

ДОДАТОК А

Перелік джерел посилання за науковими напрямами керівника та науковців
кафедри програмної інженерії

2. Лановий О. Ф. Візуалізація в методах тестування програмного забезпечення: thesis. 2017. URL: <http://openarchive.nure.ua/handle/document/9432> (дата звернення: 02.03.2024).

ДОДАТОК Б

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



Ім'я користувача:
Кардаш Євген Вікторович каф.ПІ

ID перевірки:
1016363240

Дата перевірки:
15.06.2024 14:33:25 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
15.06.2024 14:39:19 EEST

ID користувача:
100013622

Назва документа: 2024_М_ПІ_ІПЗздр-22-1_Ларченко_С_О_скорочений

Кількість сторінок: 47 Кількість слів: 8941 Кількість символів: 68415 Розмір файлу: 1.17 MB ID файлу: 1016168603

2.45% Схожість

Найбільша схожість: 0.38% з джерелом з Бібліотеки (ID файлу: 1016147582)

1.48% Джерела з Інтернету

99

Сторінка 49

1.11% Джерела з Бібліотеки

70

Сторінка 49

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0% Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи

11

ДОДАТОК В

Слайди презентації

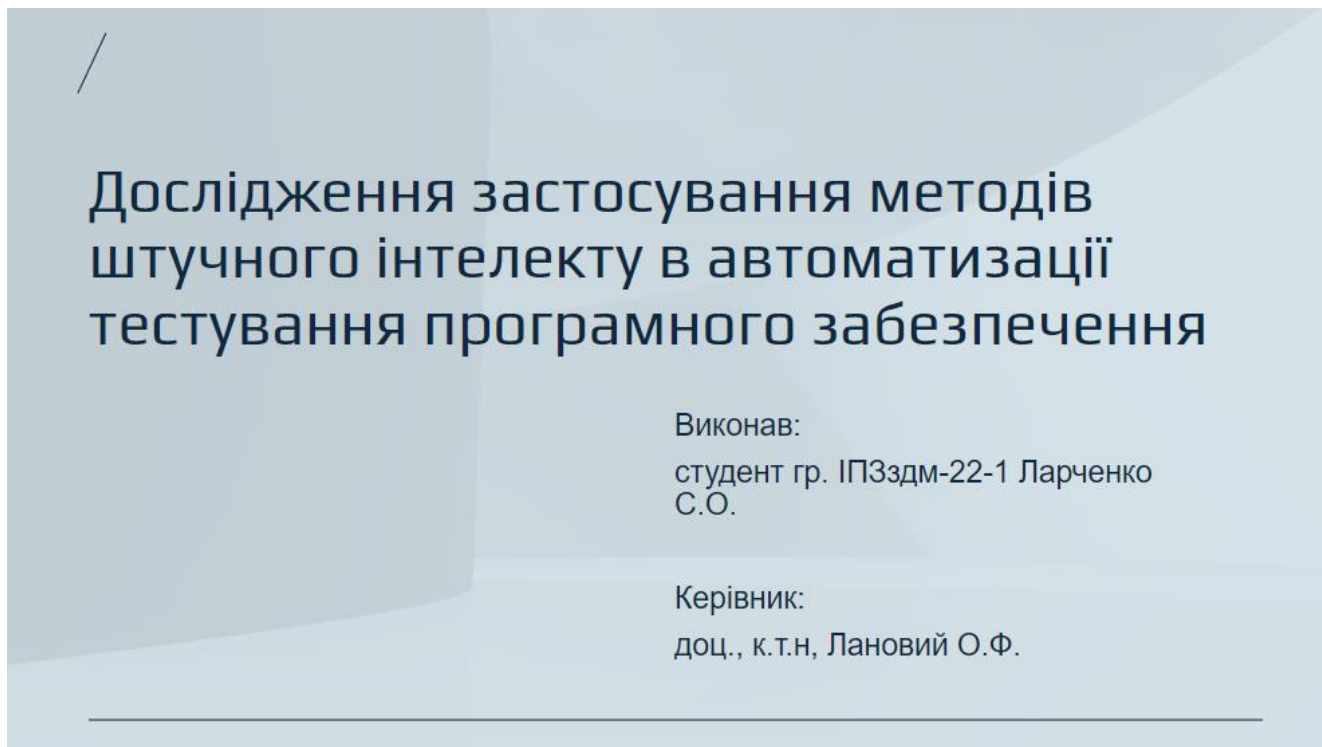


Рисунок В.1 – Слайд 1

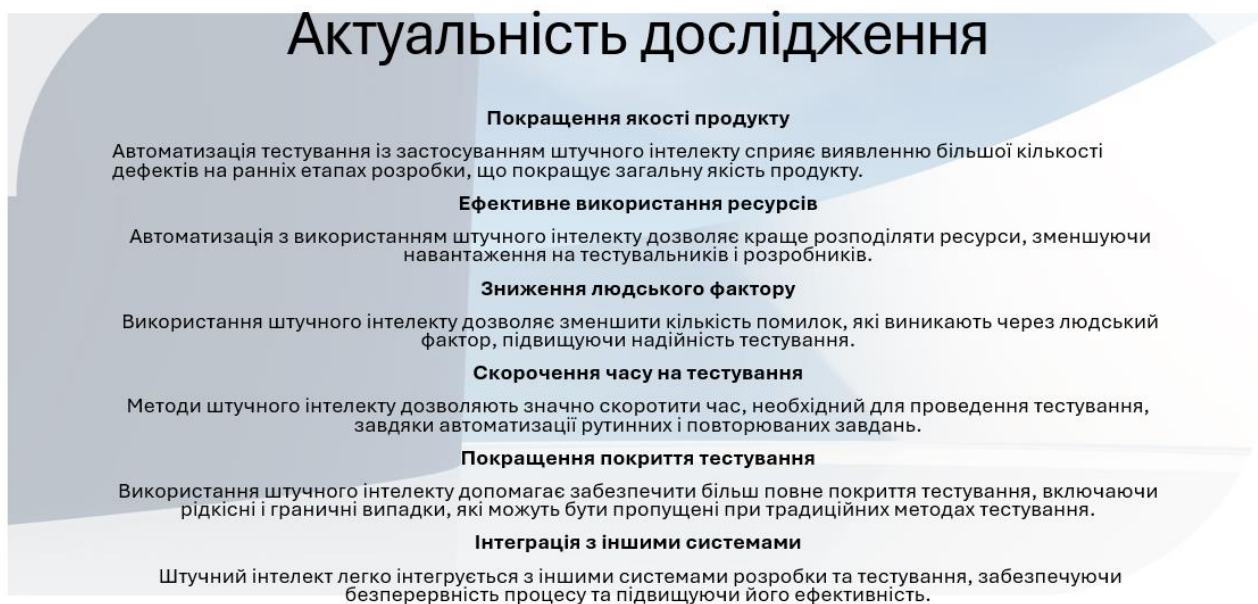


Рисунок В.2 – Слайд 2

Аналіз проблемної області

Мета роботи – дослідження методів штучного інтелекту для автоматизації тестування програмного забезпечення у веб-застосунках. Розробка рекомендації по використанню моделі ШІ, за допомогою якої можна генерувати автоматизовані тести.

- Визначити доцільність використання методів ШІ в автоматизації тестування ПЗ.
- Оцінити швидкість створення автоматичних тестових сценаріїв.
- Оцінити якість виконання тестів.
- Оцінити здатність та швидкість знаходження багів.

Рисунок В.3 – Слайд 3

Постановка задачі

- Визначити метод ШІ, який буде використовуватись у дослідженні.
- Визначити мовні моделі, які будуть використовуватись у дослідженні.
- Створити середовище для виконання отриманих тестів та збору інформації про результати їх виконання.
- Провести експерименти з генерації автоматизованих тестів
- Провести аналіз отриманих результатів та сформулювати висновки роботи.

Рисунок В.4 – Слайд 4

Планування експериментального дослідження

- Визначення мовних моделей, які будуть використовуватись у дослідженні.
- Вибір та побудова середовища для виконання згенерованих тестів.
- Етап генерації тестів за допомогою моделей
- Виконання тестів та вимірювання і збір результатів виконання
- Аналіз отриманих результатів
- Висновки та рекомендації щодо використання результатів дослідження

Рисунок В.5 – Слайд 5

Аналіз та вибір методу Machine Learning для дослідження

Метод машинного навчання є найкращою альтернативою. Процес вибору альтернатив та критеріїв оцінки включав нормалізацію критеріїв до шкали від 0 до 1, застосування лінійної адитивної згортки з ваговими коефіцієнтами для розрахунку корисності кожної альтернативи, порівняння результатів та аналіз множини Парето, що показав домінування машинного навчання над іншими методами.

У розрахунках використовувалися такі методи: машинне навчання, нейронні мережі, метод опорних векторів, обробка природної мови, логічне програмування.

Рисунок В.6 – Слайд 6



Рисунок В.7 – Слайд 7

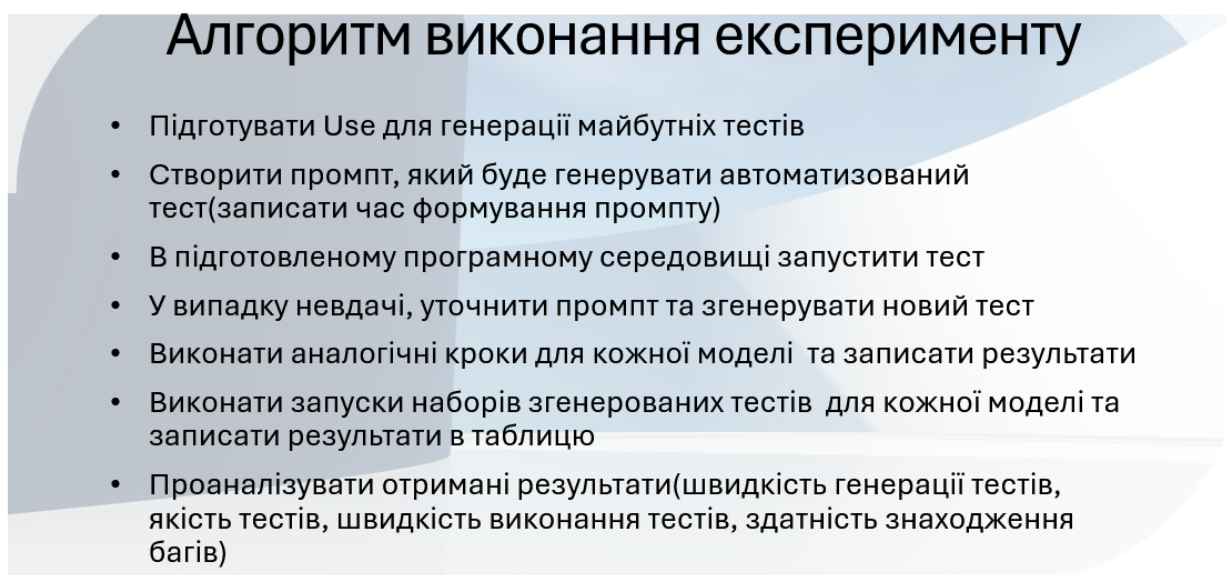


Рисунок В.8 – Слайд 8

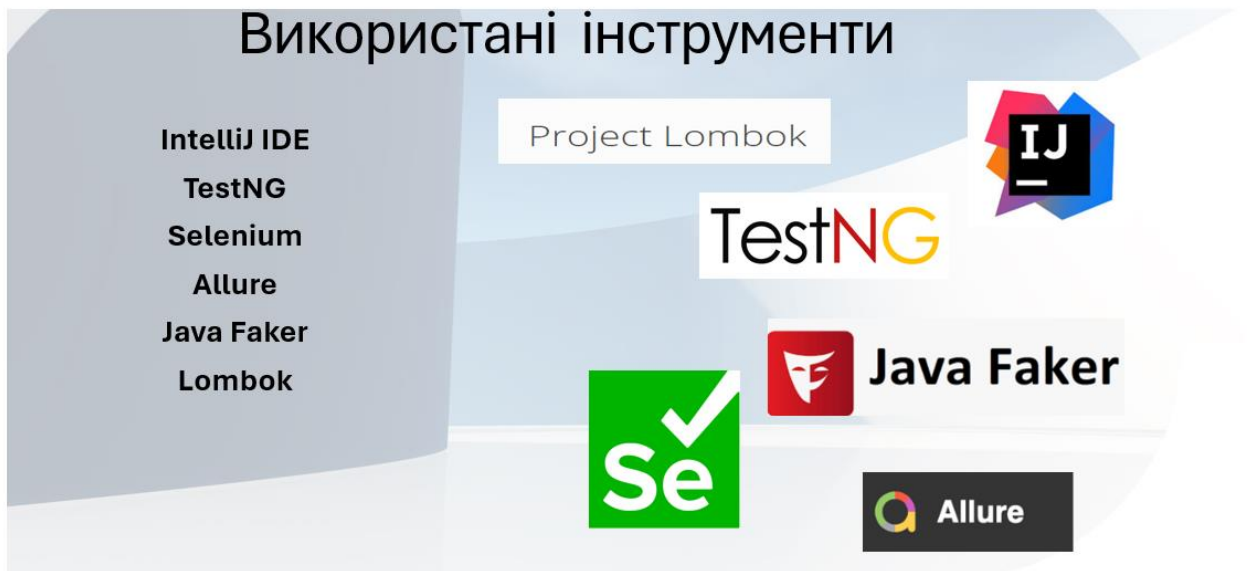


Рисунок В.9 – Слайд 9

Структура Page Object

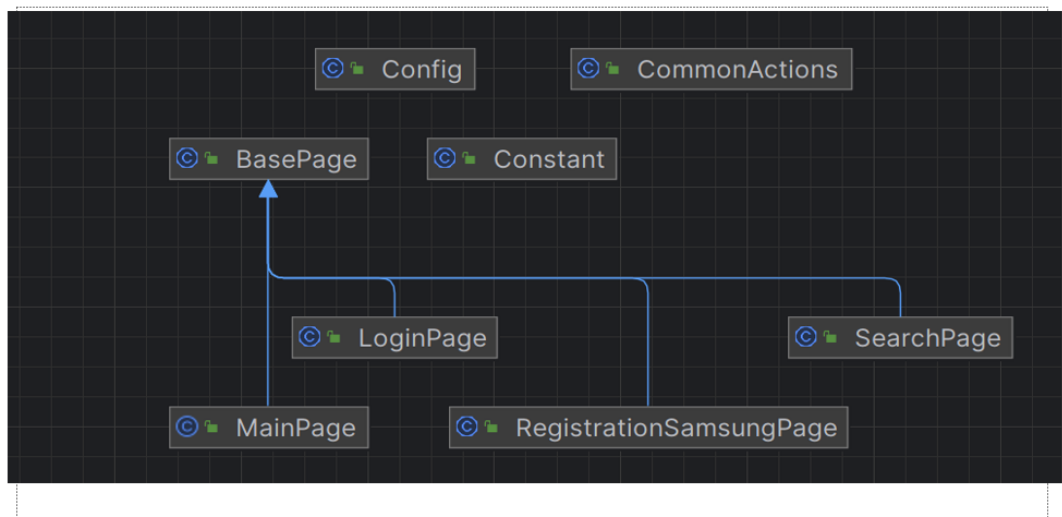


Рисунок В.10 – Слайд 10

Приклад промптів для створення автоматизованого тесту

"З наданого Use Case для тестування сайту <https://samsungshop.com.ua/auth/register/> створи автоматизований тест: перевірка можливості успішної реєстрації нового користувача з правильною інформацією. Згідно форми реєстрації маємо такі поля: ім'я, номер телефону, електронна адреса, пароль, та кнопка зареєструватись. Перевірка відображення номеру телефону на екрані після реєстрації."

"На основі наданого Use Case, локаторів, методів створи автотест в класі RegistrationGPT4Test, використовуючи Java, Selenium, TestNG, використовуючи патерн Page object. Private змінні в тестовому методі виликай через get methods(наприклад newRegisterSamsung.getNameInputField()). Для генерації текстових даних використовуй бібліотеку 'javafaker'. назви змінних для тесту nameInputField, userEmailInputField, userPhoneNumberInputField, userPasswordField, registerButton, registerPhoneNumberCheck. Методи класу RegistrationSamsungPage для використання в тесту: public void clickOnButton(By locator) { driver.findElement(locator).click(); } public void fillOutField(By locator, String text){ driver.findElement(locator).sendKeys(text); }"

Рисунок В.11 – Слайд 11

Приклад згенерованого автотесту GPT4

```

SergLach *
@Test(priority = 0)
public void testSuccessfulRegistration() {
    String name = faker.name().fullName();
    String email = faker.internet().emailAddress();
    String phone = "033" + faker.regexify("[0-9]{7}");
    String password = faker.internet().password();

    basePage.open( url: BASE_URL + REGISTER);

    newRegisterSamsung.fillOutField(newRegisterSamsung.getNameInputField(), name);
    newRegisterSamsung.fillOutField(newRegisterSamsung.getUserEmailInputField(), email);
    newRegisterSamsung.fillOutField(newRegisterSamsung.getUserPhoneNumberInputField(), phone);
    newRegisterSamsung.fillOutField(newRegisterSamsung.getUserPasswordField(), password);
    newRegisterSamsung.clickOnButton(newRegisterSamsung.getRegisterButton());

    String registeredPhone = driver.findElement(newRegisterSamsung.getRegisterPhoneNumberCheck()).getText();
    Assert.assertEquals( actual: "+38" + phone, registeredPhone,
        message: "Registered phone number is not the same as entered");
}

```

Рисунок В.12 – Слайд 12

Зведена таблиця генерації автоматизованих тестів

Сценарій	Великі мовні моделі(LLM)					
	GPT4		GPT-3.5		Bard (Gemini)	
	промпти для створення автотесту, кількість	виконання запиту, сек.	промпти для створення автотесту, кількість	виконання запиту, сек.	промпти для створення автотесту, кількість	виконання запиту, сек.
1. Перевірка можливості успішної реєстрації нового користувача з правильною інформацією.	2	92.21	6	89.25	4	43.91
2. Перевірка валідації порожніх полів воду	4	66.72	5	72.33	4	39.91
3. Перевірка відображення помилки при введенні імені менше 4 символів	1	9.61	1	9.52	2	15.29
4. Перевірка відображення помилки при введенні не вірного формату пошти	1	9.38	2	15.4	2	11.26
5. Перевірка відображення помилки при введенні не вірного формату телефону	1	8.57	1	8.84	2	10.98
6. Перевірка відображення помилки при введенні паролю менше 8 символів	1	8.42	2	20.36	1	4.81
7. Перевірка відображення помилки при введенні паролю більше 30 символів	1	9.88	1	9.67	1	5.53
8. Спроба реєстрації користувача з вже зареєстрованою поштою	1	19.57	2	26.54	2	12.4
9. Вхід в кабінет з валідними даними.	1	10.18	1	9.68	1	8.04
10. Пошук товару по сайті	2	67.01	3	34.99	3	23.2
Всього:	15	301.55	24	296.58	22	175.33

Рисунок В.13 – Слайд 13

Вивід результатів запуску тестів згенерованих моделлю Gemini

The screenshot displays the Allure test results interface. On the left, a sidebar contains navigation options: Overview, Categories, Suites, Graphs, Timeline, Behaviors, and Packages. The main area shows a list of test suites under the path 'C:/Users/Serhill/IdeaProjects/Practic/src/test/resources'. The suite 'test.registration.RegistrationBardTest' is expanded, showing a list of test cases with their status (pass/fail), name, and duration. Test case #10, 'testSearchSmartSockets', is highlighted in yellow and marked as failed. To the right, a detailed view of the failed test case is shown, indicating the failure message: 'expected [8] but found [3]'. The failure details include categories ('Product defects'), severity ('normal'), and duration ('02s 670ms'). The execution steps are listed as 'Set up' and 'Tear down'.

Рисунок В.14 – Слайд 14

Результати запуску тестів для Gemini

запуск тестів, сек.	Gemini									
	run1	run2	run3	run4	run5	run6	run7	run8	run9	run10
Сценарій										
1. Перевірка можливості успішної реєстрації нового користувача з правильною інформацією.	3.668	4.459	4.716	5.159	4.702	4.505	4.704	4.479	4.682	4.448
2. Перевірка валідації порожніх полів вводу	4.839	4.112	4.131	4.349	4.135	4.1	4.061	4.055	4.163	4.285
3. Перевірка відображення помилки при введенні імені менше 4 символів	6.964	6.986	6.955	7.082	7.02	6.952	6.952	6.932	6.907	6.933
4. Перевірка відображення помилки при введенні не вірного формату пошти	5.07	2.937	3.077	3.37	2.936	3.041	3.434	2.942	2.985	2.919
5. Перевірка відображення помилки при введенні не вірного формату телефону	2.927	3.335	2.932	3.091	3.07	2.911	2.927	2.91	2.93	2.94
6. Перевірка відображення помилки при введенні паролю менше 8 символів	2.932	3.059	3.014	3.21	2.908	3.152	2.877	2.914	2.923	3.04
7. Перевірка відображення помилки при введенні паролю більше 30 символів	2.946	2.989	3.163	3.182	2.958	2.972	3.041	2.96	2.967	2.954
8. Спроба реєстрації користувача з вже зареєстрованою поштою	3.92	3.395	3.369	3.697	3.334	3.344	3.34	3.327	3.412	3.475
9. Вхід в кабінет з валідними даними.	3.92	3.597	3.473	3.386	3.421	3.472	3.421	3.43	3.568	3.535
10. Пошук товару по сайті	4.206	4.268	4.301	4.899	4.251	4.477	4.177	4.823	4.208	4.241
Всього витрачено часу Bard:	41.39	39.137	39.131	41.425	38.735	38.926	38.934	38.772	38.745	38.77

Рисунок В.15 – Слайд 15

Результати запуску тестів для GPT4

запуск тестів, сек.	GPT4									
	run1	run2	run3	run4	run5	run6	run7	run8	run9	run10
Сценарій										
1. Перевірка можливості успішної реєстрації нового користувача з правильною інформацією.	5.212	5.548	3.921	6.004	5.064	5.372	4.948	5.125	5.372	5.088
2. Перевірка валідації порожніх полів вводу	4.354	4.502	4.149	4.481	4.188	4.12	4.576	4.176	4.12	4.201
3. Перевірка відображення помилки при введенні імені менше 4 символів	7.153	7.332	7.12	7.39	7.213	7.092	7.412	7.371	7.092	7.299
4. Перевірка відображення помилки при введенні не вірного формату пошти	3.498	3.417	3.204	3.5	3.276	3.248	3.8	3.502	3.248	3.394
5. Перевірка відображення помилки при введенні не вірного формату телефону	3.355	3.909	3.379	3.984	3.637	3.259	3.919	3.549	3.259	3.59
6. Перевірка відображення помилки при введенні паролю менше 8 символів	3.328	3.544	3.435	4.093	3.62	3.18	4.002	3.633	3.18	3.486
7. Перевірка відображення помилки при введенні паролю більше 30 символів	3.46	3.964	3.437	4.039	3.602	3.572	4.026	3.598	3.572	3.761
8. Спроба реєстрації користувача з вже зареєстрованою поштою	3.708	3.932	3.565	4.399	3.945	3.564	4.333	3.921	3.564	3.795
9. Вхід в кабінет з валідними даними.	4.737	3.835	3.814	3.997	3.748	3.718	4.126	3.915	3.718	4.112
10. Пошук товару по сайті	9.314	5.095	5.079	4.822		4.389	4.879	4.435	4.389	5.088
Всього витрачено часу GPT4:	48.12	45.078	41.103	46.709	38.293	41.514	46.021	43.225	41.514	43.814

Рисунок В.16 – Слайд 16

Результати запуску тестів для GPT-3.5

запуск тестів, сек.	GPT3.5									
	run1	run2	run3	run4	run5	run6	run7	run8	run9	run10
Сценарій										
1. Перевірка можливості успішної реєстрації нового користувача з правильною інформацією.	8.77	5.224	5.418	5.289	5.305	5.7	5.474	4.943	5.23	6.329
2. Перевірка валідації порожніх полів вводу	4.373	4.069	4.25	4.284	4.298	4.301	4.225	4.049	4.099	4.824
3. Перевірка відображення помилки при введенні імені менше 4 символів	7.453	7.127	7.11	7.319	7.335	7.577	7.36	7.102	7.173	7.653
4. Перевірка відображення помилки при введенні не вірного формату пошти	3.672	3.405	3.615	3.657	3.617	3.647	3.606	3.435	3.503	4.012
5. Перевірка відображення помилки при введенні не вірного формату телефону	3.621	4.196	3.35	4.034	4.1	3.655	4.139	3.377	3.825	3.981
6. Перевірка відображення помилки при введенні паролю менше 8 символів	3.323	4.034	3.3	4	3.984	3.633	4.2	3.513	3.647	3.935
7. Перевірка відображення помилки при введенні паролю більше 30 символів	3.644	4.3	3.443	4.066	4.414	3.646	4.208	3.395	3.949	3.947
8. Спроба реєстрації користувача з вже зареєстрованою поштою	3.852	4.145	3.742	4.197	4.364	3.889	4.264	3.59	3.867	4.102
9. Вхід в кабінет з валідними даними.	4.03	4.214	3.759	4.649	4.058	3.825	4.17	3.867	3.896	4.508
10. Пошук товару по сайті	5.348	4.847	5.229	5.086	4.718	4.521	5.145	4.907	4.432	5.277
Всього витрачено часу GPT3-5:	48.09	45.561	43.216	46.581	46.193	44.394	46.791	42.178	43.621	48.568

Рисунок В.17 – Слайд 17

Діаграми знаходження багу та витрат часу на виконання набору тестів

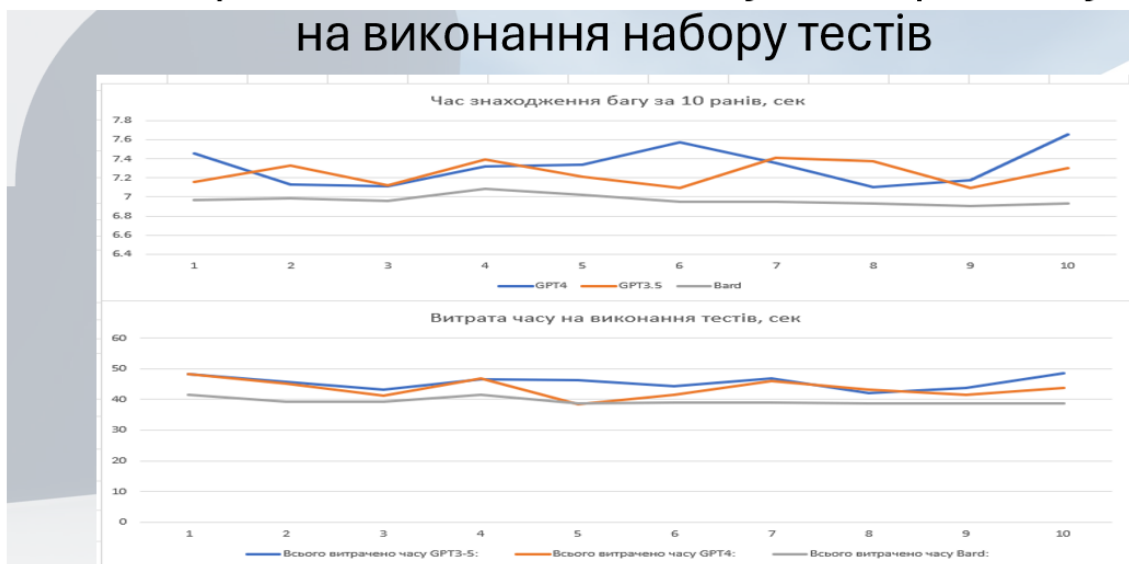


Рисунок В.18 – Слайд 18

Приклад використання результатів дослідження в реальному проекті

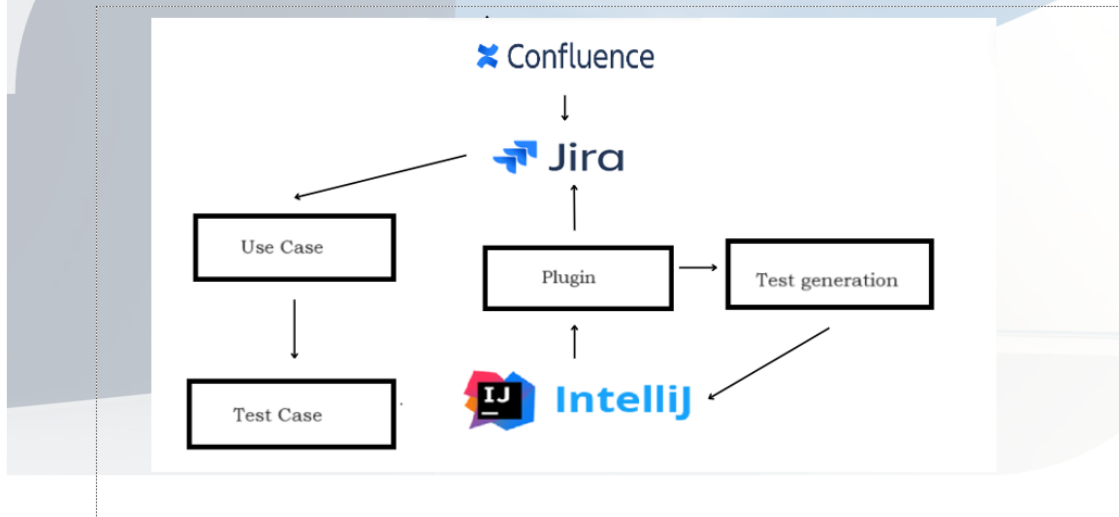


Рисунок В.19 – Слайд 19

Висновки

- Вибрано для проведення експерименту три мовні моделі: GPT 3-5, GPT 4, Gemini.
- Розроблено план експериментального дослідження
- Розроблено три набори автоматизованих тестів на основі різних мовних моделей з використанням патерну Page Object
- На основі отриманих результатів було запропоновано приклад використання результатів дослідження в реальному проекті
- Подано тези доповіді на двадцять восьмий міжнародний форум РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У ХХІ СТОЛІТТІ», що проходив 16 – 18 квітня 2024 р.

Рисунок В.20 – Слайд 20

ДОДАТОК Г

Апробація результатів роботи

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
РАДІОЕЛЕКТРОНІКИ**

**МАТЕРІАЛИ ХХVІІІ МІЖНАРОДНОГО МОЛОДІЖНОГО
ФОРУМУ**

**«РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ
У ХХІ СТОЛІТТІ»**

16 – 18 квітня 2024 р.

Том 6

**КОНФЕРЕНЦІЯ
«ІНФОРМАЦІЙНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ»
INFORMATION INTELLIGENT SYSTEMS**

Харків 2024

ЗАСТОСУВАННЯ ШІ В АВТОМАТИЗАЦІЇ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Ларченко С. О.

Науковий керівник – к.т.н., доц. Лановий О. Ф.

Харківський національний університет радіоелектроніки, каф. ПІ

м.Харків, Україна

e-mail: serhii.larchenko.cpe@nure.ua

This work focuses on the latest techniques for applying artificial intelligence (AI) in software test automation, in particular, the ability to generate test scenarios using AI. The focus is on using machine learning technologies to improve the automation and efficiency of testing processes. Challenges and prospects for further research in this area are discussed, with an emphasis on the search for the latest technologies for continuous improvement of the quality and efficiency of automated software testing.

Автоматизація процесу тестування стала невід'ємною частиною сучасних практик в індустрії розробки програмного забезпечення. Існує значна кількість методів та технологій контролю за якістю та надійністю ПЗ [1]. Автоматизовані тести можуть бути виконані швидко та ефективно та здатні виконувати багато операцій одночасно і працювати безперервно. Виконання автоматизованого тестування визначає, чи було отримано відповідні результати, та чи може бути проведено об'єднання призвести до успіху або збою інтеграції [2]. Машинне навчання (ML) та інші технології штучного інтелекту (ШІ) можуть сприяти ширшому покриттю тестування і виявленню багів, які можуть знизити рівень мануального тестування. Нарешті, автоматизоване тестування має важливе значення для збереження ресурсів.

В останні роки штучний інтелект з генерацією (GenAI) привернув увагу та зацікавленість у сфері програмної інженерії. Інструменти GenAI, такі як GitHub Copilot і ChatGPT, були швидко досліджені в різних професійних контекстах.

Дослідження використання ШІ для автоматизованого тестування спрямоване на визначення оптимальних стратегій застосування штучного інтелекту для створення автоматизованих тестових сценаріїв, для подальшого використання в розробці тестового фреймворку.

Було розглянуто п'ять методів ШІ таких як: Machine Learning, Neural Networks, Support Vector Machines, Natural Language Processing, Logic Programming. Для визначення кращого методу було використано введення багатокритеріальної задачі вибору, яка передбачає наявність кількох конкуруючих критеріїв, які мають бути оптимізовані для досягнення найкращого результату у контексті цільової системи. Аналіз критеріїв для багатокритеріального вибору допомагає об'єктивно оцінити кожен із

варіантів, враховуючи різноманітні фактори, що впливають на вибір методу.

Методи ML, було обрано кращим для подальшого дослідження, тому що він дозволяє системам вчитися і зростати з досвіду, поступово покращуючи свою продуктивність і ефективність. Методи ML підвищують ефективність життєвого циклу програмної інженерії (SE) [3]. В автоматизації тестування, ML часто використовується для предиктивного аналізу, ідентифікації моделей та змінних, що викликають помилки, та автоматичного генерування сценаріїв тестування.

На етапі експерименту, будуть розглянуті можливості генерації автоматичних тестів, за допомогою Large Language Model (LLM), використовуючи User story, як основу для генерації авто-тесту. Для генерації авто-тестів будуть використовуватись такі мовні моделі як GPT-3.5, GPT-4, Bart.

Експерименти допоможуть зібрати інформацію про якість створених тестів, швидкість, точність виконання запиту, також на сформованих кожною моделлю наборах тестів, буде проведено порівняння в швидкості виконання сформованих авто-тестів, час знаходження помилки, оцінка придатності тесту для пошуку дефекту.

Основні висновки використання ШІ в автоматизації тестування, полягають у виявленні можливостей для підвищення якості програмного забезпечення, шляхом генерації тестових сценаріїв, зменшення часу, необхідного для проведення тестів, та збільшення точності виявлення дефектів. Крім того, це може сприяти покращенню розробки та впровадженню нових стратегій тестування, що в свою чергу призведе до постійного розвитку галузі автоматизованого тестування.

Список використаних джерел:

1. Лановий О. Ф. Візуалізація в методах тестування програмного забезпечення: thesis. 2017. URL: <http://openarchive.nure.ua/handle/document/9432> (дата звернення: 02.03.2024).
2. Фундукян А. Метод розстановки пріоритетів тестів. education and science of today: intersectoral issues and development of sciences / chair О. Лановий. 2021. URL: <https://doi.org/10.36074/logos-19.03.2021.v2.31> (дата звернення: 29.02.2024).
3. Kotti Z., Galanopoulou R., Spinellis D. Machine Learning for Software Engineering: A Tertiary Study. ACM Computing Surveys. 2022. URL: <https://doi.org/10.1145/3572905> (date of access: 29.02.2024).

