

        LOGGER.info("**************************** Test Initiated
****************************");

        try {

            openWebsite(driver, HOME_PAGE_URL);

        } catch (Exception e) {

            LOGGER.error(e.getMessage());

            throw new RuntimeException("Home page didn't opened.");

        }

        LOGGER.info("Home page is opening");

        verifyTitle(driver, HACKME_MAIN_TITLE);

        testAuthorization.testauth();

        testAccountSettings.accset();

        testManagePrivacy.testmanage();

}}
```

Клас TestRegistration.java

```
package org.example.service;
```

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.time.Duration;

import static org.example.config.AppConfig.*;

import static org.example.util.CheckUtils.openWebsite;

import static org.example.util.CheckUtils.verifyTitle;

public class TestRegistration {

    private static final Logger LOGGER =
LoggerFactory.getLogger(TestRegistration.class);

    private final WebDriver driver;

    private final Actions builder;

    public TestRegistration(WebDriver driver) {

        this.driver = driver;

        this.builder = new Actions(driver);

    }

    public void regist() {

        try {

            WebElement dropDownListSingIn =
driver.findElement(By.xpath("/html/body/div[1]/div/div[2]/ul/li[4]/a/i"));

            builder.moveToElement(dropDownListSingIn).sendKeys(" Sign up
").doubleClick().perform();

        } catch (Exception e) {

            LOGGER.error(e.getMessage());

            throw new RuntimeException("Sign in dropDownList is not found");

        } builder.pause(Duration.ofSeconds(3));

        try {

            openWebsite(driver, REGISTRATION_PAGE_URL);

        } catch (Exception e) {

            LOGGER.error(e.getMessage());

            throw new RuntimeException("Registration page didn't opened.");

        }

        LOGGER.info("Registration page is opening");
```

```
        verifyTitle(driver, HACKME_REGIST_TITLE);
        builder.pause(Duration.ofSeconds(1));
        enterFirstName();
        builder.pause(Duration.ofSeconds(1));
        enterLastName();
        builder.pause(Duration.ofSeconds(1));
        enterOrganization();
        builder.pause(Duration.ofSeconds(1));
        enterEmail();
        builder.pause(Duration.ofSeconds(1));
        enterConfirmEmail();
        builder.pause(Duration.ofSeconds(1));
        enterUsername();
        builder.pause(Duration.ofSeconds(1));
        enterPassword();
        builder.pause(Duration.ofSeconds(1));
        enterConfirmPassword();
        builder.pause(Duration.ofSeconds(1));
        LOGGER.info("All fields on the registration page are filled.");
        builder.pause(Duration.ofSeconds(2));
        try {
            WebElement proceedButton = driver.findElement(By.id("btnproceed"));
            builder.moveToElement(proceedButton).click().perform();
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Proceed button is not found");        }
        LOGGER.info("Registration successful");
    }
    private void enterFirstName() {
        try {
            WebElement firstNameField = driver.findElement(By.id("name"));
            builder.moveToElement(firstNameField).build().perform();
            driver.findElement(By.id("name")).sendKeys(USER_FIRST_NAME);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
```

```
            throw new RuntimeException("First name field is not found");
        }
    }
    private void enterLastName() {
        try {
            WebElement lastNameField = driver.findElement(By.id("lastname"));
            builder.moveToElement(lastNameField).build().perform();
            driver.findElement(By.id("lastname")).sendKeys(USER_lAST_NAME);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Last name field is not found");
        }
    }
    private void enterOrganization() {
        try {
            WebElement lastNameField = driver.findElement(By.id("organization"));
            builder.moveToElement(lastNameField).build().perform();
            driver.findElement(By.id("organization")).sendKeys(USER_ORGANIZATION);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Organization field is not found");
        }
    }
private void enterEmail() {
        try {
            WebElement emailField = driver.findElement(By.id("email"));
            builder.moveToElement(emailField).build().perform();
            driver.findElement(By.id("email")).sendKeys(USER_EMAIL);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Email field is not found");
        }
    }
    private void enterConfirmEmail() {
        try {
```

```
            WebElement emailConfirmField = driver.findElement(By.id("remail"));
            builder.moveToElement(emailConfirmField).build().perform();
            driver.findElement(By.id("remail")).sendKeys(USER_EMAIL);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Confirm email address field is not
found");
        }
    }
    private void enterUsername() {
        try {
            WebElement enterUsername = driver.findElement(By.id("nickname"));
            builder.moveToElement(enterUsername).build().perform();
            driver.findElement(By.id("nickname")).sendKeys(USER_LOGIN);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Username field is not found");
        }
    }
    private void enterPassword() {
        try {
WebElement passwordField = driver.findElement(By.id("password"));
            builder.moveToElement(passwordField).build().perform();
            driver.findElement(By.id("password")).sendKeys(USER_PASSWORD);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Password field is not found");
        }
    }
    private void enterConfirmPassword() {
        try {
            WebElement confirmPasswordField =
driver.findElement(By.id("repassword"));
            builder.moveToElement(confirmPasswordField).build().perform();
            driver.findElement(By.id("repassword")).sendKeys(USER_PASSWORD);
        } catch (Exception e) {
```

```
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Password field is not found");
        }
    }
}
```

Файл TestAuthorization.java

```
package org.example.service;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.interactions.Actions;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import java.time.Duration;
import static org.example.config.AppConfig.*;
import static org.example.config.AppConfig.HACKME_REGIST_TITLE;
import static org.example.util.CheckUtils.openWebsite;
import static org.example.util.CheckUtils.verifyTitle;
public class TestAuthorization {
    private static final Logger LOGGER =
LoggerFactory.getLogger(TestAuthorization.class);
    private final WebDriver driver;
    private final Actions builder;
    public TestAuthorization(WebDriver driver) {
        this.driver = driver;
        this.builder = new Actions(driver);
    }
    public void testauth() {
        openWebsite(driver, LOGIN_PAGE_URL);
        LOGGER.info("Manage your privacy page is opening");
        verifyTitle(driver, HACKME_LOGIN_TITLE);
        builder.pause(Duration.ofSeconds(1));
        enterUsername();
        builder.pause(Duration.ofSeconds(1));
        enterPassword();
```

```
        builder.pause(Duration.ofSeconds(1));
        LOGGER.info("All fields on the authorization page are filled.");
        try {
            WebElement loginButton = driver.findElement(By.id("btnlogin"));
            builder.moveToElement(loginButton).click().perform();
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Login button is not found");
        }
        LOGGER.info("Authorization successful");
    }
    private void enterUsername() {
        try {
            WebElement enterUsername = driver.findElement(By.id("username"));
            builder.moveToElement(enterUsername).build().perform();
            driver.findElement(By.id("username")).sendKeys(USER_LOGIN);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Username field is not found");
        }
    }
    private void enterPassword() {
        try {
            WebElement passwordField = driver.findElement(By.id("password"));
            builder.moveToElement(passwordField).build().perform();
            driver.findElement(By.id("password")).sendKeys(USER_PASSWORD);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Password field is not found");
        }
    }
}
```

Файл TestAccountSettings.java

```
package org.example.service;
```

```
import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.time.Duration;

import static org.example.config.AppConfig.*;

import static org.example.util.CheckUtils.openWebsite;

import static org.example.util.CheckUtils.verifyTitle;

import java.awt.*;

import java.awt.datatransfer.Clipboard;

import java.awt.datatransfer.StringSelection;

import java.awt.event.KeyEvent;

public class TestAccountSettings {

    private static final Logger LOGGER =
LoggerFactory.getLogger(TestAccountSettings.class);

    private final WebDriver driver;

private final Actions builder;

    private final Robot robot;

    public TestAccountSettings(WebDriver driver) {

        this.driver = driver;

        this.builder = new Actions(driver);

        try {

            this.robot = new Robot();

        } catch (AWTException e) {

            LOGGER.error(e.getMessage());

            throw new RuntimeException("Robot framework is not working.");

        }

    }

    public void accset() {

        LOGGER.info("Profile page is opening");

        verifyTitle(driver, HACKME_PROFILE_TITLE);

        builder.pause(Duration.ofSeconds(1));

        try {
```

```
            WebElement firstNameField = driver.findElement(By.id("firstname"));
            builder.moveToElement(firstNameField).clickAndHold().perform();
            builder.pause(Duration.ofSeconds(1));
            robot.keyPress(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_DELETE);
            robot.keyPress(KeyEvent.VK_TAB);
            robot.keyPress(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_DELETE);
            robot.keyPress(KeyEvent.VK_TAB);
            robot.keyPress(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_A);
robot.keyRelease(KeyEvent.VK_A);
            robot.keyRelease(KeyEvent.VK_CONTROL);
            robot.keyPress(KeyEvent.VK_DELETE);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Clearing text fields failed.");
        }
        LOGGER.info("All fields on the profile page are cleared.");
        builder.pause(Duration.ofSeconds(1));
        enterFirstName();
        builder.pause(Duration.ofSeconds(1));
        enterLastName();
        builder.pause(Duration.ofSeconds(1));
        enterOrganization();
        LOGGER.info("All fields on the profile page are filled.");
        try {
            WebElement saveButton = driver.findElement(By.id("btninfo"));
            builder.moveToElement(saveButton).click().perform();
```

```
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Save button is not found");
        }
        LOGGER.info("Account settings changed");
    }
    private void enterFirstName() {
        try {
            WebElement firstNameField = driver.findElement(By.id("firstname"));
            builder.moveToElement(firstNameField).build().perform();
            driver.findElement(By.id("firstname")).sendKeys(NEW_USER_FIRST_NAME);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("First name field is not found");
        }
    }
    private void enterLastName() {
        try {
            WebElement lastNameField = driver.findElement(By.id("lastname"));
            builder.moveToElement(lastNameField).build().perform();
            driver.findElement(By.id("lastname")).sendKeys(NEW_USER_lAST_NAME);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Last name field is not found");
        }
    }
    private void enterOrganization() {
        try {
            WebElement lastNameField = driver.findElement(By.id("organization"));
            builder.moveToElement(lastNameField).build().perform();
driver.findElement(By.id("organization")).sendKeys(NEW_USER_ORGANIZATION);
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Organization field is not found");
```

```
        }
    }
}
```

Файл TestManagePrivacy.java

```
package org.example.service;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.interactions.Actions;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.time.Duration;

public class TestManagePrivacy {

    private static final Logger LOGGER =
LoggerFactory.getLogger(TestManagePrivacy.class);

    private final WebDriver driver;

    private final Actions builder;

    public TestManagePrivacy(WebDriver driver) {

        this.driver = driver;

        this.builder = new Actions(driver);

    }

    public void testmanage() {

        try {

            WebElement privacyLink =
driver.findElement(By.xpath("//*[@id=\"wrap\"]/div/div[2]/div[1]/div/ul/li[6]/a"))
;

            builder.moveToElement(privacyLink).click().perform();

        } catch (Exception e) {

            LOGGER.error(e.getMessage());

            throw new RuntimeException("Privacy link is not found");

        }

        LOGGER.info("Manage privacy page is opening");

        verifyTitle(driver, HACKME_PROFILE_TITLE);

        try {

            WebElement firstNameRadioButton =
driver.findElement(By.xpath("//*[@id=\"personal\"]/tbody/tr[1]/td[3]/input"));
```

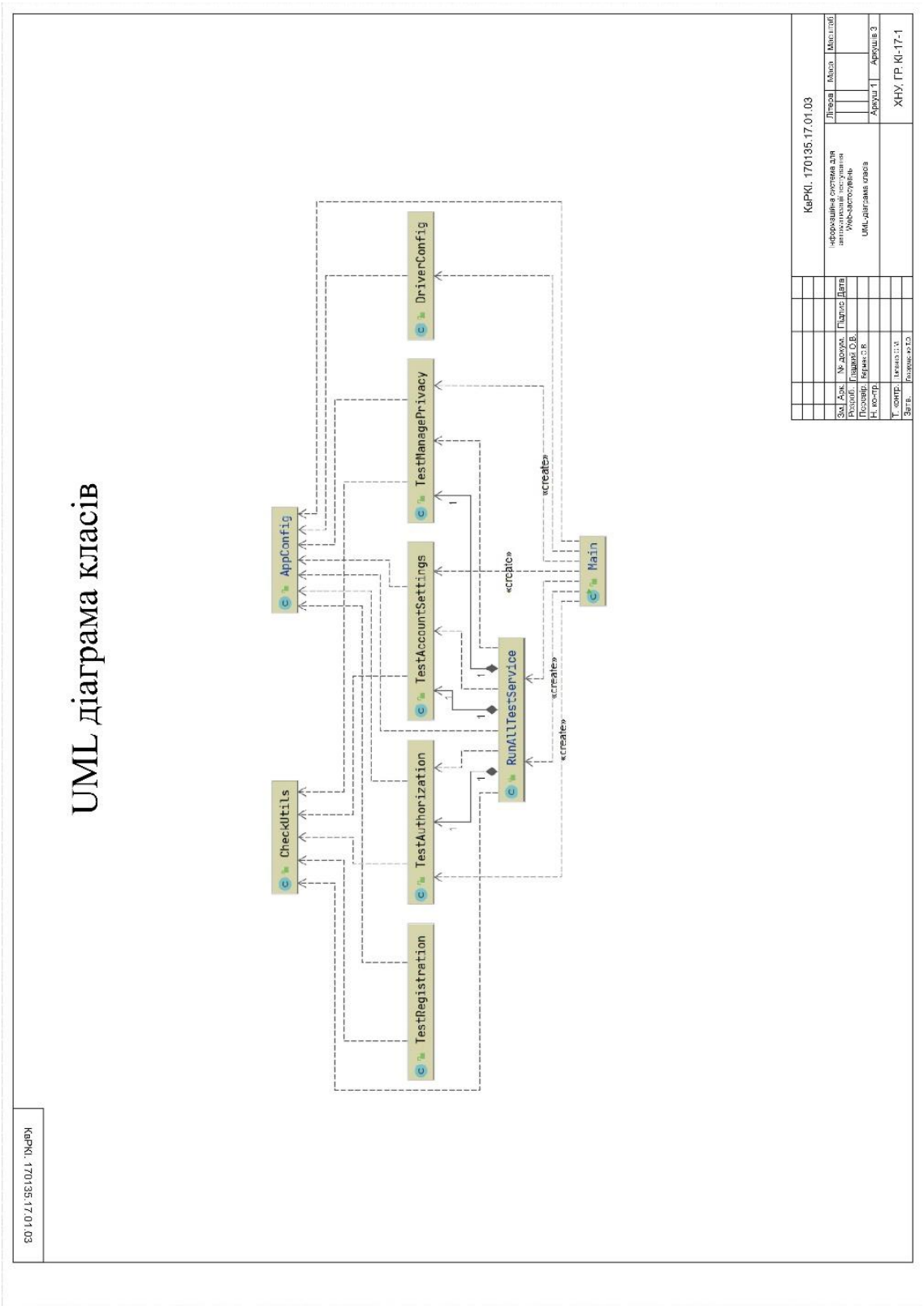
```
            builder.moveToElement(firstNameRadioButton).click().perform();
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("First name radio button is not found");
        }
        LOGGER.info("First name radio button is switched.");
        try {
            WebElement lastNameRadioButton =
driver.findElement(By.xpath("//*[@id=\"personal\"]/tbody/tr[2]/td[3]/input"));
            builder.moveToElement(lastNameRadioButton).click().perform();
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Last name radio button is not found");
        }
        LOGGER.info("Last name radio button is switched.");
        try {
            WebElement twitterRadioButton =
driver.findElement(By.xpath("//*[@id=\"social\"]/tbody/tr[1]/td[2]/input"));
            builder.moveToElement(twitterRadioButton).click().perform();
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Twitter radio button is not found");
        }
        LOGGER.info("Twitter radio button is switched.");
        try {
            WebElement facebookRadioButton =
driver.findElement(By.xpath("//*[@id=\"social\"]/tbody/tr[3]/td[2]/input"));
            builder.moveToElement(facebookRadioButton).click().perform();
        } catch (Exception e) {
            LOGGER.error(e.getMessage());
            throw new RuntimeException("Facebook radio button is not found");
        }
        LOGGER.info("Facebook radio button is switched.");
        try {
            WebElement saveButton =
driver.findElement(By.xpath("//*[@id=\"privacy\"]/button"));
            builder.moveToElement(saveButton).click().perform();
```

```
        } catch (Exception e) {

            LOGGER.error(e.getMessage());

            throw new RuntimeException("Save button is not found");

        }

        LOGGER.info("Personal information and social links are switched
successful");

    }

}
```

Додаток Б

(обов’язковий)

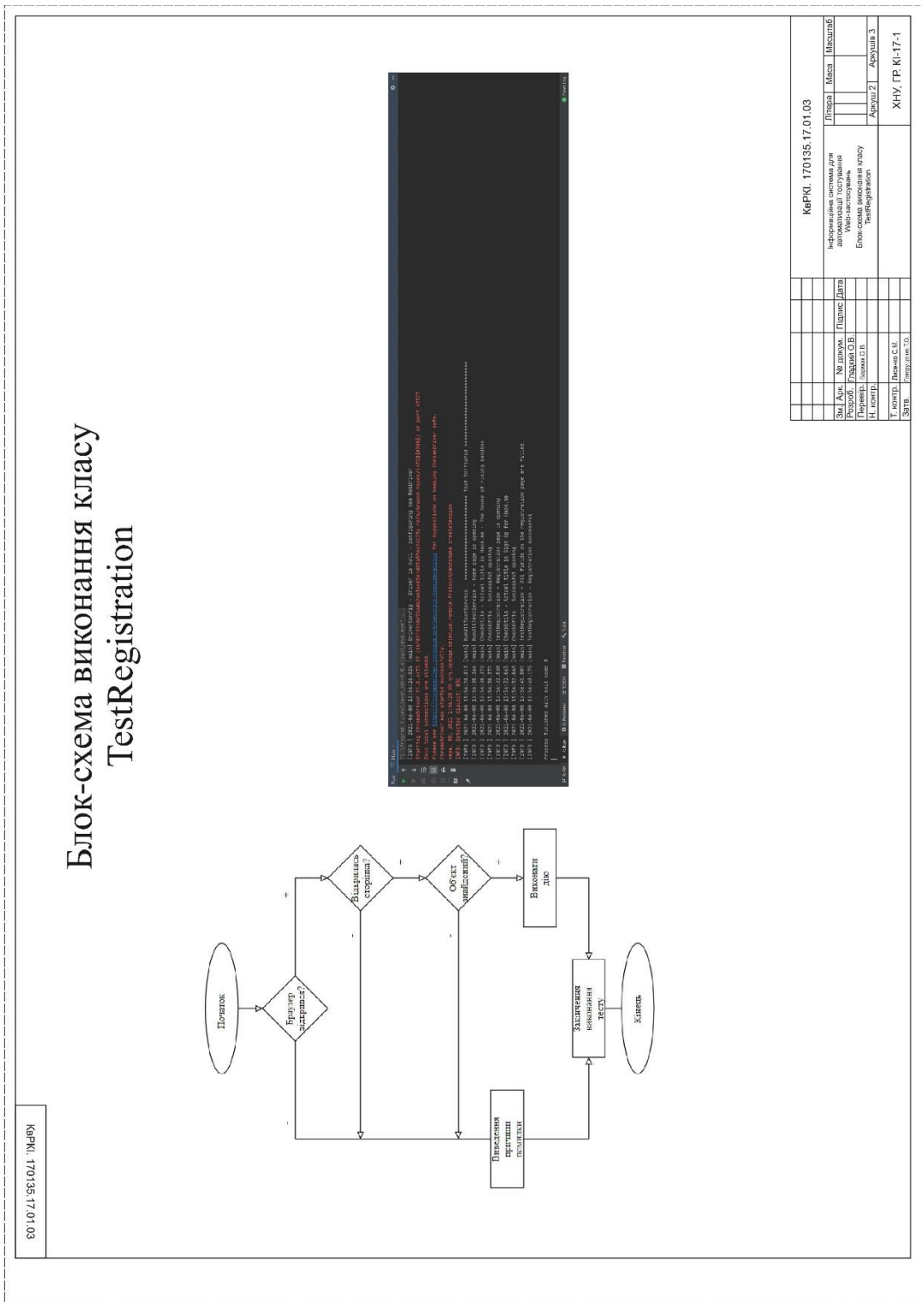
Копія креслення «UML діаграма класів»



Додаток В

(обов’язковий)

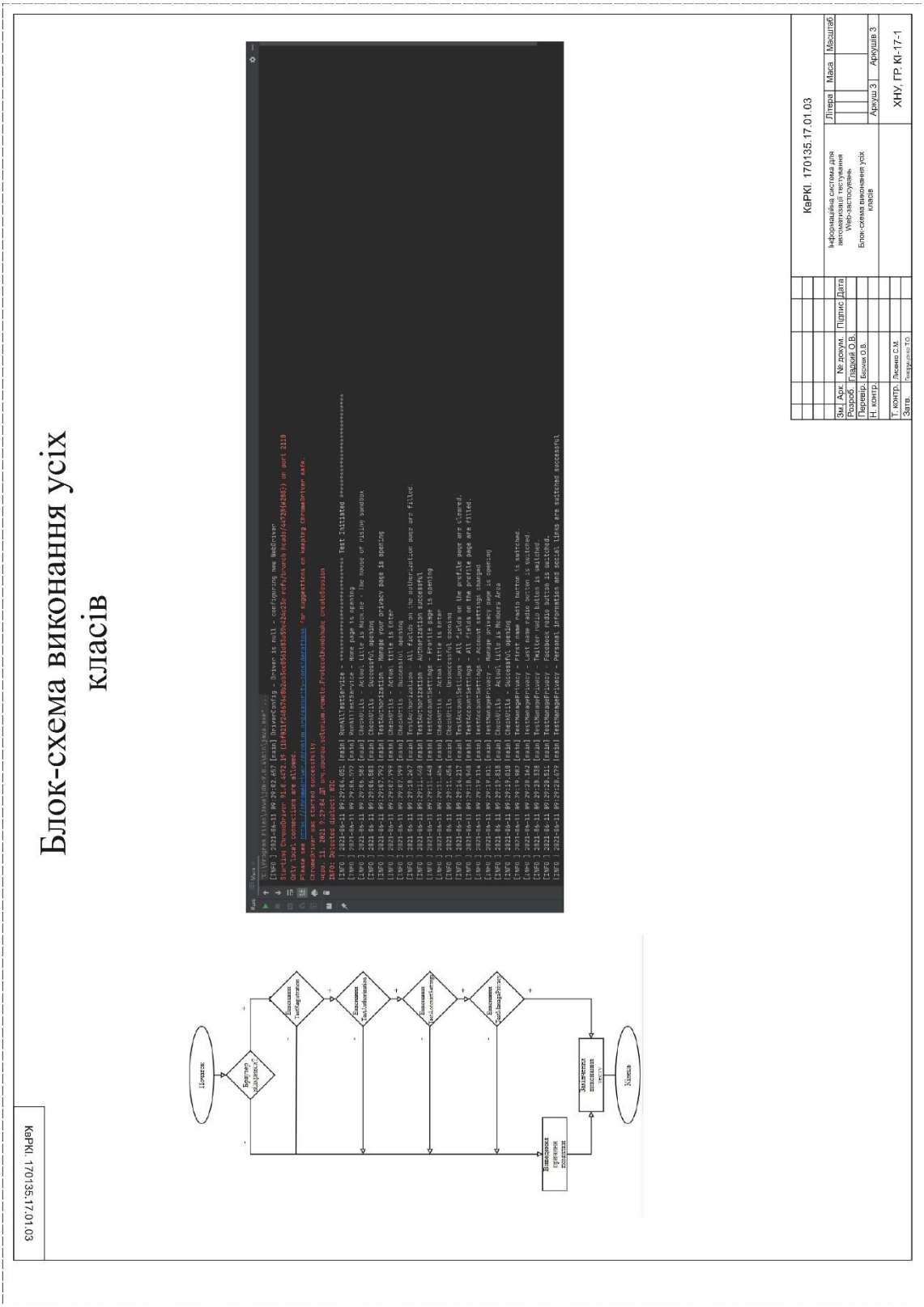
Копія креслення «Блок-схема виконання класу TestRegistration»



Додаток Г

(обов'язковий)

Копія креслення «Блок-схема виконання усіх класів»



UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1008302701

Дата перевірки:
15.06.2021 13:33:54 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.06.2021 13:37:43 EEST

ID користувача:
100005591

Назва документа: Гладкий_Інформаційна система для автоматизації тестування Web-застосувань

Кількість сторінок: 71 Кількість слів: 10484 Кількість символів: 78615 Розмір файлу: 3.55 MB ID файлу: 1008370804

2.94% Схожість

Найбільша схожість: 0.81% з джерелом з Бібліотеки (ID файлу: 1008248421)

1.76% Джерела з Інтернету 92 ..... Сторінка 73

1.28% Джерела з Бібліотеки 71 ..... Сторінка 73

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0% Вилучень

Немає вилучених джерел

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 9%**

	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
ID: 93989 Название: Інформаційна система для автоматизації тестування Web-застосувань Добавлено в БД: 2021-06-15 Авторы: Гладкий О.В. Руководители: Бармак О.В. Консультанты: Опоненты:	70780	658	962 (1%)	11 (2%)

Источник плагиата		Наличие плагиата в документе	
ID	Описание	Символы	Лексемы

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Гладкий Олександр В'ячеславович
Тема: Інформаційна система для автоматизації тестування Web-застосувань
Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:
Пояснювальна записка: 60 сторінок, 20 рисунків, 5 таблиць, 2 додатки, 23 джерела

1. Короткий зміст роботи та прийнятих рішень: метою роботи є побудова системи автоматизованого тестування програмного забезпечення веб-застосування шляхом поєднання у собі декількох способів тестування та налаштуванням безперервної інтеграції для проєкту.

2. Висновок про відповідність роботи завданню: робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи.

У першому розділі кваліфікаційної роботи проведено дослідження предметної області (аналіз типів веб-застосувань та методів їх розробки, аналіз ролі тестування при розробці програмного забезпечення та веб-застосувань, визначення вимог до систем автоматизації) та виконано постановку задачі дослідження.

У другому розділі кваліфікаційної роботи розглянуто інструменти та способи автоматизованого тестування веб-застосувань. Проведено порівняння інструментів та Java фреймворків для автоматизованого тестування та розглянуто взаємодію Java і Selenium при створенні автоматизованих тестів для веб-застосування.

У третьому розділі кваліфікаційної роботи виконано програмно-апаратну реалізацію та тестування програмно-технічного засобу. Реалізовані функціональні особливості та класи для тестових сценаріїв інформаційної системи Розроблено програмне забезпечення на базі запропонованих рішень та проведенне тестування системи.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: .в тексті записки використано застарілий термін «веб-додаток» - варто було б використати термін «веб-застосування».

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: не має.

9. Оцінка дипломної роботи: відміно

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Марчинюк Валерій Володимирович, зав. каф. АІТТС'ТК д.т.н., професор

"__" _____ 2021 р. _____ (підпис)

Завідувачу кафедри КІСП
д-ру техн.наук, проф. Говорущенко Т. О.

Гладкого О. В.
ПІБ здобувача вищої освіти

ФПКТС, 4 курсу, групи КІ-17-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність плагіату ознайомлений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

17.06.2021
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА СИСТЕМНОГО ПРОГРАМУВАННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Інформаційна система для автоматизації тестування Web-застосувань

Автор: Гладкий Олександр В'ячеславович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Бармак О. В., д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні;
3) до запозичень входять фрагменти програмного коду, що на мають авторства і містять поширені конструкції;
4) серед запозичень знаходяться загальновідомі терміни, скорочення та визначення.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 2.94 і адресується до 92 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О. В. Бармак

Гарант ОП _____ С. М. Лисенко

Завідувач кафедри КІСП _____ Т. О. Говорущенко