МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІМ. Ю.М. ПОТЕБНІ ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Порівняльне дослідження технологій автоматизованого тестування мобільних застосунків**

Виконав: студент 2 курсу, групи 8.1212-пзс-2
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
О.Є. Пасенко
(ініціали та прізвище)
Керівник доцент, к.ф.-м.н. Г.П. Коломоєць
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ Дісітел
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти другий (магістерський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Т.В. Критська
“ 01 ” вересня 2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Пасенко Олексій Євгенійович
(прізвище, ім’я, по батькові)

1.Тема роботи Порівняльне дослідження технологій автоматизованого тестування мобільних застосунків

керівник роботи доцент, к.ф.-м.н. Коломоєць Геннадій Павлович.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від
2. Строк подання студентом кваліфікаційної роботи 28.11.2023
3. Вихідні дані магістерської роботи

- комплект нормативних документів ;
- технічне завдання до роботи.

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

- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми розпізнавання мов та розробка методів її вирішення;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) 14 слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.09.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.09-05.09.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	06.09-10.09.23	виконано
3	Аналіз існуючих методів порівняння	14.09-26.09.23	виконано
4	Дослідження сучасних мобільних платформ	27.09-01.10.23	виконано
5	Дослідження автоматизованого тестування мобільних застосунків	02.10-03.10.23	виконано
6	Узгодження подальших дій з науковим керівником	04.10-10.10.23	виконано
7	Обрання інструментів автоматизованого тестування мобільних застосунків для дослідження: Android-застосунку	11.10-12.10.23	виконано
8	Розгортання тестового середовища та створення та прогонка димових тестів (основної функціональності), на кожному з фреймворків	10.10-21.10.23	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого порівняння	22.10-29.10.23	виконано
10	Порівняння вихідних даних після тестування Android-застосунку двома фреймворками	30.10-04.11.23	виконано
11	Написання результатів дослідження	06.11-12.11.23	виконано
12	Оформлення звіту	14.11-10.12.23	виконано

Студент _____ О.Є Пасенко
(підпис) (прізвище та ініціали)

Керівник роботи _____ Г. П. Коломоєць
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок : 124

Рисунків: 21

Таблиць: 1

Джерел: 30.

Пасенко О. Є. Порівняльне дослідження технологій автоматизованого тестування мобільних застосунків: кваліфікаційна робота магістра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник Коломоєць Г.П. Запоріжжя : ЗНУ, 2023. 124 с.

Мета цієї дипломної роботи полягає в порівняльному дослідженні інструментів автоматизованого тестування мобільних застосунків: Appium, Katalon Studio, Espresso для виявлення переваг, недоліків, особливостей та ефективності кожного з фреймворків. Дослідження виконане для Android-клієнта соціальної мережі Instagram. Основні завдання включають оцінку ефективності кожного інструменту в контексті реалізації тестів, а також порівняння їхньої зручності, продуктивності та можливостей взаємодії з Android-застосунком.

За результатами дослідження, найкращим інструментом автоматизації тестування за усіма дослідженими факторами, обраний Appium, найпростішим у використанні - Katalon Studio, найстабільнішим - Espresso. Результати дослідження будуть корисні фізичним особам або компаніям, які шукають для себе інструмент автоматизованого тестування мобільних застосунків, що повністю буде відповідати їхнім потребам.

Ключові слова: інструменти тестування, застосунок, інтерфейс, тестовий сценарій, Java, Android, Appium, Katalon, Espresso, Instagram.

SUMMARY

Pages: 124

Figures: 21

Tables: 1

Sources: 30

Pasenko O. E. Comparative Study of Automated Testing Technologies for Mobile Applications: Master's Thesis in the specialty 121 "Software Engineering" / Academic Supervisor Kolomoiets H.P. Zaporizhzhia : ZNU, 2023. 124 p.

The goal of this thesis is to conduct a comparative study of automated testing tools for mobile applications: Appium, Katalon Studio, Espresso. The aim is to identify the advantages, disadvantages, features, and effectiveness of each framework. The research is performed on the Android client of the social network Instagram. The main objectives include evaluating the performance of each tool in the context of test implementation and comparing their ease of use, productivity, and interaction capabilities with the Android application.

Based on the research results, Appium is selected as the best automation testing tool considering all investigated factors, Katalon Freemium is chosen as the most user-friendly, and Espresso is recognized as the most stable. The findings of this study will be beneficial for individuals or companies seeking an automated testing tool for mobile applications that fully aligns with their needs.

Keywords: testing tools, application, interface, test scenario, Java, Android, Appium, Katalon, Espresso, Instagram.

ЗМІСТ

ВСТУП

Актуальність теми

Зростання популярності мобільних застосунків приносить високі вимоги до їх якості. Користувачі очікують, що програми будуть надійними, безпечними та ефективними. Багато компаній залежать від успіху своїх мобільних застосунків для привертання та утримання клієнтів. Недоліки або помилки в програмі можуть призвести до негативного враження користувачів та втрати бізнесу.

Автоматизоване тестування стало необхідною складовою процесу розробки мобільних застосунків. Воно дозволяє автоматизувати процес перевірки функціональності, надійності, сумісності та продуктивності програми, що спрощує та прискорює процес тестування. Завдяки автоматизованому тестуванню розробники можуть виявити помилки та недоліки швидше, покращити якість програми та знизити ризик випуску продукту з дефектами.

Автоматизоване тестування має кілька переваг. По-перше, воно забезпечує швидку та ефективну перевірку функціональності та продуктивності програми, що дозволяє розробникам швидше виявляти та виправляти помилки. По-друге, автоматизоване тестування може бути повторюваним та надійним, оскільки воно виконується за допомогою програмних скриптів або інструментів. По-третє, воно дозволяє ефективно використовувати ресурси та зменшувати залежність від ручного тестування, що забезпечує економію часу та зусиль.

На сьогоднішній день існує велика кількість інструментів та технологій для автоматизованого тестування мобільних застосунків. Кожен з них має свої переваги та обмеження, а також різний рівень складності використання. Сучасні розробники стикаються з великим вибором технологій та потребою визначитися з найбільш ефективними інструментами для своїх проектів.

Таким чином, актуальність теми "Порівняльне дослідження технологій автоматизованого тестування мобільних застосунків" очевидна, у зв'язку з постійним зростанням популярності мобільних застосунків та важливістю забезпечення їх високої якості. Автоматизоване тестування стає необхідним інструментом для розробників у процесі створення мобільних застосунків, проте вибір оптимальних технологій та інструментів для автоматизованого тестування залишається складним завданням. Дослідження та порівняльний аналіз різних технологій автоматизованого тестування мобільних застосунків дозволять визначити найбільш ефективні підходи та інструменти, що сприятимуть поліпшенню процесу розробки та якості мобільних застосунків.

Мета і завдання дослідження

Основною метою даної дипломної роботи є проведення порівняльного дослідження інструментів автоматизованого тестування мобільних застосунків: Appium, Katalon Studio, Espresso для виявлення переваг, недоліків, особливостей та ефективності кожного з фреймворків. Дослідження виконане для Android-клієнта соціальної мережі Instagram. Результати дослідження будуть корисні для розробників, тестувальників та науковців, які працюють у галузі мобільних застосунків та тестування програмного забезпечення.

Об’єкт дослідження

Об’єктом дослідження є Android-клієнт соціальної мережі Instagram.

Предмет дослідження

Предметом дослідження є функціональність та зручність у використанні фреймворків автоматизованого тестування Appium, Katalon Studio, Espresso.

Методи дослідження

Для досягнення мети дослідження необхідно виконати наступні завдання:

1. Огляд літературних джерел з тематики роботи.
2. Визначення критеріїв порівняння тестових фреймворків.
3. Встановлення та налаштування інструментів автоматизації тестування.
4. Визначення базової функціональності застосунку, що буде тестуватись.
5. Розробка тест-плану та тестових сценаріїв.
6. Виконання тестів та аналіз результатів.
7. Формулювання висновків та рекомендацій.

Наукова новизна одержаних результатів

Науковою новизною роботи є порівняльне дослідження популярних, вільно розповсюджуваних інструментів, автоматизації тестування, що базується на узгодженому методологічному підході. Дослідження охопило спектр параметрів тестових фреймворків, таких як функціональність, швидкодія, стабільність тощо.

Практичне значення одержаних результатів

Результати, отримані у дипломній роботі, мають практичне значення для розробників та тестувальників мобільних застосунків, оскільки вони надають об'єктивну інформацію про переваги та недоліки кожного з інструментів, допомагають визначити оптимальні налаштування для кожного інструменту, що позитивно впливає на якість та ефективність процесу тестування.

Крім того, отримані результати можуть послужити основою для подальших досліджень та вдосконалення існуючих інструментів автоматизованого тестування мобільних додатків. Вони можуть сприяти розвитку нових методів та підходів до тестування, що забезпечуватимуть більш ефективну та якісну роботу з мобільними застосунками.

Апробація одержаних результатів

Результати дослідження були представлені на XVI науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених Запорізького

національного університету «Молода наука-2023» [1], а також на III всеукраїнської науково-практичної конференції за участю молодих науковців Інженерного навчально-наукового інституту Запорізького національного університету [2].

Глосарій

Автоматизоване тестування — це процес використання програмних засобів для виконання тестів на автоматизовану перевірку функціональності, надійності та продуктивності мобільних застосунків.

Мобільні застосунки — це програмні додатки, призначені для використання на мобільних пристроях, таких як смартфони та планшети.

Appium — це відкритий фреймворк для автоматизації тестування мобільних застосунків на різних платформах, включаючи Android та iOS.

Espresso — це відкрита бібліотека для автоматизованого тестування Android-застосунків, розроблена компанією Google. Вона надає широкий набір інструментів та API для зручного та ефективного тестування.

Java — Об'єктно-орієнтована мова програмування, яка використовується для розробки різноманітних застосунків, включно, мобільні додатки на платформі Android.

Android — Операційна система для мобільних пристроїв, розроблена компанією Google. Android дозволяє розробникам створювати різнобічні мобільні додатки для смартфонів та планшетів.

Instagram — Соціальна мережа для обміну фотографіями та відео. Розроблена компанією Facebook, вона є однією з найпопулярніших платформ для спільного використання візуального контенту.

Android Studio — Інтегроване середовище розробки (IDE) для платформи Android. Надійний інструмент для створення, тестування і вдосконалення мобільних додатків.

Appium Server GUI — Графічний інтерфейс користувача для Appium, що надає можливість налаштування та керування процесом автоматизації тестування мобільних додатків.

Appium Inspector — Інструмент для візуального інспектування елементів інтерфейсу користувача мобільних додатків під час тестування з використанням Appium.

Android Debug Bridge (ADB) — Командний інтерфейс для взаємодії з пристроями Android. Використовується для відладки, копіювання файлів і виконання різних команд на пристроях Android.

Node.js — Платформа з відкритим кодом для виконання JavaScript на сервері. Використовується, наприклад, для розгортання серверної частини додатків.

Katalon Studio — Інтегроване середовище розробки та автоматизованого тестування, спеціально призначене для веб та мобільних додатків.

Selenium — Набір інструментів для автоматизованого тестування вебдодатків. Якщо використовується для мобільних додатків, то часто як бібліотека для мов програмування, таких як Java.

Функціональність — це здатність мобільного застосунку виконувати необхідні функції та завдання згідно вимог користувача.

Швидкодія — це швидкість та продуктивність мобільного застосунку під час його виконання та реакції на взаємодію користувача.

Масштабованість — це здатність мобільного застосунку працювати ефективно та стабільно, навіть при збільшенні обсягу даних чи навантаженні.

Стабільність — це надійність та стійкість мобільного застосунку під час його виконання та взаємодії з різними компонентами системи.

Розширюваність — це здатність мобільного застосунку до додавання нових функцій, модулів або компонентів без значних змін в його архітектурі.

UI-тестування — це тестування графічного інтерфейсу (UI) мобільного застосунку з метою перевірки його відповідності дизайну, правильності розташування елементів і взаємодії з користувачем.

API-тестування — це тестування програмного інтерфейсу (API) мобільного застосунку з метою перевірки правильності передачі даних та взаємодії з іншими системами чи сервісами.

Тестовий сценарій — це список кроків або дій, які виконуються під час тестування мобільного застосунку для перевірки його функціональності чи характеристик.

Тестовий набір — це колекція тестових сценаріїв, які виконуються разом для перевірки різних аспектів мобільного застосунку.

Матриця підтримки — це таблиця, що відображає сумісність та підтримку мобільних платформ, пристроїв та версій операційних систем інструментом автоматизованого тестування.

Автоматизація інтерфейсу користувача (UI) — це процес автоматизації взаємодії користувача з мобільним застосунком шляхом емуляції дій та перевірки результатів.

Інтеграційні тести — це тести, що перевіряють взаємодію між різними компонентами або модулями мобільного застосунку.

Юніт-тести — це тести, що перевіряють коректність роботи окремих функціональних блоків або модулів мобільного застосунку.

Навантажувальне тестування — це тести, що оцінюють реакцію мобільного застосунку на великі навантаження або одночасну роботу багатьох користувачів.

Крос-платформове тестування — це тестування, що здійснюється на різних мобільних платформах, таких як Android і iOS, з використанням спеціальних інструментів або фреймворків.

Інструменти Continuous Integration (CI) — це системи, які автоматизують процес збірки, тестування та розгортання мобільного застосунку з метою забезпечення постійного контролю якості.

Локалізація — це процес адаптації мобільного застосунку для використання в різних регіонах та культурах шляхом перекладу текстів, форматування дат та інших локальних налаштувань.

Деструктивне тестування — це тести, що не впливають на роботу мобільного застосунку та не змінюють його стану, а лише перевіряють його функціональність.

Гібридні застосунки — це мобільні застосунки, що поєднують в собі веб-технології та можливості нативних додатків.

Тестування безпеки — це процес перевірки мобільного застосунку на вразливості та можливість несанкціонованого доступу до даних.

Переносимість — це здатність мобільного застосунку працювати на різних пристроях, розмірів екрану та платформах з мінімальними змінами.

Перевантаження — це ситуація, коли навантаження або кількість користувачів перевищує максимальні можливості мобільного застосунку, що призводить до його некоректної роботи або аварійного завершення.

Модульне тестування — це тестування окремих модулів, класів або функцій мобільного застосунку з метою перевірки їх роботи відокремлено від інших компонентів.

Регресійне тестування — це перевірка мобільного застосунку після внесення змін або оновлень для виявлення нових помилок або негативних впливів на раніше працюючий функціонал.

Тестування на реальних пристроях — це тестування мобільного застосунку на фізичних пристроях різних моделей та варіацій для перевірки сумісності та правильності відображення.

Покриття коду (Code coverage) — це метрика, що вказує на те, яка частина програмного коду була протестована тестами.

Фрагментація пристроїв (Device fragmentation) — це розподіленість різних моделей та версій мобільних пристроїв, що може вплинути на роботу мобільного застосунку.

Емулятор (Emulator) — це програмне забезпечення, що емулює роботу мобільного пристрою, дозволяючи тестувати застосунки без фізичного пристрою.

Тестування Monkey (Monkey testing) — це автоматизоване тестування, яке випадково вводить дії та взаємодіє з мобільним застосунком для виявлення потенційних помилок.

Тестування продуктивності (Performance testing) — це тестування, що оцінює швидкість, ефективність та стійкість роботи мобільного застосунку під великим навантаженням.

Регресійне тестування (Regression testing) — це перевірка мобільного застосунку після внесення змін для переконання в тому, що вже раніше працюючий функціонал не був порушений.

Знімок екрану (Screenshot) — це зображення, що відображає вигляд мобільного застосунку на екрані під час виконання тесту.

Фреймворк автоматизації тестування (Test automation framework) — це сукупність інструментів, бібліотек та прийомів, що допомагають забезпечити структуру та організацію автоматизованих тестів.

Тестовий випадок (Test case) — це конкретний сценарій або послідовність дій, що повинні бути виконані під час тестування для перевірки певної функціональності.

Тестові дані (Test data) — це дані, що використовуються під час виконання тестів, включаючи вхідні значення, передумови та очікувані результати.

РОЗДІЛ 1 СУЧАСНА КЛАСИФІКАЦІЯ МОБІЛЬНИХ ЗАСТОСУНКІВ ТА ТЕХНОЛОГІЙ

1.1 Огляд сучасних мобільних платформ

У цьому розділі ми розглянемо сучасні мобільні платформи, щоб надати читачеві розуміння їхніх особливостей та технічних характеристик. Аналіз Android, iOS, Windows 10 Mobile та Microsoft Launcher дозволить визначити унікальні аспекти кожної платформи та їх вплив на розробку та тестування мобільних застосунків.

1.1.1 Android

Android - оперативна система, розроблена компанією Google, визначає стандарти сучасних мобільних технологій та володіє найбільшим світовим ринком мобільних пристроїв [3]. Її відкрита архітектура дозволяє створювати різноманітні додатки, які можуть взаємодіяти з різними сервісами та апаратним забезпеченням. З цього погляду, Android стає ключовим гравцем для багатьох розробників, оскільки він дозволяє створювати додатки для різних типів пристроїв та версій операційної системи.

Android має величезну кількість користувачів та широкий розмах пристроїв, від бюджетних до флагманських, що відображається у високій різноманітності апаратного забезпечення. Така різноманітність може створювати виклики для розробників та тестувальників, оскільки необхідно забезпечити оптимальну працездатність на різних пристроях та версіях операційної системи.

Android має ряд основних характеристик, які визначають його високий рівень популярності [3]:

- Відкритість та вільність: Android - відкрита платформа, що надає широкі можливості в роботі з кодом та взаємодії з іншими додатками.

- Широкий асортимент пристроїв: Оскільки Android використовується на багатьох різновидностях пристроїв від різних виробників, розробники мають можливість створювати додатки для різних цільових аудиторій та враховувати особливості апаратного забезпечення.
- Google Play Store: Великий магазин додатків, де користувачі можуть легко знаходити та встановлювати додатки, сприяє поширенню та використанню додатків на платформі.

Один із визначальних елементів розробки на Android - це Android SDK, який надає необхідні інструменти та бібліотеки для розробки додатків. Разом з Android Studio, інтегрованою середовищем розробки, можна створювати ефективні, потужні та відзначені застосунки для мобільних пристроїв. Вибір мови програмування, таких як Java або Kotlin, дає можливість працювати зручно та вибирати оптимальний підхід до вирішення конкретних завдань[3].

Основні компоненти архітектури Android включають:

- Activity (Діяльність): Це основний елемент користувацького інтерфейсу, відповідає за взаємодію з користувачем. Один додаток може мати декілька діяльностей, які взаємодіють між собою.
- Service (Сервіс): Цей компонент використовується для виконання фонових задач, навіть коли додаток не активний. Сервіси можуть взаємодіяти з іншими компонентами та забезпечувати продовження роботи додатка в фоновому режимі.
- Broadcast Receiver (Приймач трансляцій): Відповідає за приймання та обробку трансляцій, таких як сповіщення від системи чи інших додатків. Це дозволяє реагувати на різні події пристрою або додатків.
- Content Provider (Постачальник контенту): Відповідає за управління та надання доступу до даних додатків. Забезпечує обмін даними між різними додатками.

Розробка та тестування додатків на Android вимагає уважного вивчення цих характеристик [3], а також врахування різноманіття пристроїв та версій ОС для забезпечення стабільності та високої ефективності додатків.

Переваги Android:

1. Велика кількість користувачів та широкий розповсюджений ринок.
2. Відкрита платформа, що надає велику свободу та гнучкість.
3. Розгалужена екосистема з багатим набором бібліотек та інструментів розробки.

Недоліки Android:

1. Фрагментація платформи, оскільки існує багато різних версій Android та пристроїв з різними характеристиками.
2. Низька сумісність між різними пристроями.
3. Більше проблем з безпекою порівняно з іншими платформами.

Android - це потужна та розширювана платформа для розробки мобільних додатків. Вона володіє багатим функціоналом, мультиплатформенністю та відкритістю, що робить її привабливою для розробників. За допомогою Android Studio, SDK та інших інструментів розробки, можуть ефективно створювати високоякісні Android-додатки та забезпечувати їх тестування та підтримку.

1.1.2 iOS

Ios - операційна система виробництва компанії Apple, займає зазначене місце в елітному світі мобільних платформ [4]. Завдяки своїй унікальній екосистемі, стійкому контролю над апаратним та програмним забезпеченням, а також спрощеному порядку розробки. iOS визначає стандарти якості та дизайну.

iOS, як операційна система, має ряд ключових характеристик, що формують його виняткову позицію в світі мобільних технологій [4]:

- Єдність обладнання та програмного забезпечення. Однією з ключових рис iOS є тісна інтеграція обладнання та програмного забезпечення. Кожен пристрій, випущений компанією Apple, працює на iOS, що дозволяє стабільну та оптимізовану роботу.

- Спрощена процедура розробки. Використання Xcode та мови програмування Swift робить процес розробки додатків для iOS досить спрощеним та зручним для розробників. Висока продуктивність та зручний інтерфейс сприяють створенню високоякісних додатків.
- Екосистема Apple. iOS входить в екосистему Apple, яка включає iPhone, iPad, Apple Watch та Mac. Ця єдина екосистема створює сприятливі умови для інтерактивності та синергії між пристроями, що дозволяє користувачам насолоджуватися безшовними переходами.
- Високий стандарт дизайну. Apple відомий своїм високим стандартом дизайну, який стає характерною рисою додатків для iOS. Це створює єдність та зручність користування для користувачів.
- Безпека та приватність. iOS славиться своєю високою захищеністю та уважністю до приватності користувачів. Інтеграція функцій шифрування та контролю дозволів допомагає забезпечити конфіденційність даних.

Серцем розробки під iOS є Xcode та мова програмування Swift, які вирізняються високою продуктивністю та швидкістю. Xcode – інтегроване середовище розробки для macOS – дозволяє розробникам легко створювати додатки для iPhone, iPad, Apple Watch та інших пристроїв.

Архітектура iOS є системою, побудованою на кількох ключових компонентах, які визначають її унікальний характер. Ці компоненти включають [4]:

- Core OS: Це ядро операційної системи, яке відповідає за низькорівневі операції, включаючи роботу з пам’яттю та мережею.
- Core Services: Набір функцій та бібліотек, які надають базові сервіси для додатків, такі як бази даних, аутентифікація та інші.
- Media: Відповідає за обробку мультимедійних даних, включаючи графіку та аудіо.

- Cocoa Touch: Фреймворк для розробки користувацького інтерфейсу та взаємодії з користувачем. Включає інструменти для роботи з сенсорним екраном, анімації та інші.

Ці компоненти працюють разом, створюючи стабільну та ефективну основу для розробки та функціонування додатків під iOS. Розуміння цієї архітектури важливо для розробників та тестувальників, щоб ефективно використовувати інструменти та забезпечити високу якість мобільних додатків на цій платформі [5].

Переваги iOS:

1. Інтуїтивний та естетичний інтерфейс користувача.
2. Оптимізована для продуктивності та стабільності платформа.
3. Висока сумісність між різними пристроями та версіями iOS.

Недоліки iOS:

1. Обмежена екосистема та контроль Apple над розробкою та розповсюдженням застосунків.
2. Високі вимоги до апаратного забезпечення.
3. Висока вартість розробки та обслуговування.

1.1.3 Windows 10 Mobile

Windows 10 Mobile, розроблене корпорацією Microsoft, представляє собою мобільну операційну систему, яка входить до сімейства Windows 10 [6]. Навіть при тому, що ця платформа не завоювала таку широку популярність серед користувачів, як Android або iOS, вона відзначається уніфікацією з іншими продуктами Microsoft і можливостями для розробників.

Основні характеристики Windows 10 Mobile:

- Інтеграція в одну екосистему Windows. Windows 10 Mobile вплетена в єдину екосистему Windows 10, що дозволяє використовувати уніфіковані сервіси та додатки на різних пристроях, включаючи комп'ютери,

планшети та смартфони. Це створює зручний та однорідний досвід для користувачів.

- Універсальні додатки (Universal Apps). Концепція універсальних додатків дозволяє створювати один додаток, який працюватиме на різних пристроях з Windows 10. Це спрощує розробку та підтримку додатків для Windows 10 Mobile, дозволяючи їм адаптуватися до різних екранних розмірів та формфакторів.
- Інтеграція з Microsoft Office та іншими сервісами. Windows 10 Mobile інтегрується з ключовими сервісами Microsoft, такими як Office 365 та OneDrive. Це робить його привабливим для користувачів, які активно використовують продукти Microsoft у своїй роботі.
- Забезпечення високого рівня безпеки. Windows 10 Mobile славиться своєю високою системою безпеки, що особливо важливо в корпоративному середовищі. Інтеграція зі службами Windows Defender та BitLocker сприяє захисту користувачів від загроз та втрати конфіденційної інформації.
- Однорідний досвід користувача. Інтерфейс Windows 10 Mobile має багатий та зручний дизайн, що надає користувачам спільний досвід при використанні різних пристроїв у єдиній операційній системі Windows 10.

Windows 10 Mobile використовує своє власне інтегроване середовище розробки, включаючи Visual Studio та мову програмування C#. Це надає зручний інструментарій для створення додатків для цієї платформи. Однак, відносно невелика кількість пристроїв, що використовують Windows 10 Mobile, ставить перед розробниками завдання забезпечення сумісності та ефективності на обмеженому колі пристроїв [6].

Архітектура Windows 10 Mobile базується на принципі "універсального ядра" та складається з таких ключових компонентів:

- Універсальне Ядро (Universal Kernel): Визначає основні функції та принципи роботи операційної системи на різних пристроях, забезпечуючи узгодженість між Windows 10 та Windows 10 Mobile.
- Інтегровані засоби розробки (Integrated Development Tools): Включають у себе такі інструменти, як Visual Studio, що дозволяють створювати уніфіковані додатки для різних пристроїв.
- Архітектурні рішення, використані в Windows 10 Mobile, дозволяють створювати додатки, які максимально адаптовані до різних пристроїв і забезпечують спільний досвід користувача на платформах Windows.

Переваги Windows 10 Mobile [6]:

1. Інтеграція зі стандартними програмами та сервісами Windows 10.
2. Універсальна платформа, яка дозволяє розробляти застосунки для різних типів пристроїв (смартфонів, планшетів, ПК).
3. Підтримка Microsoft-екосистеми та інструментів розробки.

Недоліки Windows 10 Mobile:

1. Низька популярність та обмежений ринок користувачів.
2. Обмежена підтримка сторонніх розробників та додатків.
3. Невиразна стратегія компанії Microsoft стосовно мобільних пристроїв.

Windows 10 Mobile є мобільною операційною системою, яка пропонує розширені можливості та інтеграцію зі складовими Windows 10. За допомогою універсальних додатків, користувацького інтерфейсу Live Tiles та інструментів розробки, таких як Universal Windows Platform та Visual Studio, можна ефективно створювати та підтримувати додатки для Windows 10 Mobile.

1.1.4 Microsoft Launcher

Microsoft Launcher – це застосунок для особистого налаштування інтерфейсу на базі операційної системи Android, розроблений компанією Microsoft [7]. Цей лаунчер став прикладом того, як корпорації можуть взаємодіяти з іншими мобільними екосистемами, надаючи користувачам можливість інтегрувати свої улюблені сервіси Microsoft в екосистему Android.

Microsoft Launcher відрізняється рядом важливих особливостей, що роблять його привабливим для користувачів, які використовують екосистему Microsoft [7]:

- Інтеграція зі службами Microsoft: Лаунчер забезпечує повну інтеграцію з популярними сервісами Microsoft, такими як Outlook, Microsoft 365, Skype, OneDrive та інші. Це дозволяє користувачам зручно керувати своїми робочими та особистими завданнями.
- Настроєння стартового екрану: Користувачі можуть персоналізувати свій стартовий екран, використовуючи різноманітні теми, шпалери та плитки, що дозволяє створити унікальний інтерфейс під свої власні потреби та вподобання.
- Контекстне меню: Microsoft Launcher надає доступ до контекстного меню, яке дозволяє швидко отримувати важливу інформацію та взаємодіяти з додатками безпосередньо зі стартового екрану. Функція "Останні файли": Даний лаунчер включає функцію "Останні файли", яка допомагає користувачам легко знаходити та отримувати доступ до недавно використаних файлів, що зберігаються в OneDrive.

Microsoft Launcher базується на ідеї високої настроєності, де користувач може адаптувати свій стартовий екран, використовуючи різноманітні теми, шпалери та плитки. Такий підхід створює унікальний інтерфейс, але одночасно вносить виклик для розробників, оскільки їхні додатки повинні надійно функціонувати в різноманітних конфігураціях [7].

Архітектура та інструменти розробки Microsoft Launcher:

- Архітектура Microsoft Launcher базується на стандартній архітектурі Android та включає такі ключові компоненти:
- Ядро Android: Microsoft Launcher використовує базове ядро Android для забезпечення операційної системи та роботи з апаратними компонентами пристрою.
- Прикладний програмний інтерфейс (API): Microsoft Launcher використовує API Android для взаємодії з різними функціями та

можливостями пристрою, такими як камера, геолокація, повідомлення та інші.

- Microsoft Graph API: Для інтеграції з сервісами та додатками Microsoft, Microsoft Launcher використовує Microsoft Graph API. Цей API надає доступ до різноманітних сервісів Microsoft, таких як Outlook, OneDrive, Calendar і багатьох інших.

Переваги Microsoft Launcher:

1. Інтеграція з екосистемою Microsoft та сервісами.
2. Налаштування та персоналізація інтерфейсу користувача.
3. Зручний доступ до файлів та додатків на пристрої.

Недоліки Microsoft Launcher:

1. Обмежена функціональність та можливості порівняно з іншими мобільними платформами.
2. Невисока популярність та ринкова частка.

1.2 Типи мобільних застосунків

Одним із важливих аспектів при розробці мобільних додатків є вибір технології, яка буде використовуватись для створення додатку. У другому розділі ми розглянемо типи мобільних застосунків, що існують на ринку. Нативні (Native) застосунки, веб-застосунки та гібридні застосунки представляють собою різні підходи до розробки та використання мобільних застосунків. Аналіз цих типів дозволить визначити, які підходи є оптимальними для визначених завдань та цільових аудиторій.

1.2.1 Нативні (Native)

Нативні (Native) застосунки є парадигмою розробки, де кожна програма спеціально адаптована для конкретної мобільної платформи та використовує мови програмування та інструментарій, які є специфічними для цієї платфо-

рми [8]. У своїй основі, нативні застосунки прагнуть максимально використовувати потенціал конкретної операційної системи та пристрою, щоб забезпечити високий рівень продуктивності та оптимізації.

Ще однією з ключових особливостей нативних застосунків є їхня можливість повного використання всіх функцій та API, які пропонує певна платформа. Це означає, що розробники можуть максимально ефективно використовувати унікальні можливості та функціонал певної операційної системи, забезпечуючи глибоку інтеграцію та оптимальний досвід користувача.

Проте, процес розробки нативних застосунків може стати викликом через необхідність створення окремого коду для кожної платформи. Для платформи Android використовується мова програмування Java або Kotlin, в той час як для iOS - мова Swift або Objective-C. Це може викликати значний обсяг роботи та зусиль, а також призводити до дублювання функціоналу.

Тестування нативних застосунків також має свої особливості. З урахуванням різноманіття апаратного забезпечення та версій операційних систем Android та iOS, тестування стає більш важким завданням. Використання інструментів автоматизованого тестування, які спеціально підтримують нативні застосунки, допомагає розробникам та тестувальникам впевнитися у стабільності та сумісності їхніх додатків з різними пристроями [8].

Переваги нативних застосунків:

1. Найкраща продуктивність: Нативні застосунки зазвичай мають найкращу продуктивність та швидкість роботи. Вони оптимізовані для конкретної платформи та можуть використовувати всі можливості, які надаються цією платформою.
2. Легка інтеграція з платформою: Нативні додатки легко інтегруються з функціями та сервісами, що надаються операційною системою. Це дозволяє використовувати всі можливості пристрою та взаємодіяти з іншими додатками.
3. Кращий доступ до апаратних компонентів: Нативні застосунки мають прямий доступ до апаратних компонентів пристрою, таких як камера,

геолокація, сенсори та інші. Це дає можливість створювати додатки з більш розширеним функціоналом.

Недоліки нативних застосунків:

1. Більш складна розробка: Розробка нативних застосунків вимагає володіння специфічними мовами програмування та інструментами для кожної платформи. Це може збільшити складність та тривалість розробки.
2. Високі витрати на розробку: Розробка нативних застосунків може бути витратною, оскільки потрібно володіти різними навичками та інструментами для кожної платформи.
3. Обмежена мультиплатформенність: Нативні додатки розробляються для конкретної платформи, тому вимагають окремої розробки для кожної платформи. Це може бути проблемою, якщо вам потрібно підтримувати додаток на різних платформах.

Нативні застосунки мають свої власні переваги та недоліки. Вони надають найкращу продуктивність, легку інтеграцію з платформою та доступ до апаратних компонентів пристрою. Однак, вони вимагають більшої складності та витрат на розробку, а також обмеженої мультиплатформності.

1.2.2 Веб-застосунки

Веб-застосунки є категорією мобільних додатків, яка базується на використанні веб-технологій та стандартів. Вони забезпечують універсальний доступ до функціоналу незалежно від платформи через веб-браузер [9]. Цей тип додатків надає більшу гнучкість, оскільки вони можуть використовувати відому та стабільну інфраструктуру веб-розробки.

Веб-застосунки - це програми, що працюють в Інтернеті та доступні користувачам через веб-браузер. Вони базуються на веб-технологіях, таких як HTML, CSS та JavaScript, і надають можливість взаємодіяти зі вмістом та функціями через веб-інтерфейс [9].

Веб-застосунки дозволяють створювати додатки, які можуть працювати на різних пристроях та платформах, враховуючи лише сумісність з веб-браузерами. Це знижує витрати на розробку та управління кодовою базою, оскільки більшість логіки та ресурсів розташовані на веб-сервері, а не на самому пристрої.

Однак, незважаючи на багато переваг, веб-застосунки можуть стикається з викликами, пов'язаними із залежністю від швидкості й надійності інтернет-з'єднання. Доступ до певних функцій пристрою, таких як камера або датчики, може бути обмеженим в порівнянні з нативними застосунками.

Тестування веб-застосунків має свої особливості, оскільки потрібно враховувати різноманіття веб-браузерів на різних пристроях та платформах. Використання інструментів автоматизованого тестування, які підтримують веб-технології, є ключовим для забезпечення стабільності та сумісності веб-застосунків [9].

Переваги веб-застосунків:

1. Кросплатформеність: Веб-застосунки можуть працювати на різних операційних системах та пристроях, що забезпечує широку доступність та зручність для користувачів.
2. Зручність оновлень: Оновлення веб-застосунків відбуваються на сервері, що дозволяє користувачам отримувати оновлення автоматично без необхідності вручну встановлювати нову версію.
3. Гнучкість: Веб-застосунки можуть бути легко змінені та адаптовані до нових вимог та потреб користувачів.

Недоліки веб-застосунків:

1. Залежність від Інтернету: Для роботи веб-застосунків необхідне постійне підключення до Інтернету. Відсутність зв'язку з мережею може обмежувати доступність та функціональність додатка.
2. Обмежені можливості роботи в автономному режимі: Веб-застосунки зазвичай не мають повноцінного режиму роботи в автономному режимі,

що може бути не зручним для користувачів, які потребують доступу до даних та функцій без доступу до Інтернету.

3. Обмежена функціональність: Веб-застосунки можуть мати обмежені можливості порівняно з нативними додатками, оскільки вони залежать від функцій, які доступні через веб-браузер та веб-технології.

Веб-застосунки є важливою складовою сучасного цифрового світу, забезпечуючи користувачам доступ до функцій та сервісів через веб-браузер. Вони мають свої переваги, такі як кросплатформеність, зручність оновлень та гнучкість, але також мають недоліки, такі як залежність від Інтернету та обмежені можливості роботи в автономному режимі.

1.2.3 Гібридні застосунки

Гібридні застосунки представляють собою універсальний підхід до розробки мобільних додатків, який поєднує в собі елементи як нативних (Native), так і веб-застосунків [10]. Цей підхід дозволяє розробникам використовувати спільний код для створення додатків для різних платформ, зменшуючи тим самим час і витрати на розробку.

Основною перевагою гібридних застосунків є можливість використання веб-технологій, таких як HTML, CSS та JavaScript, що робить їх більш доступними для більшості розробників. Зокрема, фреймворки, такі як React Native, Flutter та Ionic, надають можливість використовувати єдиний код для створення додатків, які можуть працювати як на Android, так і на iOS.

Однак гібридні застосунки не завжди досягають такого ж рівня продуктивності та швидкодії, як нативні застосунки, оскільки вони обмежені функціональністю веб-браузера та вказаними фреймворками. Деякі функції та можливості платформи можуть бути недоступними або обмеженими через використання гібридного підходу.

Тестування гібридних застосунків також вимагає специфічних підходів, оскільки вони поєднують елементи різних технологій. Використання інстру-

ментів автоматизованого тестування, які розуміють особливості гібридних додатків, стає ключовим для забезпечення їхньої стабільності та сумісності з різними пристроями та операційними системами [10].

Переваги гібридних застосунків:

1. Кросплатформеність: Гібридні застосунки дозволяють створювати один застосунок, який працюватиме на різних платформах, що зменшує час та затрати на розробку та підтримку.
2. Швидкість розробки: Використання веб-технологій дозволяє швидко створювати користувацький інтерфейс та логіку застосунків.
3. Оновлення в реальному часі: Гібридні застосунки можуть оновлюватись в реальному часі, без необхідності встановлення оновлень з магазинів додатків. Це дозволяє швидко внести зміни та виправити помилки без затримок.

Недоліки гібридних застосунків:

1. Обмежена функціональність: Гібридні застосунки можуть мати обмежену функціональність порівняно з нативними застосунками, оскільки вони залежать від доступних функцій фреймворків.
2. Потужність та продуктивність: У порівнянні з нативними застосунками, гібридні застосунки можуть мати обмежену продуктивність та швидкість реакції, особливо при роботі з важкими завданнями або використанні графічно-інтенсивних операцій.
3. Залежність від фреймворків: Розробка гібридних застосунків вимагає використання певних фреймворків, що може обмежувати розробників у виборі технологій та можливості підтримки.

Гібридні застосунки є ефективним рішенням для кросплатформеної розробки мобільних додатків, забезпечуючи швидкість розробки та кросплатформену сумісність. Вони дозволяють використовувати веб-технології та мати доступ до нативних функцій пристроїв. Однак, вони також мають обмежену функціональність та можуть бути менш продуктивними порівняно з нативними застосунками.

1.3 Висновки до розділу 1

У даному розділі ми детально розглянули характеристики сучасних мобільних платформ, визначивши основні особливості кожної з них. Огляд сучасних мобільних платформ виявив, що Android, iOS та їхні варіації, такі як Windows 10 Mobile та Microsoft Launcher, є ключовими гравцями на ринку, кожен з яких має свої унікальні риси та підходи до розробки мобільних застосунків.

Ми розглянули різні типи мобільних застосунків, такі як нативні (Native), веб-застосунки та гібридні застосунки, визначивши їхні особливості та переваги. Це важливо при виборі стратегії розробки в залежності від конкретного проекту та його вимог.

У цьому висновку я можу підкреслити важливість правильного вибору платформи та типу застосунку, залежно від конкретних потреб проекту.

РОЗДІЛ 2 АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ

2.1 Автоматизоване тестування мобільних застосунків

Тестування мобільних застосунків є критичним етапом у процесі їх розробки та випуску на ринок. Традиційно тестування проводилося вручну, де тестувальники виконували різні сценарії та перевіряли правильність роботи додатків. Однак, з появою автоматизованих тестувальних інструментів, таких як фреймворки для автоматизації тестування, процес тестування мобільних застосунків значно полегшився і став більш ефективним.

Автоматизоване тестування мобільних застосунків має свої переваги та недоліки порівняно з ручним тестуванням. Хоча автоматизація дозволяє зекономити час, збільшити покриття тестування та підвищити точність, вона також вимагає додаткових витрат на розробку та підтримку тестових скриптів. Крім того, вона може пропустити нові типи помилок та вимагати постійного оновлення тестових скриптів. Враховуючи ці фактори, тестувальна команда повинна розглянути переваги та недоліки обох підходів та визначити найкращу стратегію для тестування мобільних застосунків відповідно до вимог проекту та ресурсів, які вона має [11].

2.1.1 Обґрунтування важливості автоматизації у контексті мобільних додатків

У сучасному цифровому світі, де мобільні додатки відіграють важливу роль у повсякденному житті користувачів, важливість автоматизованого тестування стає невідомою. Мобільні застосунки швидко розвиваються і оновлюються, що ставить під сумнів їхню стабільність та надійність.

Автоматизація тестування в контексті мобільних додатків дозволяє забезпечити швидке виявлення та виправлення помилок, що стає критичною умовою в конкурентному світі розробки програмного забезпечення. Швидкі

цикли розробки та випуску вимагають ефективних засобів тестування, і автоматизація надає можливість автоматично виконувати тести на різних пристроях та операційних системах.

Зменшення ручного тестування і впровадження автоматизації дозволяє збільшити обсяг тестів та покриття функціональності. Це важливо, оскільки мобільні застосунки часто мають різноманітні конфігурації, що вимагає широкого спектру тестів для переконливості в їхній працездатності.

Крім цього, автоматизоване тестування дозволяє ефективно впроваджувати тестові скрипти в процес розробки, забезпечуючи невідкладний зворотний зв'язок розробникам. Це сприяє виправленню помилок на ранніх етапах розробки, що робить процес більш ефективним та економічним.

Враховуючи динаміку мобільних технологій, автоматизоване тестування визначається не просто як процес, але як ключовий елемент стратегії розробки мобільних додатків, який сприяє їхньому успішному впровадженню та функціонуванню.

2.1.2 Переваги та обмеження автоматизованого тестування мобільних застосунків

Автоматизоване тестування мобільних додатків вносить значний вклад у розробку стабільних та високоякісних мобільних застосунків. Однак, як і будь-яка технологія, воно має свої переваги та обмеження [11].

Переваги:

1. Швидкість виконання тестів. Автоматизовані тести можуть бути виконані значно швидше, порівняно з ручним тестуванням. Це дозволяє розробникам швидко перевірити стабільність нових змін та вносити виправлення перед релізом.
2. Повторюваність та стабільність. Автоматизовані тести гарантують повторюваність виконання, що особливо важливо при тестуванні різних конфігурацій та сценаріїв. Це сприяє стабільності та надійності додатків.

3. Великий обсяг тестів. Завдяки автоматизації можна легко охоплювати великі обсяги тестів, включаючи різноманітні сценарії та конфігурації, що забезпечує високу якість продукту.
4. Виявлення дефектів на ранніх етапах. Автоматизовані тести дозволяють виявляти дефекти на ранніх етапах розробки, забезпечуючи швидше виправлення та відсутність дефектів у кінцевому продукті.
5. Ефективне тестування різних пристроїв та платформ. Автоматизація дозволяє легко переносити тести на різні пристрої та платформи, забезпечуючи адаптовані тести для різних умов використання.

Обмеження:

1. Вартість впровадження. Впровадження автоматизованих тестів може вимагати значних витрат на навчання персоналу та на вибір та налаштування відповідних інструментів.
2. Складність тестування UI/UX. Деякі аспекти тестування користувацького інтерфейсу (UI) та користувацького досвіду (UX) можуть бути складні для автоматизації, такі як перевірка візуального оформлення чи взаємодії з графічними елементами.
3. Залежність від інструментів. Успішність автоматизованих тестів часто залежить від якості та правильного вибору інструментів. Неправильний вибір може призвести до неефективності тестування.
4. Обмеження в тестуванні винятків. Автоматизовані тести можуть бути менш ефективними у виявленні складних сценаріїв або винятків, які можуть виникнути в реальних умовах використання.
5. Потреба в постійному оновленні. Зміни в мобільних платформах та інструментах розробки можуть вимагати постійного оновлення та адаптації автоматизованих тестів.

Незважаючи на обмеження, автоматизоване тестування мобільних застосунків залишається важливою складовою процесу розробки, яка забезпечує високу якість та ефективність продукту.

2.1.3 Вибір інструментів для автоматизованого тестування та їхні особливості

Вибір інструментів для автоматизованого тестування мобільних додатків є стратегічним завданням, оскільки від нього залежить ефективність та успішність процесу тестування. Основні вимоги до інструментів включають підтримку різних мобільних платформ, можливість автоматизації різних типів додатків (нативних, веб-застосунків, гібридних), а також простоту використання та налаштування.

На ринку існує ряд інструментів для автоматизованого тестування мобільних додатків, кожен з яких має свої особливості [12]. Деякі з них, такі як Selenium та WebDriver, визначаються своєю універсальністю та підтримкою кросплатформного тестування. Інші, такі як TestComplete або XCUITest, можуть спеціалізуватися на певних платформах та мовах програмування.

Основна увага також приділяється інтеграції з іншими інструментами розробки та CI/CD системами. Зручна інтеграція може значно полегшити процес виконання автоматизованих тестів в різних етапах розробки [12].

У виборі інструментів для автоматизованого тестування також важливо враховувати технічні вимоги проекту, такі як підтримка різних версій ОС, можливість використання хмарних сервісів для тестування реальних пристроїв, а також можливість паралельного виконання тестів для забезпечення швидкості та ефективності тестування [12].

У кінці, вибір інструментів повинен відповідати конкретним потребам та характеристикам проекту, забезпечуючи оптимальну комбінацію функціональності та продуктивності для ефективного автоматизованого тестування мобільних додатків.

2.1.4 Підготовка тестового середовища для мобільних додатків

Ефективне тестування мобільних додатків залежить від належно підготованого тестового середовища, яке відіграє роль у виявленні помилок та забезпеченні високої якості продукту. Основні аспекти цього процесу включають репрезентативність, автоматизацію, безпеку та легкість підтримки.

Репрезентативність тестового середовища означає його здатність точно відображати умови реального використання мобільного додатку. Розробники повинні створити різноманітне середовище, що відображає різні моделі пристроїв, версії операційних систем, мережеві умови та інші фактори, які можуть впливати на функціонування додатку.

Автоматизація підготовки тестового середовища виявляється особливо важливою, забезпечуючи швидке і безпомилкове встановлення необхідних конфігурацій. Це робить процес тестування більш гнучким та адаптованим до швидко змінюваних вимог.

Безпечність тестового середовища має вирішальне значення. Забезпечення конфіденційності та ізоляції тестового середовища дозволяє виконувати тести на реальних даних без ризику витоку інформації чи негативного впливу на продакшн-середовище.

Розглянута оптимальна стратегія підготовки тестового середовища для мобільних додатків, в рахується аспект ефективності, стабільності та легкості впровадження автоматизованих тестів.

2.2 Загальні вимоги до засобів кросплатформного тестування застосунків різних типів

Технології мобільних застосунків дуже швидко розвиваються, автоматизоване тестування є і буде важливою ланкою для забезпечення високої якості програмного забезпечення. Однак різноманітність платформ та типів додатків створює виклик у виборі засобів тестування, які були б ефективними та універсальними. У цьому розділі ми розглянемо загальні вимоги до засобів

кросплатформного тестування, які дозволяють забезпечити високий рівень покриття для застосунків різних типів та операційних систем [14].

2.2.1 Підтримка різних операційних систем

Підтримка різних операційних систем, таких як Android і iOS, є важливою для максимізації охоплення аудиторії.

Оскільки користувачі мають схильність використовувати різні мобільні пристрої з різними операційними системами, важливо, щоб засоби автоматизованого тестування були здатні працювати ефективно на всіх популярних платформах. Це включає в себе підтримку останніх версій операційних систем, а також можливість взаємодії з ранніми версіями для забезпечення повноцінного покриття [15].

При розгляді підтримки різних операційних систем слід враховувати не тільки відмінності у версіях, але й специфічні особливості кожної платформи. Такий підхід дозволяє гарантувати, що мобільний додаток функціонує оптимально та надійно на будь-якому пристрої, незалежно від операційної системи. Засоби кросплатформного тестування, які забезпечують широку підтримку операційних систем, стають невід'ємною складовою в процесі розробки та підтримки мобільних застосунків.

2.2.2 Підтримка різних типів застосунків (Native, ВЕФ, Гібридні)

В сучасному ландшафті мобільних додатків різноманітність типів застосунків потрібна для відповіді на різні потреби користувачів. Засоби автоматизованого тестування повинні бути гнучкими та адаптивними для взаємодії з різними типами застосунків, щоб забезпечити повноцінне покриття тестування.

Застосунки Native, розроблені для конкретної операційної системи, вимагають особливого підходу та підтримки у засобах тестування. Веб-застосунки, які працюють в браузері, також потребують відповідної підтримки для

ефективного виконання тестів. Гібридні застосунки, поєднуючи риси інших двох типів, створюють додаткові виклики та вимоги для тестування [16].

Забезпечення підтримки для різних типів застосунків важливо не лише з точки зору покриття функціональності, але і для забезпечення оптимальної продуктивності та надійності. Адаптовані засоби тестування мають можливість ефективно взаємодіяти із специфічними особливостями кожного типу додатка, щоб забезпечити якість та відповідати вимогам розробників та користувачів [16].

2.2.3 Підтримка різних мов програмування

Широка різноманітність мов програмування є стандартом, оскільки розробники вибирають інструменти, що відповідають їхнім потребам та преференціям. Засоби автоматизованого тестування повинні бути гнучкими та адаптованими до різних мов, щоб забезпечити повну інтеграцію з проектами, розробленими на різних технологічних стеках.

Підтримка різних мов програмування важлива для того, щоб забезпечити, що тестові скрипти можуть бути інтегровані безперешкодно в будь-який проект. Це особливо актуально в умовах розробки гібридних та кросплатформових додатків, де використовуються різні технології та мови програмування для різних частин додатку [17].

Забезпечення підтримки різних мов програмування також дозволяє використовувати свої улюблені мови та інструменти, що сприяє високій продуктивності та ефективності в процесі автоматизованого тестування мобільних застосунків. Гнучкість та розширюваність засобів тестування в цьому контексті визначають їхню ефективність та відповідність сучасним стандартам розробки.

2.2.4 Підтримка різних інструментів розробки

Під час розробки мобільних додатків, розробники використовують різні інструменти, зокрема різні IDE (Integrated Development Environment), мови

програмування та фреймворки. Засоби автоматизованого тестування повинні бути адаптованими для різних комбінацій інструментів розробки, що використовуються в проекті.

Важливим аспектом є підтримка найпоширеніших мов програмування, таких як Java для Android-розробки та Swift для iOS. Це дозволяє обирати найзручніший інструментарій для свого проекту, не обмежуючи можливості автоматизованого тестування.

Застосунки для мобільних пристроїв можуть бути розроблені за допомогою різних фреймворків, таких як React Native, Flutter або Xamarin. Підтримка цих фреймворків в засобах автоматизованого тестування дозволяє гармонійно взаємодіяти з різними технологіями розробки та підтримувати відкритий підхід до вибору інструментів.

Широка сумісність з інструментами розробки сприяє створенню єдиного екосистемного підходу до автоматизованого тестування, де можливо повністю зосередитися на ефективній розробці, знаючи, що тестування може бути легко інтегроване в їхнє середовище без додаткових труднощів.

2.3 Ідентифікація елементів інтерфейсу

У розробці мобільних застосунків та автоматизованому тестуванні однією з важливих задач є ідентифікація елементів інтерфейсу, що дозволяє здійснювати взаємодію з ними та виконувати різні дії для тестування.

Ідентифікація елементів інтерфейсу в мобільних застосунках відноситься до процесу знаходження та визначення конкретних елементів інтерфейсу, таких як кнопки, текстові поля, списки, картинки тощо, з метою їх подальшої взаємодії та тестування [18]. Ідентифікація елементів інтерфейсу відбувається за допомогою унікальних ідентифікаторів або властивостей, які дозволяють звертатися до цих елементів під час автоматизованого тестування.

2.3.1 Види ідентифікації елементів інтерфейсу

Ідентифікація за допомогою ідентифікаторів: деякі елементи інтерфейсу мають унікальні ідентифікатори, які дозволяють прямо вказати на потрібний елемент. Це можуть бути атрибути, надані розробниками, такі як ідентифікатори, назви або теги, які використовуються для звернення до елементів.

Ідентифікація за допомогою css-селекторів: у деяких випадках, особливо при розробці веб-додатків, можна використовувати css-селектори для ідентифікації елементів інтерфейсу [18]. Css-селектори вказують на конкретний елемент або групу елементів на основі їх структури, класів, ідентифікаторів тощо.

Ідентифікація за допомогою шляху елемента (xpath): xpath є мовою запитів, яка дозволяє точно вказати шлях до елемента в документі або на вебсторінці. Використання xpath дає змогу знаходити елементи за їх позицією в дереві dom (document object model).

Ідентифікація за допомогою властивостей елементів: деякі елементи інтерфейсу можуть бути ідентифіковані за допомогою їх властивостей, таких як текст, колір, розмір, координати на екрані тощо. Використання цих властивостей дозволяє знаходити та взаємодіяти з елементами [18].

Ідентифікація елементів інтерфейсу є одним з етапом в автоматизованому тестуванні мобільних застосунків. Визначення ідентифікації елементів інтерфейсу полягає у знаходженні та визначенні елементів за допомогою унікальних ідентифікаторів, css-селекторів, xpath або властивостей елементів. Вибір певного методу ідентифікації залежить від конкретного випадку та вимог тестування.

2.3.2 Методи ідентифікації елементів інтерфейсу

Методи ідентифікації елементів інтерфейсу: ідентифікація за ідентифікатором, ідентифікація за текстом, ідентифікація за координатами, ідентифікація за класом або типом.

У сучасному світі розробки мобільних застосунків виникає необхідність в автоматизованому тестуванні, яке дозволяє виявляти помилки та забезпечувати якість програмного продукту. Одним з ключових аспектів автоматизованого тестування є ідентифікація елементів інтерфейсу, що включається в мобільних застосунках.

Ідентифікація за ідентифікатором: цей метод передбачає присвоєння унікальних ідентифікаторів елементам інтерфейсу. Кожен елемент має свій унікальний ідентифікатор, який можна використовувати для пошуку та взаємодії з ним [18]. Цей метод зручний та ефективний, оскільки дозволяє однозначно ідентифікувати елементи.

Ідентифікація за текстом: у цьому методі елементи інтерфейсу ідентифікуються за їх текстовим вмістом. Наприклад, кнопка з написом "вхід" може бути ідентифікована за допомогою тексту "вхід" [18]. Цей метод широко використовується, оскільки текстовий вміст елементів часто є унікальним та стабільним.

Ідентифікація за координатами: в цьому методі елементи інтерфейсу ідентифікуються за їх позицією на екрані, використовуючи координати x та y. Цей метод менш стабільний, оскільки можуть виникати проблеми зміни розмірів екрану та розташування елементів [18]. Проте, в деяких випадках цей метод може бути використаний, особливо якщо інші методи не є доступними.

Ідентифікація за класом або типом: у цьому методі елементи інтерфейсу ідентифікуються за їх класом або типом. Наприклад, кнопка може бути ідентифікована за допомогою класу "button" або типу "button" [18]. Цей метод забезпечує швидку та просту ідентифікацію елементів, але може бути менш стабільним, оскільки класи та типи елементів можуть змінюватись у майбутньому.

Порівняння методів ідентифікації:

Ефективність: ідентифікація за ідентифікатором та за класом або типом є швидкими та ефективними методами, оскільки вони дозволяють прямий доступ до елементів. Ідентифікація за текстом та за координатами можуть бути

трохи повільнішими, оскільки вони вимагають додаткових обчислень або пошуку.

Стабільність: ідентифікація за ідентифікатором є найстабільнішим методом, оскільки ідентифікатори є унікальними та малоймовірними до змін. Ідентифікація за текстом та за координатами можуть бути менш стабільними, оскільки текст може змінюватися, а координати можуть залежати від розміру екрану або орієнтації [18].

Підтримка різних типів елементів: усі методи ідентифікації підтримують різні типи елементів інтерфейсу, такі як кнопки, тексти, списки тощо [18]. Однак, ефективність та стабільність методів може варіюватися в залежності від типу елемента.

Ми розглянули різні методи ідентифікації елементів інтерфейсу мобільних додатків, включаючи ідентифікацію за ідентифікатором, текстом, координатами та класом або типом. Кожен метод має свої переваги та обмеження, і їх вибір залежить від конкретного контексту тестування. Важливо враховувати ефективність, стабільність та підтримку різних типів елементів при виборі методу ідентифікації для автоматизованого тестування мобільних застосунків.

2.4 Тестування користувацького введення

Розробка мобільних застосунків не зможе нехтувати важливим етапом, таким як тестування користувацького введення. Коректність та ефективність обробки користувацького введення впливають на функціональність, надійність та задоволення користувачів. У цьому пункті дипломної роботи ми розглянемо визначення тестування користувацького введення та порівняємо різні методи тестування, зокрема тестування введення тексту, вибору елементів зі списку та натискання на кнопки.

Тестування користувацького введення включає перевірку правильності та відповідності обробки введених даних користувачем. Це охоплює різні типи введення, такі як текст, вибір зі списку, натискання на кнопки тощо. Метою

тестування користувацького введення є виявлення помилок, некоректних реакцій та забезпечення зручності взаємодії користувача з додатком [19].

Тестування введення тексту спрямоване на перевірку коректності обробки текстових даних, введених користувачем. У цьому тестуванні перевіряються такі аспекти, як правильність збереження та відображення введеного тексту, обробка спеціальних символів, перевірка наявності обов'язкових полів та обробка помилок під час введення.

Тестування вибору елементів зі списку перевіряє правильність обробки вибору користувачем пунктів зі списку [19]. Це може включати перевірку правильності відображення елементів списку, можливість вибору одного або кількох пунктів, обробка зміни вибору пунктів та коректність збереження вибраних значень.

Тестування натискання на кнопки перевіряє правильність реакції додатку на натискання користувачем різних кнопок. У цьому тестуванні перевіряється виконання очікуваних дій після натискання на кнопки, наприклад, перехід на іншу сторінку, виклик функції або збереження даних.

Існує кілька методів тестування користувацького введення, включаючи ручне тестування, автоматизоване тестування та використання фреймворків тестування. Кожен метод має свої переваги та недоліки, які потрібно враховувати при виборі оптимального підходу.

Ручне тестування: Цей метод передбачає виконання тестових сценаріїв вручну спеціально натренованими тестувальниками. Він дозволяє отримати детальні результати та виявити незрозумілі аспекти взаємодії з додатком [19]. Однак, він вимагає багато ресурсів, часу та праці.

Автоматизоване тестування: Автоматизоване тестування використовує спеціальні інструменти та скрипти для виконання тестових сценаріїв. Воно дозволяє автоматизувати тестові процеси, зменшити час тестування та забезпечити більшу точність [19]. Однак, його ефективність залежить від якості розроблених скриптів та може бути обмежена в разі змін у користувацькому інтерфейсі.

Використання фреймворків тестування: Фреймворки тестування надають розробникам можливості для автоматизації тестування користувацького введення. Вони забезпечують API та інструменти для взаємодії з елементами інтерфейсу, виконання дій та перевірки очікуваних результатів.

Тестування користувацького введення є важливим етапом в розробці мобільних застосунків. Тестування введення тексту, вибору елементів зі списку та натискання на кнопки допомагають перевірити правильність обробки користувацького введення та забезпечити зручність взаємодії. Вибір методу тестування, такого як ручне тестування, автоматизоване тестування або використання фреймворків тестування, залежить від вимог проекту, ресурсів та потреб розробників.

2.5 Тестування датчиків

Датчики є невіддільною частиною мобільних пристроїв, і їх правильне функціонування є критичним для багатьох додатків, для забезпечення взаємодії між користувачем та пристроєм. Тестування датчиків є важливою складовою процесу розробки мобільних додатків, оскільки воно допомагає перевірити правильність роботи датчиків і їх відповідність заданим специфікаціям.

Тестування датчиків є процесом перевірки функціональності та точності роботи датчиків у мобільних пристроях. Це включає проведення різних тестів для перевірки, чи працюють датчики правильно, чи вони забезпечують вірні дані та чи вони відповідають заданим критеріям якості. Тестування датчиків допомагає виявити можливі проблеми та недоліки, що допомагає розробникам вдосконалювати якість мобільних додатків [20].

Акселерометр є одним з основних датчиків, що вимірює прискорення пристрою у трьох основних вимірах: вперед-назад, вліво-вправо та вгору-вниз. Під час тестування акселерометра, необхідно перевірити його точність вимірювання, реакцію на різні рухи пристрою та правильність передачі даних додаткам. Тестування акселерометра може включати реалізацію різних сценаріїв

руху, замір точності вимірювання та перевірку збереження даних в межах встановлених меж.

Геолокаційний датчик визначає місцезнаходження пристрою на основі сигналів gps, wi-fi або мобільної мережі. Під час тестування геолокації, необхідно перевірити точність визначення місцезнаходження, реакцію на зміну місцеположення та правильність передачі даних додаткам [20]. Тестування геолокації може включати створення симуляційних сценаріїв руху, перевірку збереження координат, визначення точності та перевірку сумісності з різними джерелами місцезнаходження.

Камера є у багатьох мобільних пристроях і використовується для фотографування, відеозйомки та відеодзвінків. Під час тестування камери, необхідно перевірити правильність збільшення, фокусування та налаштування експозиції. Тестування камери може включати створення тестових сценаріїв зйомки, перевірку якості зображень та перевірку роботи додатків, які використовують камеру [20].

Тестування датчиків допомагає перевірити функціональність та точність роботи датчиків. Тестування акселерометра, геолокації та камери є ключовими аспектами тестування датчиків у мобільних пристроях.

2.6 Висновок до розділу 2

У цьому розділі було проведено дослідження технологій автоматизованого тестування мобільних застосунків. Дослідження показало, що існують різні види тестування Android- застосунків, такі як юніт-тестування, функціональне тестування, тестування інтеграції та тестування взаємодії з базою даних. Крім того, виявлено загальні вимоги до засобів кросплатформного тестування, такі як підтримка різних платформ, типів застосунків, легкість використання та функціонал для автоматизованого тестування.

Також було досліджено питання ідентифікації елементів інтерфейсу та різні методи ідентифікації, такі як ідентифікація за ідентифікатором, текстом,

координатами, класом або типом. Це є важливим аспектом автоматизованого тестування, оскільки правильна ідентифікація елементів інтерфейсу є основою успішного тестування.

Крім того, було розглянуто тестування користувацького введення, таке як введення тексту, вибір елементів зі списку та натискання на кнопки. А також тестування датчиків, таких як акселерометр, геолокація та камера. Виявлено, що правильне тестування цих аспектів є важливим для забезпечення якості мобільних застосунків.

На підставі проведеного дослідження можна зробити висновок, що вибір правильних технологій та інструментів для автоматизованого тестування мобільних застосунків є ключовим фактором у забезпеченні якості та успішної розробки мобільних додатків.

РОЗДІЛ 3 ВИБІР ІНСТРУМЕНТІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ

3.1 Обрання інструментів тестування

Під час обрання інструментів для тестування мобільних застосунків, ключовим завданням є визначення найбільш прийнятних засобів, які відповідають конкретним потребам проекту [21]. У даному випадку, для дослідження обрано три популярні інструменти - Appium, Katalon Studio та Espresso, які представляють різні підходи та можливості в автоматизації тестування мобільних додатків.

Appium, як інструмент кросплатформеного тестування, є відмінним вибором для проектів, які мають мобільні додатки для обох основних платформ - Android та iOS. Його універсальність полягає в тому, що він підтримує різні мови програмування, і це робить його привабливим для розробників із різним технічним стеком [22].

Appium активно оновлюється та має широкий актив спільноти, що гарантує підтримку для нових версій операційних систем та пристроїв. Використання Appium може бути вигідним у випадках, коли команда розробників працює з різними технологіями та мовами програмування.

Katalon Studio привертає увагу своєю простотою використання та готовністю до використання в умовах різного рівня складності проекту. Забезпечуючи графічний інтерфейс для створення тестових сценаріїв, він полегшує процес роботи тестувальників та надає можливість використовувати мову програмування Groovy для розширення функціональності [23].

До того ж, Katalon Studio може інтегруватися з такими інструментами, як JIRA чи Jenkins, що полегшує взаємодію з іншими елементами процесу розробки та тестування [23]. Його можливості безкоштовного використання дозволяють проектам з обмеженим бюджетом використовувати потужність автоматизованого тестування.

Espresso спеціалізується на тестуванні Android-додатків та володіє глибокою інтеграцією з Android Studio. Це робить його відмінним вибором для проектів, орієнтованих на розробку для платформи Android. Espresso акцентується на швидкості виконання тестів та надійності результатів [24].

Його вбудована підтримка для тестування асинхронного коду та UI компонентів робить його потужним інструментом для розробників Android. Однак, важливо враховувати, що Espresso є специфічним інструментом для Android, і його використання може бути обмеженим, якщо проект охоплює кілька платформ.

Обираючи між цими інструментами, дослідження буде орієнтоване на порівняння їхньої продуктивності, можливостей і адаптованості до конкретного випадку використання. Це дозволить визначити, який інструмент найбільш відповідає вимогам конкретного проекту для ефективного тестування мобільного додатку.

3.2 Визначення Android-застосунку для дослідження

Для проведення дослідження технологій автоматизованого тестування мобільних застосунків, необхідно чітко визначити об'єкт дослідження. У даному випадку, вибір падає на мобільний додаток платформи Android.

Обрання Android-застосунку для дослідження має кілька обґрунтованих причин. По-перше, платформа Android є однією з найбільш популярних серед користувачів мобільних пристроїв у світі. Дослідження технологій тестування на цій платформі дозволить отримати практичні результати, які будуть важливими для широкого кола розробників та тестувальників.

По-друге, маючи на увазі розповсюдженість різних типів додатків для Android (нативні, веб-застосунки, гібридні), вибір падає на представника кожного типу. Це дозволить провести більш повне та репрезентативне дослідження, охоплюючи різні сценарії розробки та використання.

Важливим аспектом визначення Android-застосунку є також його функціональність та потенціал для автоматизованого тестування. Обраний додаток повинен включати елементи, які часто зустрічаються в реальних проектах, такі як введення користувача, взаємодія з базою даних, робота з мережею та інші, для максимальної релевантності дослідження.

Визначення Android-застосунку для дослідження є важливим етапом, який враховує потреби реальних проектів та сприяє отриманню значущих результатів для розширення загального розуміння автоматизованого тестування мобільних додатків.

3.2.1 Вибір Android клієнта Instagram як об'єкта дослідження

Після детального перегляду різних мобільних застосунків на платформі Android для автоматизованого тестування, було обрано соціальний застосунок Instagram з ряду обґрунтованих причин.

Популярність та розповсюдженість: Instagram є однією із найпопулярніших соціальних мереж у світі зі значною кількістю активних користувачів. Обрання такого, широко використовуваного, застосунку дозволяє провести дослідження, яке враховує потреби великої аудиторії та показники, що відображають реальні умови використання мобільних додатків [25].

Різноманітність функціоналу: клієнт Instagram представляє собою багатофункціональний застосунок, який включає елементи соціальної мережі, мультимедійного вмісту, взаємодії користувачів та інші функції. Вибір клієнта Instagram для дослідження дозволяє охопити різні аспекти автоматизованого тестування, включаючи роботу зі зображеннями, відео, аудіо, а також різноманітні елементи інтерфейсу [25].

Слідування трендам та інноваціям: клієнт Instagram постійно вдосконалюється та впроваджує нові функції. Вибір такого застосунку для дослідження дозволяє враховувати сучасні тенденції розробки мобільних додатків та випробовувати засоби тестування на новітніх можливостях, які можуть з'явитися у майбутньому [25].

Використання різних технологій: існують клієнти Instagram, які підтримують, як Android, так і iOS платформи, що дає змогу провести порівняльний аналіз ефективності технологій тестування на різних мобільних операційних системах.

3.2.2 Опис функціоналу та особливостей застосунку

Клієнт Instagram визначається своїм основним завданням - спільним обміном візуальним контентом. Користувачі можуть завантажувати фотографії та відео, взаємодіяти зі своїм контентом через "лайки" та коментарі, а також підписуватися на облікові записи інших користувачів.

Історії та живі відео: Введення функції "Історії" дозволяє користувачам додавати тимчасовий контент, який зникає після 24 годин. "Живі Відео" дозволяють транслювати відео у режимі реального часу, взаємодіяти з глядачами через коментарі та висловлювати свої реакції [26].

Функції взаємодії: Функції вподобань, коментарів та тегів сприяють взаємодії користувачів. Використання власних тегів та підписування на акаунти інших користувачів забезпечує більше взаємодії та зростання аудиторії.

Розміщення реклами: Instagram став популярною платформою для рекламодавців. Користувачі можуть створювати рекламні кампанії, розміщувати рекламу в історіях та стрічці, використовувати кнопки "Купити" для продажу товарів та послуг [26].

Використання інтеграцій та розширень: Instagram підтримує інтеграції з іншими соціальними мережами, що дозволяє користувачам легко ділитися своїм контентом на інших платформах. Розширення та фільтри роблять процес редагування та вдосконалення контенту більш творчим [26].

Бізнес-опції та аналітика: Instagram надає бізнес-користувачам можливість створювати бізнес-профіль, використовувати функції "Купити" для продажу товарів, а також має аналітичні інструменти для вимірювання ефективності рекламних кампаній та статистики аудиторії [26].

Стрічка та алгоритмічне сортування: Алгоритмічне сортування стрічки дозволяє показувати користувачам контент, який ймовірно їм сподобається, забезпечуючи персоналізований досвід використання [26].

Множинні платформи та операційні системи: існують клієнти Instagram для Android, так і для iOS, що робить цей застосунок універсальним та доступним на більшості мобільних пристроїв.

3.3 Проєктування тестових сценаріїв

Проєктування тестових сценаріїв - це стратегічний і творчий процес, спрямований на створення детальних та ефективних планів дій для виконання тестування мобільних застосунків. Цей етап визначає ключові кроки та дії, які будуть виконуватися з метою перевірки функціональності, продуктивності та надійності застосунків [27].

Перед початком проєктування тестових сценаріїв, необхідно ретельно аналізувати вимоги до застосунку та визначити стратегічні цілі тестування. Це включає в себе визначення обсягу тестування та вибір головних функціональностей застосунку.

Проєктування тестових сценаріїв передбачає розробку докладних планів дій, видом мого тестування буде користувацьке введення, всі введенні данні будуть валідними.

Тестовий сценарій є основою для подальшої реалізації тестових скриптів у вибраному інструменті автоматизації тестування мобільних застосунків.

3.3.1 Визначення ключових функцій для тестування Android клієнта Instagram

Було обрано три ключові функції для порівняльного дослідження, такі як: Авторизація, створення нової історії та перегляд контенту іншого користу-

вача у його профілі. Кожен з трьох тестів містить перевірку основної функціональності застосунку Instagram, враховуючи ключові аспекти взаємодії та враження від використання [13].

Авторизація у застосунку:

Тестування цієї функціональності включає в себе не лише перевірку правильності введення даних, але і визначення ефективності механізмів відновлення паролю та заходів безпеки для облікового запису. Успішна авторизація є критичною для забезпечення безпеки та конфіденційності даних користувача [27].

Створення нової історії:

Функціонал додавання історій є важливим для платформ, які ставлять акцент на вмісті. Тестування цієї функції включає в себе перевірку коректності додавання текстового та мультимедійного контенту, вибору та додавання тегів до історій. Забезпечення коректної роботи цього функціоналу гарантує, що користувачі можуть легко та ефективно ділитися своїми враженнями та контентом [27].

Перегляд контенту іншого користувача у його профілі:

Взаємодія з контентом інших користувачів визначає соціальний характер багатьох мобільних застосунків. Тестування цієї функції включає перевірку доступу до профілю іншого користувача, відображення та коректність контенту на його сторінці. Забезпечення коректної реалізації цього аспекту дозволяє користувачам насолоджуватися повністю соціальним досвідом використання застосунку [27].

Обрані функціональності стануть основою для розробки детальних тестових сценаріїв, охоплюючи всі можливі аспекти їхньої взаємодії та використання. Цей підхід спрямований на забезпечення високої якості, надійності та задоволення потреб користувачів у використанні мобільного застосунку.

3.3.2 Створення тестових сценаріїв з необхідними перевірками

3.3.2.1 Авторизація

Мета тесту:

Перевірити, чи працює процес авторизації користувача у соціальній мережі Instagram та чи правильно здійснюється вхід в обліковий запис.

Кроки тесту:

1. Запуск застосунку:

Відкрити застосунок Instagram на мобільному пристрої.
Перевірити наявність полів для логіну та паролю на сторінці (Рисунок 1).

2. Введення даних користувача для авторизації:

Ввести коректні дані для авторизації у відповідні поля (Рисунок 1) (існуючий логін або електронна пошта та пароль).

Перевірити екран на відсутність повідомлень про похибку, не коректно введених даних.

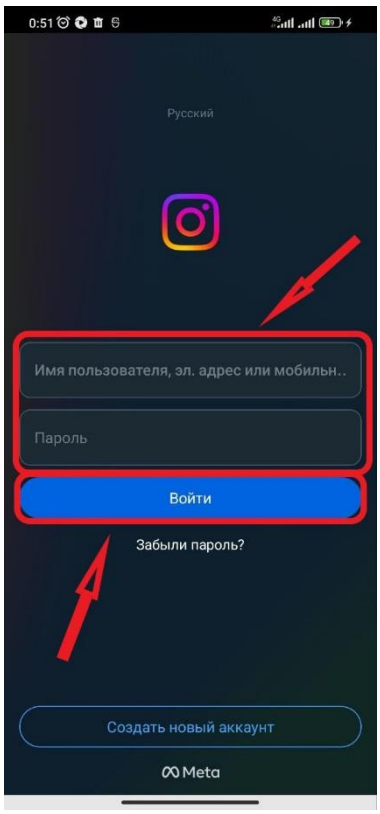

Рисунок 1 – Сторінка авторизації

3. Вхід під своїм обліковим записом:

Виконати клік по кнопці Увійти (Рисунок 1).

Перевірити, чи змінився екран авторизації на екран з можливістю збереження своїх даних для наступних авторизацій (Рисунок 2).

4. Відмова від збереження даних для авторизації:

Виконати клік по кнопці Не зараз (Рисунок 2).

Переконатися, що вхід успішний, перевіривши наявність елементу інтерфейсу Дім (Рисунок 3).

Очікуваний результат:

- Процес авторизації працює швидко та ефективно.
- Користувач отримує доступ до головної сторінки після успішної авторизації.

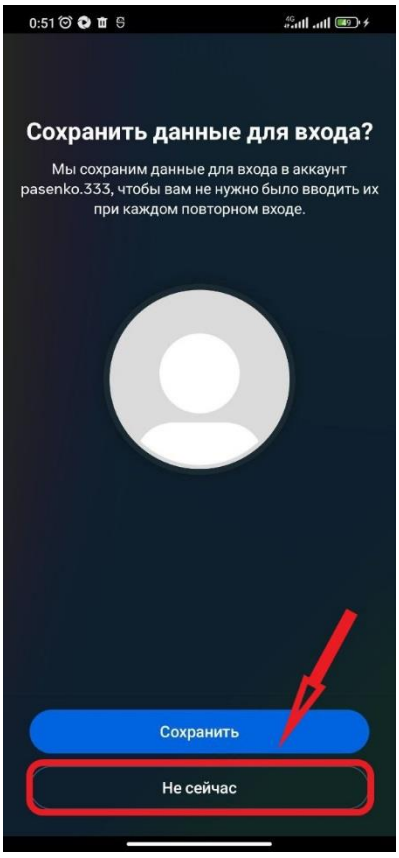

Рисунок 2 – Сторінка збереження даних

Рисунок 3 – Головка сторінка

3.3.2.2 Створення нової історії

Мета тесту:

Перевірити, чи коректно та ефективно працює функція додавання нової історії в застосунку Instagram.

Кроки тесту:

1. Запуск застосунку:

Відкрити застосунок Instagram на мобільному пристрої, якщо в застосунку нема авторизованого користувача, то потрібно авторизуватись під акаунтом тестового призначення.

Переконатися, що застосунок відкритий та користувач авторизований, перевіривши наявність елементу інтерфейсу Дім (Рисунок 3).

2. Перехід до розділу історії:

Виконати свайп зліва на право, натискаємо на ліву частину екрану та проводимо пальцем по екрану до правої його частини, приблизно по центру екрана.

Переконатись, що перехід до сторінки створення історії пройшов успішно, перевіривши наявність елементу інтерфейсу Повідомлення (Рисунок 4).

3. Дозвіл до фото та відео контенту мобільного пристрою:

Виконати клік по кнопці Тільки зараз (Рисунок 4).

Переконатись, що було дозволено доступ до фото та відео контенту, перевіривши наявність елементу інтерфейсу Мікрофон (Рисунок 5).

4. Дозвіл до мікрофону мобільного пристрою:

Виконати клік по кнопці Тільки зараз (Рисунок 5).

Переконатись, що було дозволено доступ до мікрофону, перевіривши наявність елементу інтерфейсу Камера (Рисунок 6).

5. Дозвіл до фото та мультимедію мобільного пристрою:

Виконати клік по кнопці Дозволити (Рисунок 6).

Переконатись, що було дозволено доступ до фото та мультимедії, перевіривши наявність елементу інтерфейсу Камера.

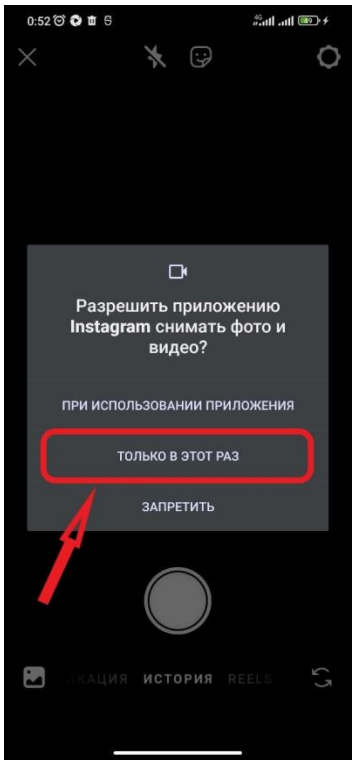

Рисунок 4 – Повідомлення про надання доступу до фото та відео контенту

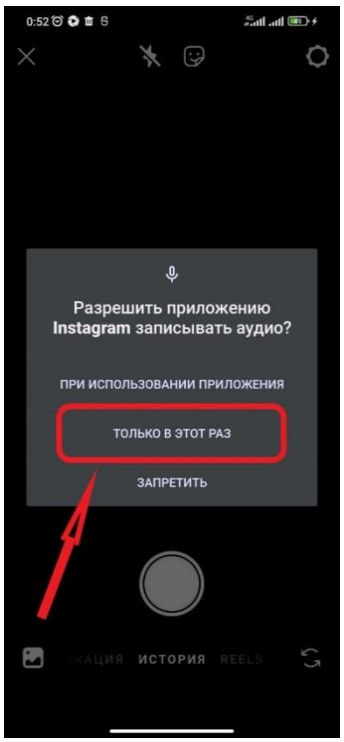

Рисунок 5 – Повідомлення про надання доступу до мікрофону

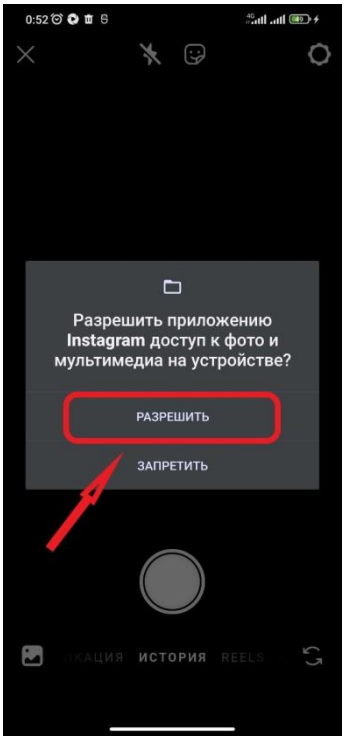

Рисунок 6 – Повідомлення про надання доступу до фото та мультимедії

6. Відкриття галереї:

   Виконати клік по кнопці Галерея (Рисунок 7).

Переконатись, що галерея відкрита, перевіривши наявність елементу інтерфейсу Галерея (Рисунок 8).

7. Вибір фотографії:

Виконати клік по елементу Фото[1] (Рисунок 8).

Переконатись, що відкрилась сторінка редагування фотографії, перевіривши наявність відповідної фотографії на екрані.

8. Вибір аудиторії:

Виконати клік по кнопці Близькі друзі (Рисунок 9).

Переконатись, що вибір аудиторії пройшов успішно, перевіривши наявність зеленої рамки навколо елементу інтерфейсу Близькі друзі.

9. Опублікування історії:

Виконати клік по кнопці Опублікувати.

Переконатись, що на екрані відображається головна сторінка та наша історія з’явилась у стрічці новин, перевіривши наявність зеленої круглої рамки навколо елементу інтерфейсу Моя історія (Рисунок 10).

Рисунок 7 – Сторінка створення нової історії

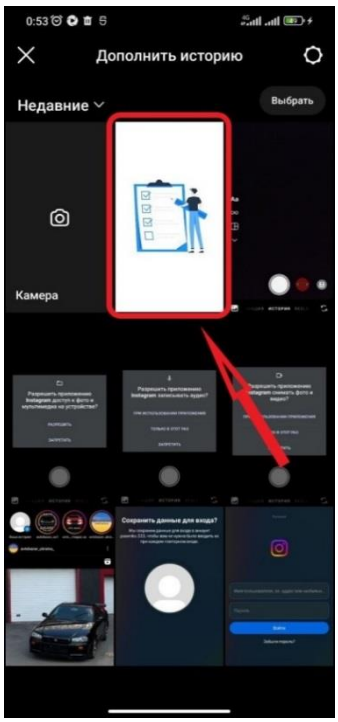

Рисунок 8 – Сторінка галереї

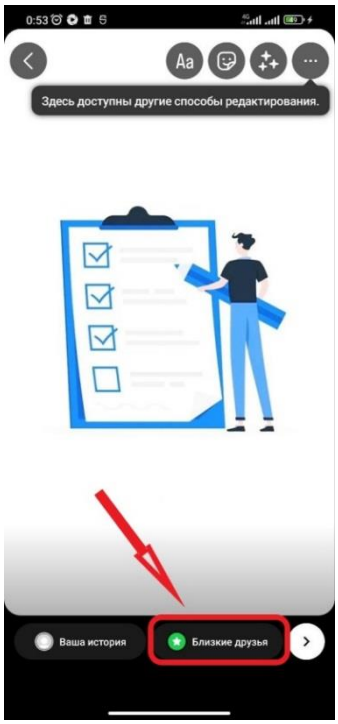

Рисунок 9 – Сторінка редагування історії

10.Перевірка опублікування історії:

Виконати клік по кнопці Моя історія (Рисунок 10).

Переконатись, що опублікування нової історії пройшло успішно, перевіривши наявність відповідної фотографії на екрані (Рисунок 11).

Очікуваний результат:

- Процес додавання історії працює швидко та ефективно.
- Додана історія відображається на основній сторінці у стрічці новин.

Рисунок 10 – Нова історія

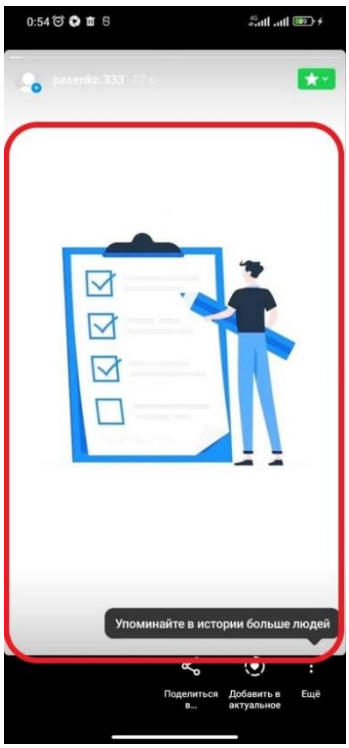

Рисунок 11 – Перегляд опублікованої історії

3.3.2.3 Перегляд контенту іншого користувача у його профілі

Мета тесту:

Перевірити, чи коректно та ефективно працює функція пошуку користувача, перегляду профілю та його контенту в застосунку Instagram.

Кроки тесту:

1. Запуск застосунку:

Відкрити застосунок Instagram на мобільному пристрої, якщо в застосунку нема авторизованого користувача, то потрібно авторизуватись під акаунтом тестового призначення.

Переконатись, що застосунок відкритий та користувач авторизований, перевіривши наявність елементу інтерфейсу Дім (Рисунок 3).

2. Перехід на сторінку пошуку користувачів:

Виконати клік по кнопці Пошук (Рисунок 12).

Переконатись, що сторінка пошуку відкрита успішно, перевіривши наявність елементу інтерфейсу Поле пошуку (Рисунок 13).

Рисунок 12 – Кнопка сторінки пошуку

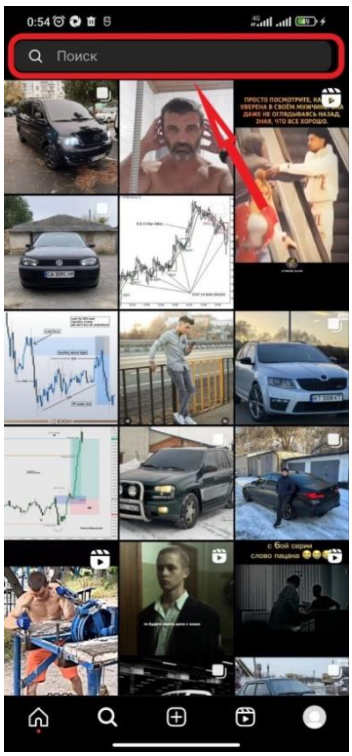

Рисунок 13 – Сторінка пошуку

3. Введення імені користувача

Виконати введення у поле пошуку ім’я користувача (Рисунок 14).

Переконатись, що в полі пошуку відображається введене нами ім’я користувача, перевіривши наявність шуканого користувача у списку результатів пошуку (Рисунок 14).

4. Перехід до профілю користувача:

Виконати клік по іконці шуканого користувача (Рисунок 14).

Переконатись, що сторінка профілю користувача відкрита успішно, перевіривши наявність елементу з іменем шуканого користувача (Рисунок 15).

5. Перегляд контенту користувача:

Виконати клік по першій фотографії зі списку контенту користувача Фото[1] (Рисунок 15).

Переконатись, що відкрилась стрічка перегляду контенту користувача, перевіривши наявність елементу інтерфейсу Лайк (Рисунок 16).

6. Поставити лайк:

Виконати клік по кнопці Лайк (Рисунок 16).

Переконатись, що було зафіксована взаємодія з кнопкою Лайк, перевіривши наявність елементу інтерфейсу Лайк1.

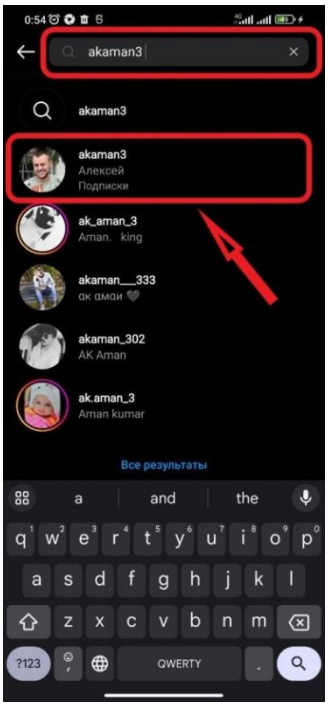

Рисунок 14 – Пошук та результати пошуку користувача

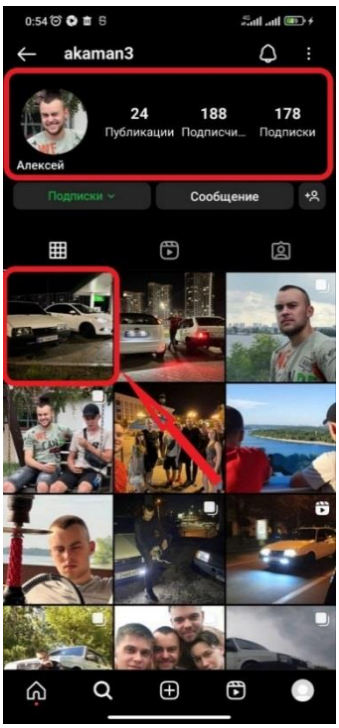

Рисунок 15 – Профіль шуканого користувача

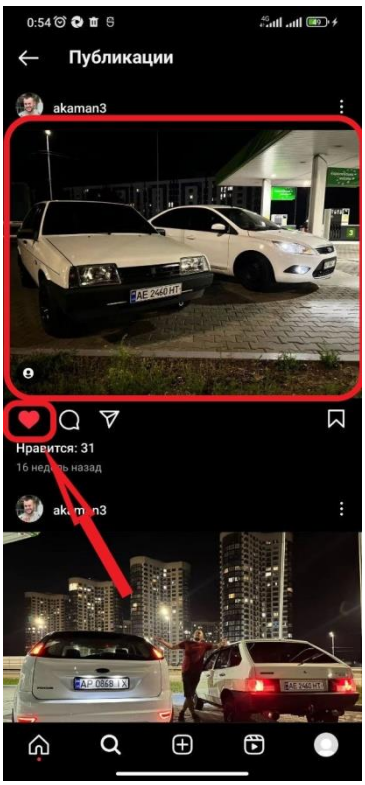

Рисунок 16 – Сторінка контенту користувача

7. Вихід із профілю:

Виконуємо клік по кнопці Головна сторінка.

Переконатись, що профіль користувача закритий, перевіривши наявність елементу інтерфейсу Дім (Рисунок 3).

Очікуваний результат:

- Профіль користувача відображається коректно та інформативно.
- Після пошуку за іменем користувача, він з’явився у списку результатів.
- Фотографії та відео завантажуються.
- Взаємодія з кнопкою «Лайк» відображається правильно.

3.4 Розгортання засобів тестування

В процесі автоматизованого тестування мобільних застосунків, ефективне розгортання засобів тестування визначається не лише вибором відповід-

них інструментів, але й правильною конфігурацією та налагодженням у відповідності до специфікацій та вимог проекту. У даному розділі розглядається етап дослідження - розгортання обраних засобів тестування на платформі Android.

Визначено три платформи для тестування - Appium, Katalon Studio та Espresso. Ці інструменти забезпечують надійне та ефективне тестування мобільних додатків, і тепер настав час їхнього розгортання для обраного Android-застосунку. Але, оскільки доступ до вихідного коду застосунку, для якого виконувалося тестування, є недоступним, а це є необхідним для тестування з використанням Espresso, то практичне порівняння обмежилося порівнянням Appium з Katalon Studio.

Перед налаштуванням робочого середовища потрібно оголосити назву пристроїв та їх характеристики. Налаштування та написання коду, в тому числі й підключення Android пристрою, було проведено на ноутбуці Lenovo 320-15IKB (Type 81BG, 81BT) Laptop (ideapad) - Type 81BG з наступними характеристиками:

1. Чотирьохядерний процесор Intel Core i5-8250U CPU з тактовою частотою 1.6 ГГц.
2. Графічний процесор NVIDIA GeForce MX 150.
3. 12 ГБ оперативної пам’яті.
4. Операційна система Windows 10.

Android – пристроєм, на якому знаходиться застосунок Instagram та відбувались всі тестування, був телефон Xiaomi Redmi 9T Global Version. Він має наступні характеристики:

1. Операційна система Android, версії 12 SKQ1.211202.001.
2. Процесор Qualcomm Snapdragon 662 з типом ядра Kryo 260, кількістю ядер 4+4 та з тактовою частотою 2.0 ГГц + 1.8 ГГц.
3. 4 ГБ оперативної пам’яті.
4. Діагональ екрана 6.53 та з роздільною здатністю дисплея 2340 x 1080.
5. Роз’їм USB Type-C.

Також для підключення телефону до ноутбука використовувався звичайний кабель USB Type-C, що йшов в комплекті до телефону.

Мета даного розділу - надати детальний інсайт у процес розгортання засобів тестування та визначити, як обрані інструменти можуть бути оптимально інтегровані в розробку Android-застосунку. Через вивчення цього етапу дослідження ми отримаємо розуміння практичного використання обраних засобів тестування у реальних умовах та їхнього впливу на якість та продуктивність мобільних додатків.

3.4.1 Налаштування середовища розробки для фреймворку Appium

Для ефективного проведення тестування мобільних застосунків за допомогою фреймворку Appium, важливо правильно підготувати робоче середовище на комп'ютері. Перед цим необхідно впевнитися, що на комп'ютері вже встановлено JDK і налаштовано змінні оточення JAVA_HOME та PATH.

Додатково, для тестування Android-пристроїв обов'язково встановлення Android SDK. Це можна легко зробити шляхом встановлення Android Studio, що включає в себе не лише Android SDK, але й необхідні інструменти, такі як Android Debug Bridge та Virtual Device Manager. Встановлення Android Studio автоматично забезпечить наявність необхідних компонентів та дозволить налаштувати Android Emulator для запуску емальованих Android-пристроїв [28].

Оскільки Appium Server розроблений для використання на платформі Node.js, першим крок перед початком використання - це встановлення цього фреймворку. Процедура інсталяції здійснюється стандартним чином, і якщо ви користуєтеся операційною системою Windows, програмне забезпечення автоматично розміщується у каталозі C:\Program Files\nodejs. Для перевірки успішної інсталяції, потрібно у cmd ввести наступні команди:

Встановлення Node.js: > node –v

Перевірка успішного встановлення: > npm -v

Далі, використовуючи менеджер пакетів Node.js, виконайте процес встановлення сервера Appium за допомогою команди [28]:

> npm i - -location=global appium

У випадку Windows, Appium розміщується у каталозі C:\Users\ім’я_користувача\AppData\Roaming\npm\node_modules\appium.

Необхідно також перевірити правильність та версію встановленого сервера Appium, щоб забезпечити його належне функціонування та готовність до подальшого використання, використовуючи команду:

> appium -v

Після успішної установки сервера, наступним етапом є встановлення відповідного драйвера для конкретної платформи. Використайте команду для виведення списку доступних драйверів та подальшу команду для встановлення драйвера UiAutomator2 [28].

Виведення списку: > appium driver list

Команда для встановлення: > appium driver install uiautomator2

Тепер, щоб запустити сервер Appium, достатньо використовувати командний рядок і викликати команду > appium, передаючи необхідні значення для бажаних можливостей як опції. Завершити роботу сервера Appium можна, просто натиснувши Ctrl + C у вікні командного рядка, в якому сервер був запущений.

При звичайній установці Appium рекомендується також встановити утиліту, що використовується під час розробки тестів за допомогою Appium, — це Appium Inspector. Appium Inspector фактично є клієнтом сервера Appium, аналогічним клієнтській бібліотеці, такій як Java або Python. Однак його особливість полягає в тому, що він надає графічний інтерфейс для введення опцій, пошуку та взаємодії з елементами інтерфейсу мобільного застосунку [28]. Appium Inspector також дозволяє записувати користувацькі дії з застосунком у скрипті команд Appium.

Щоб завантажити Appium Inspector, слід перейти до репозиторію за посиланням https://github.com/appium/appium-inspector/releases та обрати відпові-

дну версію для вашої операційної системи. Інсталяція здійснюється стандартним чином, наприклад, у Windows в каталозі C:\Program Files\Appium Inspector.

Зазначу, що налаштування робочого середовища для розробки тестів мобільних застосунків на платформі Android за допомогою фреймворку Appium у операційній системі Windows, тепер вважається завершеним.

3.4.2 Налаштування середовища розробки для фреймворку Katalon Studio

Кроки налаштування середовища розробки для фреймворку Katalon будуть іноді такими же як і для фреймворку Appium, бо що Appium, що Katalon використовують однакові допоміжні інструменти для своєї роботи.

Для налаштування середовища розробки для фреймворку Katalon, з початку переконатись, що на комп'ютері вже встановлено JDK, а змінні оточення JAVA_HOME та PATH налаштовані правильно [29].

Для тестування на Android-пристроях потрібно встановити Android SDK. Рекомендується встановити Android Studio, що включає в себе Android SDK, Android Debug Bridge та Virtual Device Manager. Інсталяція Android Studio автоматично налаштує необхідні компоненти та дозволить налаштувати Android Emulator для емуляції Android-пристроїв.

Середовищем розробки для Katalon є Katalon Studio, саме у Katalon Studio буде написаний код автоматизованих тестів для мобільного застосунку, на мові програмування Java. Саме тому встановлюємо це середовище розробки, встановити його можна з офіціального сайту. У випадку Windows, Katalon Studio розміщується у каталозі C:\Users\ ім’я_користувача \Katalon Studio [29].

Katalon Studio включає в себе Mobile Recorder, дозволяє записувати дії під час взаємодії з мобільним додатком, а потім генерує відповідний код для автоматизованого тестування. Це полегшує створення тестових сценаріїв для мобільних додатків без необхідності написання коду вручну.

Оскільки Katalon Studio використовує Appium Server для зв’язку з мобільним додатком, інсталяція Node.js є обов'язковою. Процедура інсталяції здійснюється стандартним чином, і якщо ви користуєтеся операційною системою Windows, програмне забезпечення автоматично розміщується у каталозі C:\Program Files\nodejs [29]. Для перевірки успішної інсталяції, потрібно у cmd ввести наступні команди:

Встановлення Node.js: > node –v

Перевірка успішного встановлення: > npm -v

Далі, використовуючи менеджер пакетів Node.js, виконайте процес встановлення сервера Appium за допомогою команди:

> npm i - -location=global appium

У випадку Windows, Appium розміщується у каталозі C:\Users\ім’я_користувача\AppData\Roaming\npm\node_modules\appium.

Необхідно також перевірити правильність та версію встановленого сервера Appium, щоб забезпечити його належне функціонування та готовність до подальшого використання, використовуючи команду:

> appium -v

Після успішної установки сервера, наступним етапом є встановлення відповідного драйвера для конкретної платформи. Використайте команду для виведення списку доступних драйверів та подальшу команду для встановлення драйвера UiAutomator2 [29].

Виведення списку: > appium driver list

Команда для встановлення: > appium driver install uiautomator2

Налаштування робочого середовища для розробки тестів мобільних застосунків на платформі Android за допомогою фреймворку Katalon у операційній системі Windows тепер вважається завершеним.

3.4.3 Налаштування мобільного пристрою для автоматизованого тестування

Розблокування режиму розробника: Розблокування режиму розробника є необхідним для отримання доступу до розширених налаштувань на мобільному пристрої. Щоб це зробити, зазвичай потрібно кілька разів клацнути на номері збудженого пристрою або версії ПЗ у розділі "Про телефон" у налаштуваннях. Після цього з'явиться повідомлення про успішне ввімкнення режиму розробника.

Увімкнення режиму USB-налагодження: Режим USB-налагодження дозволяє здійснювати зв'язок між мобільним пристроєм і комп'ютером для виконання тестів. Після активації режиму розробника, цю опцію можна знайти у налаштуваннях у розділі "Система" або "Про телефон". Варто включити цей режим, якщо він ще не ввімкнений [30].

Дозвіл на інсталяцію додатків з невідомих джерел: Щоб встановлювати додатки, які не походять із звичайних магазинів (наприклад, Google Play), слід дозволити інсталяцію з невідомих джерел. Ця опція зазвичай знаходиться в налаштуваннях безпеки під назвою "Дозволити інсталяцію додатків з невідомих джерел". Після активації цієї опції можна встановлювати додатки, завантажені не з офіційних магазинів.

Вимкнення анімацій: Важливо відключити анімації на мобільному пристрої, оскільки вони можуть впливати на швидкість виконання тестів. Цю опцію можна знайти у розділі "Про телефон" або "Загальні налаштування", залежно від версії ОС. Зазвичай, це підпункт з назвою "Анімація вікна" або "Анімація переходів", і його можна вимкнути для поліпшення продуктивності під час тестування [30].

Ці кроки дозволяють оптимізувати мобільний пристрій для виконання автоматизованих тестів та гарантують стабільну роботу під час тестування мобільних застосунків.

Також не забуваємо встановити сам застосунок на мобільний пристрій який будемо тестувати. Потрібно відшукати в інтернеті файл застосунку

Instagram в форматі .apk. Для проведення мого порівняльного дослідження було вибрано версію застосунку: 288.1.0.22.66. А вибраним мною .apk файл був під назвою: instagram-288-1-0-22-66.apk.

3.5 Створення проектів

При виборі інструментів для автоматизованого тестування, треба звертати увагу, чи обмеження доступу до вихідного коду застосунку. У випадках, коли вихідний код недоступний, інструменти, які базуються на зовнішній взаємодії зі сторонніми додатками, можуть стати вирішенням. У даному випадку, оскільки Espresso вимагає доступу до вихідного коду для тестування, фокус порівняльного дослідження буде зосереджений на Appium та Katalon.

3.5.1 Створення проекту для Appium

Відкриваємо, завчасно встановлене, середовище розробки Android Studio.

Створюємо новий проект під назвою InstagramTestAppium.

У папці src створюємо папку під назвою AndroidTest. Шлях до папки :\\InstagramTestAppium\app\src.

Створення файлу InstagramTest.java у папці src. Шлях до файлу :\\InstagramTestAppium\app\src\AndroidTest\InstagramTest.java.

Наступний крок це початок розробки самого автотесту, автотест буде написаний за тестовим сценарієм, що був створений у попередніх підрозділах. Створювати будемо через інструмент Appium Inspector, що ми встановлювали завчасно. Особливість даного інструменту є можливість відтворення тестового сценарію завдяки запису кроків під час ручного тестування, система тим часом прописує увесь основний код тестів, який ми потім будемо експортувати до нашого середовища розробки Android Studio. Нам залишається тільки відредагувати код, якщо це потрібно, додати перевірки та залежності.

Для початку запустимо Appium Server (командою appium у вікні командного рядка). Після входу до інструменту Appium Inspector, потрібно налаштувати сесію з сервером, вказавши данні пристрою: назва платформи, її версія, драйвер, ім’я пристрою (Рисунок 17). Після цього можна запускати сесію Appium Inspector кліком по кнопці Start Session. Після запуску сесії з’явиться вікно Appium Inspector розділене на три частини (Рисунок 18): ліва частина відображає поточний інтерфейс мобільного додатка, середня частина містить код XML-розмітки інтерфейсу додатка, права частина містить верхню та нижню частину, у верхній розташовані кнопки для взаємодії з елементами поточного інтерфейсу, а у нижній частині відображається значення локаторів та атрибутів виділеного елемента інтерфейсу. Елемент може бути виділений як у дереві тегів, так і на самому інтерфейсі. Для початку запису дій користувача в Appium Inspector, потрібно виконати клік по кнопці Start Recording на Панелі інструментів.

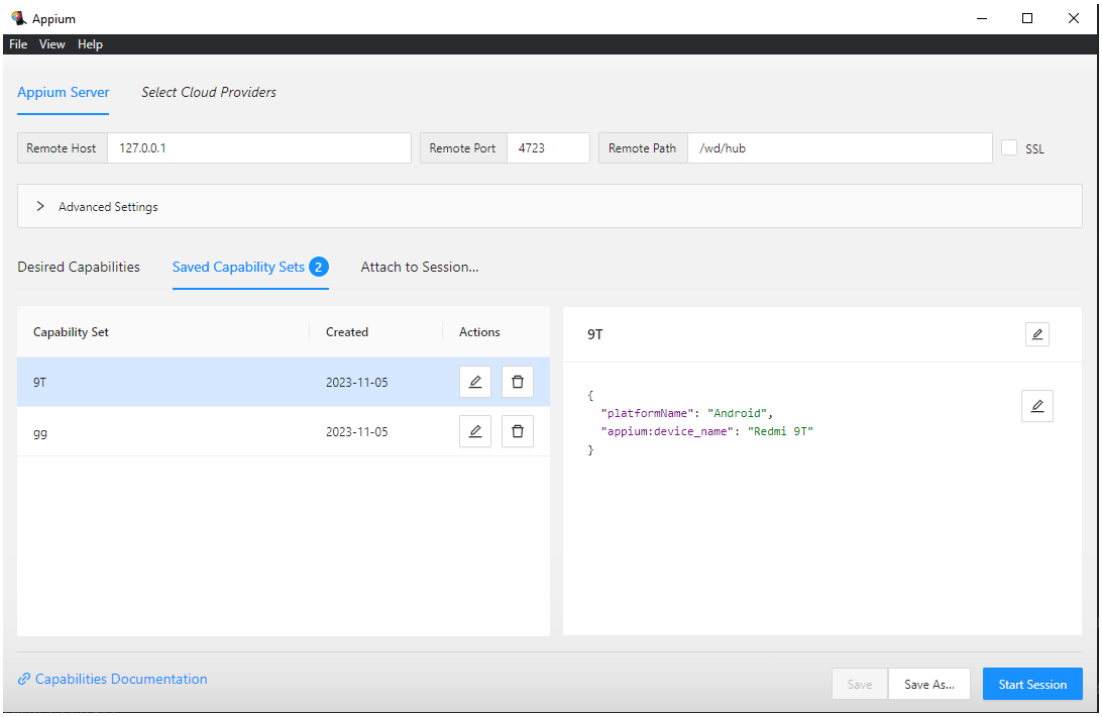

Рисунок 17 – Налаштування сесії з Appium Server

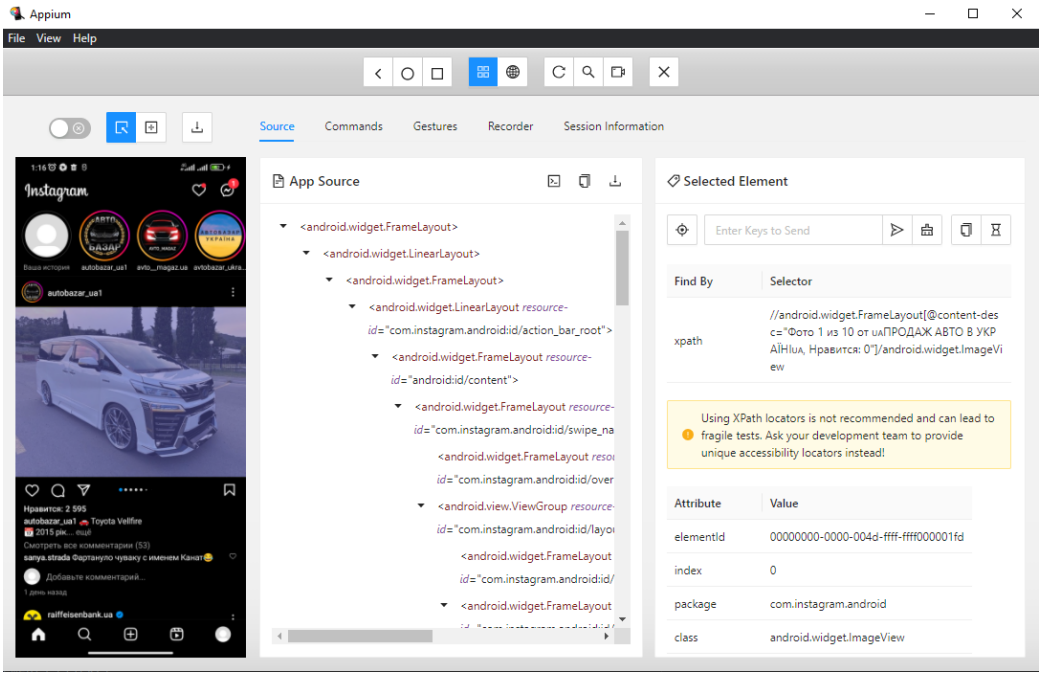

Рисунок 18 – Вікно для взаємодії з елементами в Appium Inspector

У ручному режимі проводимо тестування за сценарієм Android застосунку Instagram в інструменті Appium Inspector. Коли завершено проходження тестового сценарію, виконуємо зупинку запису тестового скрипту, виконаємо клік по кнопці Show та копіюємо наведений код, що там розташований до файлу InstagramTest.java в папці AndroidTest.

Після експорту коду до створеного нами файлу InstagramTest.java, потрібно зайти до файлу build.gradle, що знаходиться за шляхом:\\ :\\InstagramTestAppium\app\src\ build.gradle, та ввести до нього всі ті залежності які потребує згенерований код Appium Inspector. Такі як: java-client, junit-bom, junit-jupiter, selenium, slf4j (Рисунок 19). Демонстрація коду залежностей:

```
plugins {
    id 'java'
}
group = 'org.example'
version = '1.0-SNAPSHOT'
repositories {
    mavenCentral()
```

```
}
dependencies {
    testImplementation platform('org.junit:junit-bom:5.9.1')
    testImplementation 'org.junit.jupiter:junit-jupiter'
    testImplementation 'io.appium:java-client:9.0.0'
    testImplementation    'org.seleniumhq.selenium:selenium-
java:4.15.0'
    testImplementation      'org.apache.logging.log4j:log4j-
api:3.0.0-alpha1'
    testImplementation      'org.apache.logging.log4j:log4j-
slf4j-impl:3.0.0-alpha1'
}
test {
    useJUnitPlatform()
}
```

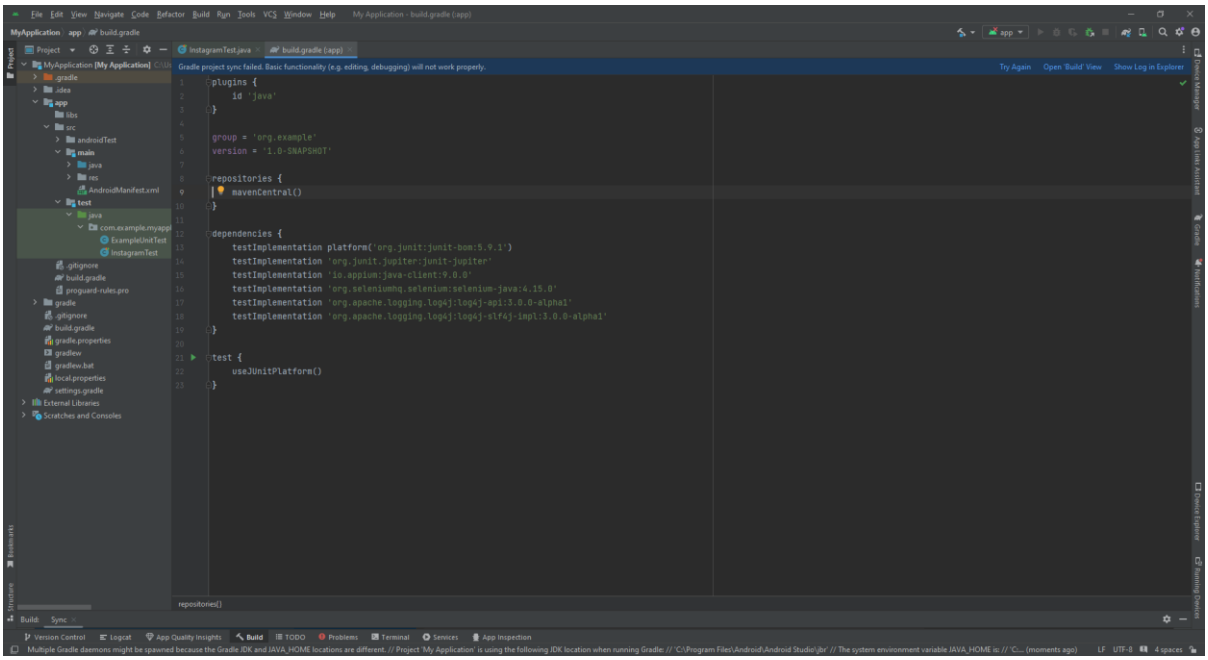

Рисунок 19 – Середовище розробки Android Studio

Наступним кроком є рефакторинг отриманого коду, що згенерував Appium Inspector під час запису мого ручного тестування.

3.5.2 Створення проекту для Katalon

Відкриваємо, завчасно встановлене, середовище розробки Katalon Studio.

Створення нового проекту під назвою InstagramTest Katalon.

У папці Test Cases створюємо папку під назвою AndroidTest. Шлях до папки :\\InstagramTestAppium\Profiles\ Test Cases.

Наступний крок - це початок розробки самого автотесту. Автотест буде написаний за тестовим сценарієм, що був створений у попередніх підрозділах. Створювати будемо через інструмент Mobile Recorder, що знаходиться у самому середовище розробки Katalon Studio. Особливістю даного інструменту є можливість відтворення тестового сценарію, завдяки запису кроків під час ручного тестування, система, тим часом, прописує увесь код та підключає бібліотеки автоматично, що нам залишається, тільки відредагувати код, якщо це потрібно, та додати перевірки.

Після входу до інструменту Mobile Recorder, обираємо телефон до якого хочемо під'єднатись, в нашому випадку це Xiaomi Redmi 9T. Робимо вибір з чого повинен відбуватись запуск, вибираємо з файлу. Та нижче вносимо Application ID застосунок, це потрібно для запуску (Рисунок 20). Тиснемо кнопку старт, Katalon запускає Appium Server для підключення та роботи фреймворку. Після запуску сесії з'явиться вікно Mobile Recorder розділене на три частини (Рисунок 20): права частина містить верхній та нижній розділ, у верхньому відображається поточний інтерфейс мобільного додатка, а у нижній частині відображається значення локаторів та атрибутів виділеного елемента інтерфейсу; середня частина також містить верхню та нижню частину, у верхній розташовані кнопки для взаємодії з елементами поточного інтерфейсу, а у нижній частині розташований код XML-розмітки інтерфейсу додатка; ліва частина відображає кроки взаємодії з елементами, що були записані під час даної сесії. Елемент може бути виділений як у дереві тегів, так і на самому інтерфейсі. Для початку запису дій користувача у Mobile Recorder, потрібно виконати клік по кнопці Run на Панелі інструментів.

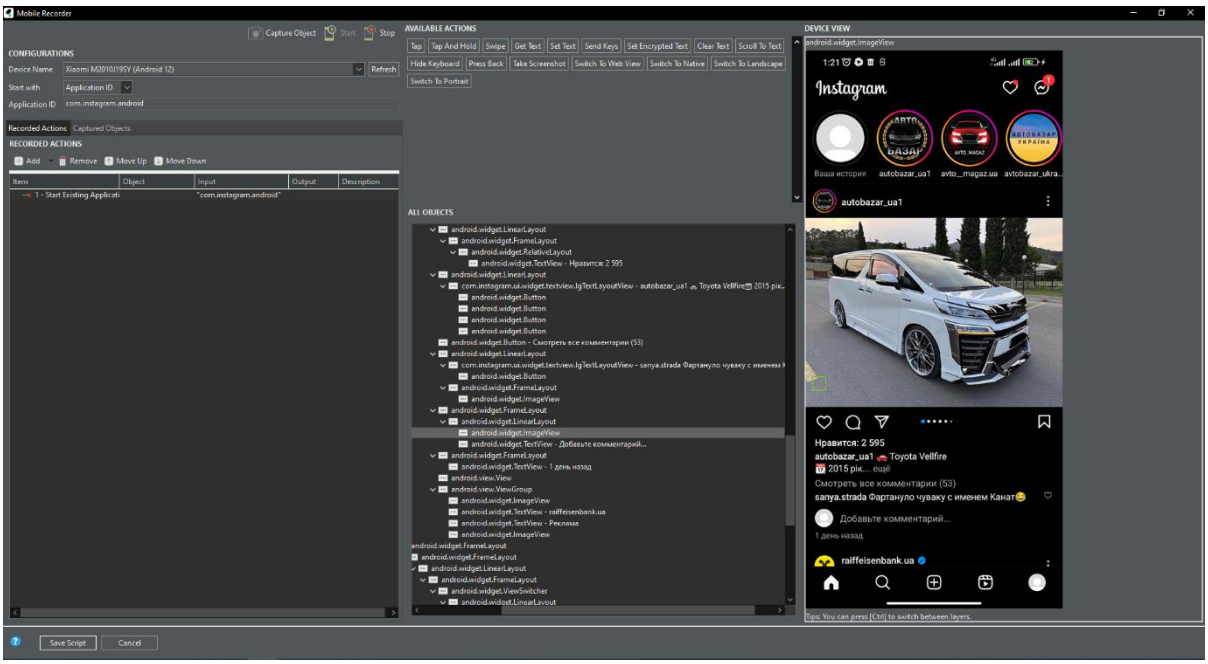

Рисунок 20 – Вікно для взаємодії з елементами в Mobile Recorder

У ручному режимі проводимо тестування за сценарієм Android застосунку Instagram в інструменті Mobile Recorder. Коли завершено проходження тестового сценарію, виконуємо зупинку запису тестового скрипту, виконаємо клік по кнопці Stop, потім зберігаємо до папки AndroidTest та даємо назву збереженому файлу InstagramTest.

Mobile Recorder і Appium Inspector різняться своїм розташуванням. Головна відмінність полягає в тому, що Appium Inspector є самостійним інструментом, незалежним від Android Studio. З іншого боку, Mobile Recorder є вбудованим засобом у середовище розробки Katalon Studio. Це означає, що під час створення проекту для фреймворка Katalon немає необхідності експортувати код з Mobile Recorder. Після збереження Mobile Recorder автоматично додає необхідні залежності, імпортує бібліотеки та створює всі необхідні тести (Рисунок 21). Саме томі в нас відсутній процес експортування, при створенні проекту для Katalon.

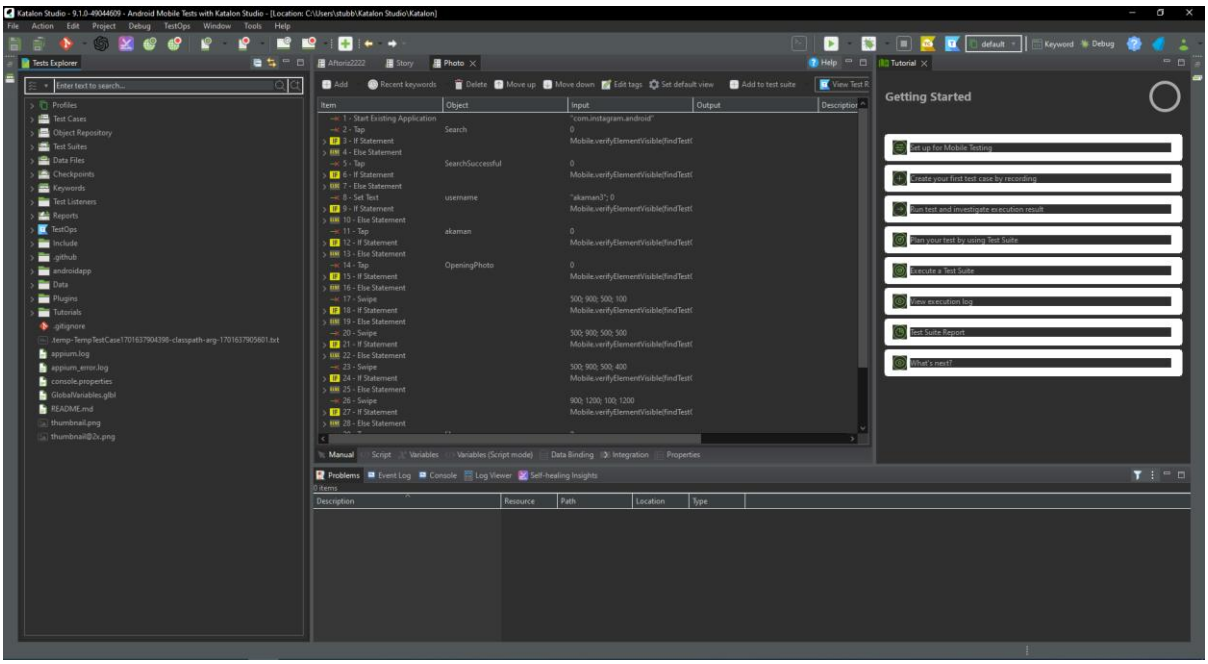

Рисунок 21 – Середовище розробки Katalon Studio

Наступним кроком є рефакторинг отриманого коду, що згенерував Mobile Recorder після ручного тестування за тестовим сценарієм.

3.6 Рефакторинг отриманого коду

Рефакторинг коду - це процес покращення коду, коли його змінюють так, щоб він був більш зрозумілим, ефективним і легше підтримувався. В основному це не вносить нового функціоналу, а лише поліпшує якість і структуру коду. Рефакторинг може включати в себе перейменування змінних, вилучення дублюючого коду, розділення складних функцій на менші, тощо. Це допомагає зробити код більш зрозумілим для розробників, покращує його підтримку і розширення, а також може зменшити ймовірність помилок.

3.6.1 Рефакторинг отриманого коду для фреймворка Appium

Після успішного експорту коду з інструменту Appium Inspector, наступним важливим етапом в процесі автоматизації тестування є рефакторинг отриманого коду. Рефакторинг - це процес удосконалення структури та якості коду

без зміни його функціональності. В контексті Appium, це може значно поліпшити читабельність, підтримку і ефективність вашого тестового скрипту.

Крок 1: Перейменування елементів:

Перейменування елементів у коді допомагає зробити його більш зрозумілим та покращує читабельність. Замініть автоматично згенеровані чи непрозорі імена на більш інформативні.

Приклад:

Було: `WebElement e1 = driver.findElement(AppiumBy .id("com.google.android.gms:id/cancel"));`

Стало: `WebElement LoginButton = driver.findElement (AppiumBy.id("com.google.android.gms:id/cancel"));`

Групування дій:

Організуйте послідовні кроки тестового сценарію в логічні групи для полегшення читання та розуміння.

Використання Page Object Pattern:

Видалення непотрібних або дубльованих дій:

Перегляньте код та вилучите непотрібні або дубльовані кроки, що покращить його ефективність.

Додавання коментарів:

Додайте пояснення та коментарі до коду, щоб інші розробники або тестувальники могли легше розібратися в логіці тестів.

Рефакторинг коду дозволяє покращити його структуру, зробити його більш модульним та зручним для обслуговування. Добре розроблений код сприяє підтримці та розширенню автоматизованих тестів в майбутньому.

Останнім кроком для завершення написання тесту є додання перевірок до тестів. На кожному з кроків виконання тесту буде проводитись перевірка, одна з наступних: перевірка знаходження на конкретному екрані, перевірка наявності елементу на екрані, перевірка відсутності елементу на екрані, тощо.

Переглянути код, який пройшов рефакторинг та містить усі перевірки, що був написаний для фреймворку Appium, можна у розділі 4.2.

3.6.2 Рефакторинг отриманого коду для фреймворка Katalon

Крок 1: Перейменування елементів:

Перейменування елементів у коді допомагає зробити його більш зрозумілим та покращує читабельність. Замініть автоматично згенеровані чи непрозорі імена на більш інформативні.

Приклад

Було: `Mobile.tap(findTestObject('Object Repository /Aftoriz/android.view.ViewGroup'), 0)`

Стало: `Mobile.tap(findTestObject('Object Repository /Aftoriz/LoginButton'), 0)`

Видалення непотрібних або дубльованих дій:

Перегляньте код та вилучите непотрібні або дубльовані кроки, що покращить його ефективність.

Додавання коментарів:

Додайте пояснення та коментарі до коду, щоб інші розробники або тестувальники могли легше розібратися в логіці тестів.

Рефакторинг коду дозволяє покращити його структуру та робить більш зручним для обслуговування. Правильно розроблений код сприяє підтримці та розширенню автоматизованих тестів в майбутньому.

Останнім кроком для завершення написання тесту є додання перевірок до тестів. На кожному з кроків виконання тесту буде проводитись перевірка, одна з наступних: перевірка знаходження на конкретному екрані, перевірка наявності елементу на екрані, перевірка відсутності елементу на екрані, тощо.

Переглянути код, який пройшов рефакторинг та містить усі перевірки, що був написаний для фреймворку Katalon, можна у розділі 4.2.

3.7 Висновок до розділу 3

У цьому розділі ми ретельно розглянули процес обрання та налаштування інструментів тестування для дослідження мобільного застосунку. Починаючи з вибору платформи і об'єкта дослідження (Instagram), ми детально описали функціонал та особливості застосунку.

Далі, ми визначили ключові функції для тестування та розробили детальні тестові сценарії, які охоплюють основні взаємодії користувача з застосунком, такі як авторизація, створення нової історії та перегляд контенту інших користувачів.

У цьому розділі ми вивчили процес розгортання засобів тестування та налаштування мобільного пристрою для автоматизованого тестування. Також, ми розглянули створення проектів для фреймворків Appium та Katalon.

Одним із ключових етапів в цьому процесі був рефакторинг отриманого коду для підвищення його читабельності, стійкості та ефективності в майбутньому використанні.

Усі ці етапи виявилися важливими для побудови ефективного тестового середовища, яке дозволить проводити надійне та комплексне тестування мобільного застосунку Instagram. За основу було взято високопродуктивні інструменти, що дозволяють автоматизувати тестові сценарії та полегшують процес тестування мобільних додатків на різних етапах їх розробки.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ ІНСТРУМЕНТІВ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ

4.1 Виконання тестових сценаріїв

Ось настав той час, коли, після написання тестового сценарію, налаштовано середовище розробки та використано інструменти Appium Inspector та Mobile Recorder для запису тестів. Було створено нові проекти, проведено рефакторинг коду у середовищах розробки Android Studio та Katalon Studio. Тепер, з усім готовим, ми готові запустити наші тести та здійснити функціональні перевірки мобільного застосунку Instagram на платформі Android.

У даному розділі розглянемо процес виконання тестів, використовуючи фреймворки Appium та Katalon. Ми розглянемо, як використовувати підготовлені тестові сценарії та інструменти, для автоматизації та перевірки функціональності вашого мобільного додатку. Детально розглянемо кроки запуску тестів, важливими аспектами є спостереження за виконанням, та аналіз результатів для забезпечення якості та стабільності продукту.

Зазначена діяльність виконується в контексті фреймворків Appium та Katalon, що дозволяє нам максимально ефективно використовувати можливості автоматизації тестування та отримувати надійні результати перевірок мобільного застосунку. Далі розділ буде насичений конкретними прикладами, які допоможуть вам легко розпочати та успішно виконати ваші тести для забезпечення якості та безпеки мобільного додатку Instagram.

4.1.1 Виконання тестових сценаріїв на фреймворку Appium в середовищі розробки Android Studio

Крок 1: Запуск Appium

Перед виконанням тестів переконуємось, що Appium сервер запущено. Це можна зробити через консоль, ввести команду appium, або використовуючи

графічний інтерфейс Appium Desktop. Під час запуску необхідно вказати адресу та порт сервера, а також інші налаштування відповідно до потреб вашого тестування.

Крок 2: Запуск емулятора або підключення реального пристрою

Перед запуском тестів переконуємось, що емулятор або реальний, пристрій готовий до використання. Запустіть емулятор через Android Virtual Device Manager або підключіть реальний пристрій через USB. Перевіряємо, що пристрій має необхідні дозволи для виконання тестів.

Крок 3: Запуск тестів у Android Studio

Відкриваємо наш проект у Android Studio та переходимо до тестового класу, який будемо виконувати. Правою кнопкою миші натискайте на класі та вибирайте "Run". Android Studio автоматично ініціює виконання тестового сценарію.

Крок 4: Спостереження за виконанням

Спостерігаємо за ходом виконання тестових сценаріїв в консолі Android Studio. Тут є можливість відслідковувати інформацію про кожен тест, його результати та будь-які помилки, які можуть виникнути під час виконання.

Крок 5: Аналіз результатів

Після завершення виконання тестів потрібно ретельно переглянути результати. Android Studio надає зручні засоби для перегляду звітів тестування та виявлення проблем. Звертайте увагу на успішність кожного тесту та, в разі потреби, переглядайте детальнішу інформацію про помилки.

Крок 6: Виправлення помилок

У випадку, якщо тест не пройшов успішно, використовуйте отриману інформацію для виправлення помилок. Це може включати в себе внесення змін у вихідний код тестів або перевірку конфігурації Appium та інших параметрів тестування.

Виконуючи описані вище кроки, є можливість забезпечити ефективне тестування Android додатку за допомогою фреймворку Appium в середовищі

розробки Android Studio. Невід'ємною частиною цього процесу є систематичний аналіз результатів, що дозволить вчасно виявляти та виправляти потенційні проблеми в програмному забезпеченні.

4.1.2 Виконання тестових сценаріїв на фреймворку Katalon в середовищі розробки Katalon Studio

Крок 1: Відкриття та Налаштування Проекту в Katalon Studio

Перш за все, потрібно відкрити свій проект у Katalon Studio та переконатись, що необхідні бібліотеки та налаштування наявні. Вибираємо відповідний проект в розділі "Projects," а потім переходимо до "Test Explorer," де знаходяться наші тестові сценарії.

Крок 2: Вибір та Запуск Тестів

Відкриваємо наш тестовий сценарій, який хочемо виконати, та використовуємо опцію "Run" із контекстного меню. Katalon Studio автоматично вибере конфігурацію та оточення для запуску тесту.

Крок 3: Спостереження за Виконанням

Спостерігати за виконанням тестового сценарію в реальному часі, можна через вікно "Log Viewer." Тут ви знайдете докладну інформацію про кожен етап виконання тесту, включаючи результати та можливі проблеми.

Крок 4: Аналіз Результатів

Після завершення тестування ретельно аналізуємо отримані результати в "Log Viewer." Розглядаємо успішно виконані кроки, виявлені проблеми та додаткові деталі, які можуть бути корисними для виправлення помилок чи покращення тестового сценарію.

Крок 5: Виправлення Помилок та Покращення Тестів

У випадку виявлення помилок чи під час аналізу результатів, повертаємось до коду в Katalon Studio та вносимо необхідні зміни. Для цього використовуємо вбудовані інструменти для редагування коду та покращення тестових сценаріїв.

Застосовуючи вищезазначені кроки, забезпечимо ефективне виконання, спостереження та аналіз тестових сценаріїв в Katalon Studio. Це надасть можливість швидко реагувати на будь-які виявлені проблеми та постійно покращувати якість програмного забезпечення.

4.2 Детальний покроковий тестовий сценарій з наведенням коду

4.2.1 Авторизація у застосунку

Кроки тестування:

1. Запуск застосунку:

Відкриваємо застосунок Instagram на мобільному пристрої.

Appium:

```
.setApp("C:\\Appium\\instagram-288-1-0-22-66.apk")
```

Katalon:

```
Mobile.startExistingApplication('com.instagram.android')
```

Перевіряємо, чи відображається екран авторизації після завершення запуску застосунку.

Appium:

```
WebElement avto =
driver.findElement(AppiumBy.xpath("//android.view.View[@content-
desc=\"Увійти\"]"));
            if (avto.isDisplayed()) {
                System.out.println("Login page on screen");
            } else {
                System.out.println("There is no login page");
            }
```

Katalon:

```
if (Mobile.verifyElementVisible(findTestObject('Object
Repository/ Afrori/LoginScreen'), 15)){
        println('Login page on screen')
} else {
        println('There is no login page')
```

```
        FailureHandling.STOP_ON_FAILURE
    }
```

2. Введення даних користувача для авторизації:

Натискаємо на поле логін для подальшого вводу даних.

Appium:

```
    WebElement LoginSelection =
driver.findElement(AppiumBy.xpath("//android.widget.FrameLayout[
@resource-
id=\"com.instagram.android:id/layout_container_main\"]/android.w
idget.FrameLayout[1]/android.widget.FrameLayout/android.widget.F
rameLayout[1]/android.view.ViewGroup/android.view.ViewGroup/andr
oid.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup
[2]/android.view.ViewGroup/android.view.ViewGroup/android.view.V
iewGroup/android.widget.EditText"));
LoginSelection.click();
```

Katalon:

```
    Mobile.tap(findTestObject('Object Repository/ Aftori/
LoginSelection'), 0)
```

Перевірити поле логіну на реагування після нашого натискання по ньому.

Appium:

```
    WebElement Login =
driver.findElement(AppiumBy.xpath("//android.widget.Button[@cont
ent-desc=\"Очистить текст \"]"));
            if (Login.isDisplayed()) {
                System.out.println("Login field selection
successful");
            } else {
                System.out.println("The login field was not
highlighted");
            }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/Aftori/Login'), 15)){
```

```
println('Login field selection successful')
} else {
println('The login field was not highlighted')
FailureHandling.STOP_ON_FAILURE
}
```

Вводимо коректні дані для авторизації у поле логін.

Appium:

```
WebElement loginSelection =
driver.findElement(AppiumBy.xpath("//android.widget.FrameLayout[
@resource-
id=\"com.instagram.android:id/layout_container_main\"]/android.w
idget.FrameLayout[1]/android.widget.FrameLayout/android.widget.F
rameLayout[1]/android.view.ViewGroup/android.view.ViewGroup/andr
oid.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup
[1]/android.view.ViewGroup/android.view.ViewGroup/android.view.V
iewGroup/android.widget.EditText"));
            loginSelection.sendKeys("***********@ukr.net");
```

Katalon:

```
Mobile.setText(findTestObject('Object
Repository/Aftori/loginSelection'), **********@ukr.net', 0)
```

Перевірити поле логін на наявність введенних даних.

Appium:

```
WebElement loginSelection2 =
driver.findElement(AppiumBy.xpath("//android.widget.EditText[@te
xt=\"**********@ukr.net\"]"));
            if (loginSelection2.isDisplayed()) {
                System.out.println("The entered login is
displayed on the screen");
            } else {
                System.out.println("The entered login is not
displayed on the screen");
            }
```

Katalon:

```
     if (Mobile.verifyElementVisible(findTestObject('Object
Repository/Aftori/loginSelection2'), 15)){
          println('The entered login is displayed on the
screen')
     } else {
          println('The entered login is not displayed on the
screen')
          FailureHandling.STOP_ON_FAILURE
     }
```

Натискаємо на поле пароль для подальшого вводу даних.

Appium:

```
     WebElement Password =
driver.findElement(AppiumBy.xpath("//android.widget.FrameLayout[
@resource-
id=\"com.instagram.android:id/layout_container_main\"]/android.w
idget.FrameLayout[1]/android.widget.FrameLayout/android.widget.F
rameLayout[1]/android.view.ViewGroup/android.view.ViewGroup/andr
oid.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup
[2]/android.view.ViewGroup/android.view.ViewGroup/android.widget
.EditText"));
             Password.click();
```

Katalon:

```
     Mobile.tap(findTestObject('Object
Repository/Aftori/Password'), 0)
```

Перевірити поле пароль на реагування після нашого натискання по ньому.

Appium:

```
     WebElement Password2 =
driver.findElement(AppiumBy.xpath("//android.widget.Button[@cont
ent-desc=\"Показать пароль\"]/android.widget.ImageView"));
             if (Password2.isDisplayed()) {
                 System.out.println("Password field selection
successful");
             } else {
```

```
                System.out.println("The password field was not
highlighted");
            }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/Aftori/Password2'), 15)){
        println('Password field selection successful')
    } else {
        println('The password field was not highlighted')
        FailureHandling.STOP_ON_FAILURE
    }
```

Вводимо коректні дані для авторизації у поле пароль.

Appium:

```
    WebElement PasswordSelection =
driver.findElement(AppiumBy.xpath("//android.widget.FrameLayout[
@resource-
id=\"com.instagram.android:id/layout_container_main\"]/android.w
idget.FrameLayout[1]/android.widget.FrameLayout/android.widget.F
rameLayout[1]/android.view.ViewGroup/android.view.ViewGroup/andr
oid.view.ViewGroup/android.view.ViewGroup/android.view.ViewGroup
[2]/android.view.ViewGroup/android.view.ViewGroup/android.widget
.EditText"));
            PasswordSelection.sendKeys("**********");
```

Katalon:

```
    Mobile.setText(findTestObject('Object
Repository/Aftori/PasswordSelection'), **********', 0)
```

Перевірити поле пароль на наявність введених даних.

Appium:

```
    WebElement PasswordSelection2 =
driver.findElement(AppiumBy.xpath("//android.widget.EditText[@te
xt=\"**********\"]"));
            if (PasswordSelection2.isDisplayed()) {
                System.out.println("The entered password is
displayed on the screen");
```

```
            } else {
                System.out.println("The entered password is not
displayed on the screen");
            }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/Aftori/PasswordSelection2'), 15)){
        println('The entered password is displayed on the
screen')
    } else {
        println('The entered password is not displayed on the
screen')
        FailureHandling.STOP_ON_FAILURE
    }
```

3. Вхід під своїм обліковим записом:

Натискаємо на кнопку входу.

Appium:

```
    WebElement correct =
driver.findElement(AppiumBy.xpath("//android.widget.Button[@cont
ent-desc=\"Войти\"]"));
            correct.click();
```

Katalon:

```
    Mobile.tap(findTestObject('Object
Repository/Aftori/correct'), 0)
```

Перевіряємо, чи змінився екран авторизації на екран з можливістю збереження своїх даних для наступних авторизацій.

Appium:

```
    WebElement correct2 =
driver.findElement(AppiumBy.xpath("//android.view.View[@content-
desc=\"Сохранить данные для входа?\"]"));
            if (correct2.isDisplayed()) {
                System.out.println("Login details are
correct");
            } else {
```

```
                System.out.println("Login information is not
correct");
            }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/Aftori/correct2'), 15)){
        println('Login details are correct')
    } else {
        println('Login information is not correct')
        FailureHandling.STOP_ON_FAILURE
    }
```

4. Успішний вхід:

Натискаємо по кнопці «Відмовитись».

Appium:

```
    WebElement successful =
driver.findElement(AppiumBy.xpath("//android.widget.Button[@cont
ent-desc=\"Не сейчас\"]"));
            successful.click();
```

Katalon:

```
Mobile.tap(findTestObject('Object
Repository/Afrori/successful'), 0)
```

Перевіряємо, чи змінився екран на головний екран застосунку. Це також свідчить про успішний вхід.

Appium:

```
    WebElement successful2 =
driver.findElement(AppiumBy.accessibilityId("Дом"));
            if (successful2.isDisplayed()) {
                System.out.println("Authorization was
successful");
            } else {
                System.out.println("Authorization failed");
            }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/Aftori/successful2'), 15)){
        println('Authorization was successful')
    } else {
        println('Authorization failed')
        FailureHandling.STOP_ON_FAILURE
    }
```

4.2.2 Створення нової історії

Кроки тестування:

1. Запуск застосунку:

Відкрити застосунок Instagram на мобільному пристрої, якщо в застосунку нема авторизованого користувача, то потрібно авторизуватись під акаунтом тестового призначення.

Перевіряємо, чи відображається у нас на екрані головна сторінка.

Appium:

```
    WebElement Home2 =
driver.findElement(AppiumBy.accessibilityId("Дом"));
                if (Home2.isDisplayed()) {
                    System.out.println("Home page on screen");
                } else {
                    System.out.println("Home page is missing");
                }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/Home2'), 7)){
        println('Home page on screen')
    } else {
        println('Home page is missing')
        FailureHandling.STOP_ON_FAILURE
    }
```

2. Перехід до розділу історій:

Здійснюємо перехід до розділу створення історій, за допомогою свайпу з ліва на право, приблизно по центру екрана.

Appium:

```
final WebElement finger = new
PointerInput(PointerInput.Kind.TOUCH, "finger");
                WebElement start = new Point(164, 563);
                WebElement end = new Point (916, 557);
                WebElement swipe = new Sequence(finger, 1);

swipe.addAction(finger.createPointerMove(Duration.ofMillis(0),
                        PointerInput.Origin.viewport(),
start.getX(), start.getY()));
swipe.addAction(finger.createPointerDown(PointerInput.MouseButto
n.LEFT.asArg()));

swipe.addAction(finger.createPointerMove(Duration.ofMillis(1000)
,
                        PointerInput.Origin.viewport(),
end.getX(), end.getY()));
swipe.addAction(finger.createPointerUp(PointerInput.MouseButton.
LEFT.asArg()));
                driver.perform(Arrays.asList(swipe));
```

Katalon:

```
Mobile.swipe(100, 400, 900, 400)
```

Перевіряємо, чи відкривається сторінка створення історії.

Appium:

```
WebElement stories =
driver.findElement(AppiumBy.xpath("//android.widget.TextView[@re
source-
id=\"com.android.permissioncontroller:id/permission_message_vide
o\"]"));
                if (stories.isDisplayed()) {
```

```
                    System.out.println("The window for
selecting content for stories is open with a request for
access");
                } else {
                    System.out.println("The window for
selecting content for stories could not be opened with an access
request");
          }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/access'), 7)){
         println('The window for selecting content for stories
is open with a request for access')
    } else {
         println('The window for selecting content for stories
could not be opened with an access request')
         FailureHandling.STOP_ON_FAILURE
    }
```

3. Дозвіл до файлів мобільного пристрою:

Перед тим як остаточно потрапити до розділу створення історії, необхідно надати доступ застосунку до фото та відео контенту, доступ до камери та мікрофону.

Натискаємо на кнопку «Тільки цього разу».

Appium:

```
    WebElement AccessVideo =
driver.findElement(AppiumBy.id("com.android.permissioncontroller
:id/permission_allow_one_time_button"));
                AccessVideo.click();
```

Katalon:

```
    Mobile.tap(findTestObject('Object
Repository/AddStory/AccessVideo'), 0)
```

Перевіряємо, чи було дозволено застосунку відео контент, що був розташований на пристрої.

Appium:

```
WebElement video =
driver.findElement(AppiumBy.xpath("//android.widget.TextView[@re
source-
id=\"com.android.permissioncontroller:id/permission_message_micr
ophone\"]"));
                if (video.isDisplayed()) {
                    System.out.println("Access to video
allowed");
                } else {
                    System.out.println("Access to video is
denied");
                }
```

Katalon:

```
if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/AccessVideo2'), 7)){
        println('Access to video allowed')
} else {
        println('Access to video is denied')
        FailureHandling.STOP_ON_FAILURE
}
```

Натискаємо на кнопку «Тільки цього разу».

Appium:

```
WebElement AccessMicrophone =
driver.findElement(AppiumBy.id("com.android.permissioncontroller
:id/permission_allow_one_time_button"));
                AccessMicrophone.click();
```

Katalon:

```
Mobile.tap(findTestObject('Object
Repository/AddStory/AccessMicrophone'), 0)
```

Перевіряємо, чи був дозволений застосунку мікрофон пристрою.

Appium:

```
WebElement microphone =
driver.findElement(AppiumBy.xpath("//android.widget.TextView[@re
```

```
source-
id=\"com.android.permissioncontroller:id/permission_message_file
\"]"));
                if (microphone.isDisplayed()) {
                    System.out.println("Access to microphone
allowed");
                } else {
                    System.out.println("Access to microphone is
denied");
                }
```

Katalon:

```
if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/AccessMicrophone2'), 7)){
     println('Access to microphone allowed')
} else {
     println('Access to microphone is denied')
     FailureHandling.STOP_ON_FAILURE
}
```

Натискаємо на кнопку «Дозволити»

Appium:

```
WebElement AccessFile =
driver.findElement(AppiumBy.id("com.android.permissioncontroller
:id/permission_allow_button"));
                AccessFile.click();
```

Katalon:

```
Mobile.tap(findTestObject('Object
Repository/AddStory/AccessFile'), 0)
```

Перевіряємо, чи були дозволені застосунку файли, що розташовані на пристрої.

Appium:

```
WebElement file =
driver.findElement(AppiumBy.xpath("//android.widget.Button[@reso
urce-id=\"com.instagram.android:id/ar_effect_in_tray_icon\"]"));
                if (file.isDisplayed()) {
```

```
                    System.out.println("Access to file
allowed");
                } else {
                    System.out.println("Access to file is
denied");
                }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/AccessFile2'), 7)){
         println('Access to file allowed')
    } else {
         println('Access to file is denied')
         FailureHandling.STOP_ON_FAILURE
    }
```

4. Знімання фото або відео:

Використовуючи камеру пристрою, зробити фото або відео для нової історії. Якщо немає змоги увімкнути камеру, то потрібно взяти відео або фото з галереї пристрою.

Натискаємо кнопку «Галерея».

Appium:

```
    WebElement Gallery =
driver.findElement(AppiumBy.accessibilityId("Галерея"));
                Gallery.click();
```

Katalon:

```
    Mobile.tap(findTestObject('Object
Repository/AddStory/Gallery'), 0)
```

Перевіряємо, чи відкрилась сторінка галереї.

Appium:

```
    WebElement Gallery2 =
driver.findElement(AppiumBy.xpath("//android.widget.Button[@cont
ent-desc=\"Відкрити камеру\"]"));
                if (Gallery2.isDisplayed()) {
                    System.out.println("Gallery is open");
                } else {
```

```
                    System.out.println("Galere is missing");
                }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/Gallery2'), 7)){
        println('Gallery is open')
    } else {
        println('Galere is missing')
        FailureHandling.STOP_ON_FAILURE
    }
```

Вибираємо та натискаємо на першу ж фотографію з галереї.

Appium:

```
    WebElement photoSuccessful =
driver.findElement(AppiumBy.xpath("//android.view.View[@content-
desc=\"Test3\"]"));
                photoSuccessful.click();
```

Katalon:

```
    Mobile.tap(findTestObject('Object
Repository/AddStory/photoSuccessful'), 0)
```

Перевірити, чи саме та фотографія або відео було вибрано, що відображається на екрані.

Appium:

```
    WebElement choice =
driver.findElement(AppiumBy.xpath("//android.view.View[@resource
-id=\"com.instagram.android:id/focus_view/content-
desc=\"Test3\"]"));
                if (choice.isDisplayed()) {
                    System.out.println("The selection of the
desired photo was successful");
                } else {
                    System.out.println("Selecting the required
photo failed");
                }
```

Katalon:

```
     if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/photoSuccessful2'), 7)){
          println('The selection of the desired photo was
successful')
     } else {
          println('Selecting the required photo failed')
          FailureHandling.STOP_ON_FAILURE
     }
```

5. Вибір аудиторії:

Вибрати аудиторію Близькі друзі, саме ця аудиторія буде мати змогу переглянути цю історію.

Натискаємо на кнопку «Близькі друзі»

Appium:

```
     WebElement publishing =
driver.findElement(AppiumBy.id("com.android.permissioncontroller
:id/permission_allow_one_time_button"));
                publishing.click();
```

Katalon:

```
     Mobile.tap(findTestObject('Object
Repository/AddStory/publishing'), 0)
```

Переконатися, що вибір аудиторії пройшов успішно.

Appium:

```
     WebElement publishing2 =
driver.findElement(AppiumBy.Id("com.android.permissioncontroller
:id/permission_allow _time_button "));
                if (publishing2.isDisplayed()) {
                    System.out.println("Completing the
publishing process is complete");
                } else {
                    System.out.println("Completing the
publishing process failed");
                }
```

Katalon:

```
     if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/publishing2'), 7)){
          println('Completing the publishing process is
complete')
     } else {
          println('Completing the publishing process failed')
          FailureHandling.STOP_ON_FAILURE
     }
```

6. Опублікування історії:

Торкнутися фотографії або натискати кнопку "Опублікувати" для розміщення нової історії на своєму обліковому записі.

Appium:

```
     WebElement publishing =
driver.findElement(AppiumBy.accessibilityId("Поділитись істо-
рією"));
                publishing.click();
```

Katalon:

```
     Mobile.tap(findTestObject('Object
Repository/AddStory/StoryPublished'), 0)
```

Перевіряємо, що після натискання на кнопку "Опублікувати", в нас на екрані відображається головна сторінка.

Appium:

```
     WebElement house2 =
driver.findElement(AppiumBy.accessibilityId("Дім"));
                if (house2.isDisplayed()) {
                    System.out.println("Completing the
publishing process is complete");
                } else {
                    System.out.println("Completing the
publishing process failed");
                }
```

Katalon:

```
     if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/StoryPublished2'), 7)){
```

```
println('Story published successfully')
} else {
println('No history')
FailureHandling.STOP_ON_FAILURE
}
```

7. Перевірка опублікування історії:

На головній сторінці, у стрічці новин знаходимо свою історію, та відкриваємо її.

Appium:

```
WebElement StoryPublished =
driver.findElement(AppiumBy.accessibilityId("История pasenko.333
в столбце 0. Непросмотренное."));
                StoryPublished.click();
```

Katalon:

```
if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/home2'), 7)){
println('Home page on screen ')
} else {
println('Home page is missing')
FailureHandling.STOP_ON_FAILURE
}
```

Перевірити успішне опублікування та коректне відображення, тієї самої фотографії чи того самого відео, що ми знімали або брали з галереї.

Appium:

```
WebElement story =
driver.findElement(AppiumBy.accessibilityId("//android.widget.Fr
ameLayout[@resource-
id=\"com.instagram.android:id/reel_viewer_image_view\"]/android.
widget.ImageView=\"Test3\""));
                if (story.isDisplayed()) {
                    System.out.println("Story published
successfully");
                } else {
                    System.out.println("No history");
```

```
                }
```

Katalon:

```
     if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/StoryPublished2'), 7)){
          println('Story published successfully')
     } else {
          println('No history')
          FailureHandling.STOP_ON_FAILURE
     }
```

Закриваємо історію за допомогою використання можливостей пристрою pressBack. Щоб визвати цей процес, потрібно зробити свайп з ліва на право.

Appium:

```
     final WebElement finger = new
PointerInput(PointerInput.Kind.TOUCH, "finger");
                WebElement start = new Point(22, 988);
                WebElement end = new Point (996, 975);
                WebElement swipe = new Sequence(finger, 1);

swipe.addAction(finger.createPointerMove(Duration.ofMillis(0),
                        PointerInput.Origin.viewport(),
start.getX(), start.getY()));
swipe.addAction(finger.createPointerDown(PointerInput.MouseButto
n.LEFT.asArg()));
swipe.addAction(finger.createPointerMove(Duration.ofMillis(1000)
,
                        PointerInput.Origin.viewport(),
end.getX(), end.getY()));
swipe.addAction(finger.createPointerUp(PointerInput.MouseButton.
LEFT.asArg()));
                driver.perform(Arrays.asList(swipe));
```

Katalon:

```
     Mobile.swipe(1400, 700, 500, 700)
```

Перевіряємо, чи після pressBack на екрані відображається головна сторінка.

Appium:

```
WebElement house3 =
driver.findElement(AppiumBy.accessibilityId("Дом"));
                if (house3.isDisplayed()) {
                    System.out.println("Return to home page
successful");
                } else {
                    System.out.println("Return to home page
failed");
                }
```

Katalon:

```
if (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/Return'), 7)){
        println('Return to home page successful')
} else {
        println('Return to home page failed')
        FailureHandling.STOP_ON_FAILURE
}
```

4.2.3 Перегляд контенту іншого користувача у його профілі

Кроки тестування:

1. Запуск застосунку:

Відкрити застосунок Instagram на мобільному пристрої, якщо в застосунку нема авторизованого користувача, то потрібно авторизуватись під акаунтом тестового призначення.

Перевіряємо, чи відображається на екрані головна сторінка.

Appium:

```
WebElement house =
driver.findElement(AppiumBy.accessibilityId("Дом"));
                if (house.isDisplayed()) {
                    System.out.println("Home page on screen");
```

```
                } else {
                    System.out.println("Home page is missing");
                }
```

Katalon:

```
if       (Mobile.verifyElementVisible(findTestObject('Object
Repository/AddStory/Home2'), 7)){
     println('Home page on screen')
} else {
     println('Home page is missing')
     FailureHandling.STOP_ON_FAILURE
}
```

2. Сторінка пошуку користувача:

На головній сторінці, у нижньому меню, натиснути кнопку «Пошук».

Appium:

```
WebElement Search =
driver.findElement(AppiumBy.accessibilityId("Поиск и интерес-
ное"));
                Search.click();
```

Katalon:

```
Mobile.tap(findTestObject('Object
Repository/ViewingUserPhoto/SearchSuccessful'), 0)
```

Перевіряємо, чи відображається на екрані сторінка пошуку.

Appium:

```
WebElement shot =
driver.findElement(AppiumBy.xpath("//android.widget.EditText[@re
source-
id=\"com.instagram.android:id/action_bar_search_edit_text\"]"));
                if (shot.isDisplayed()) {
                    System.out.println("Search page is open");
                } else {
                    System.out.println("The search page is
missing from the screen");
                }
```

Katalon:

```
if       (Mobile.verifyElementVisible(findTestObject('Object
Repository/ViewingUserPhoto/SearchSuccessful2'), 7)){
          println('Search field selection successful')
     } else {
          println('The search field was not highlighted')
          FailureHandling.STOP_ON_FAILURE
     }
```

3. Введення імені користувача

Натискаємо на поле пошуку для подальшого вводу даних.

Appium:

```
     WebElement SearchSuccessful =
driver.findElement(AppiumBy.id("com.instagram.android:id/action_
bar_search_edit_text"));
                SearchSuccessful.click();
```

Katalon:

```
     Mobile.tap(findTestObject('Object
Repository/ViewingUserPhoto/SearchSuccessful'), 0)
```

Перевіряємо поле пошуку на реагування після нашого натискання по ньому.

Appium:

```
     WebElement search =
driver.findElement(AppiumBy.xpath("//android.widget.TextView[@te
xt=\"Поиск\"]"));
                if (search.isDisplayed()) {
                    System.out.println("Search field selection
successful");
                } else {
                    System.out.println("The search field was
not highlighted");
                }
```

Katalon:

```
     if       (Mobile.verifyElementVisible(findTestObject('Object
Repository/ViewingUserPhoto/search'), 7)){
          println('Search field selection successful')
```

```
} else {
    println('The search field was not highlighted')
    FailureHandling.STOP_ON_FAILURE
}
```

Вносимо ім’я користувача у пошукову стрічку.

Appium:

```
WebElement username =
driver.findElement(AppiumBy.id("com.instagram.android:id/action_
bar_search_edit_text"));
                username.sendKeys("akaman3");
```

Katalon:

```
Mobile.setText(findTestObject('Object
Repository/ViewingUserPhoto/username'), 'akaman3', 0)
```

Перевіряємо, чи відображається конкретний користувач, якого плануємо переглядати серед списку варіантів результатів пошуку.

Appium:

```
WebElement username2 =
driver.findElement(AppiumBy.xpath("//android.widget.Button[@cont
ent-desc=\"akaman3\"]"));
                if (username2.isDisplayed()) {
                    System.out.println("The entered username is
displayed on the screen");
                } else {
                    System.out.println("The entered username is
not displayed on the screen");
                }
```

Katalon:

```
if (Mobile.verifyElementVisible(findTestObject('Object
Repository/ViewingUserPhoto/username2'), 7)){
        println('The entered username is displayed on the
screen')
} else {
        println('The entered username is not displayed on the
screen')
```

```
        FailureHandling.STOP_ON_FAILURE
    }
```

4. Перехід на профіль користувача:

Натискаємо на ім'я користувача в результатах пошуку, щоб перейти на його профіль.

Appium:

```
    WebElement akaman =
driver.findElement(AppiumBy.xpath("(//android.widget.LinearLayou
t[@resource-
id=\"com.instagram.android:id/row_search_user_info_container\"])
[1]/android.widget.LinearLayout"));
                akaman.click();
```

Katalon:

```
    Mobile.tap(findTestObject('Object
Repository/ViewingUserPhoto/akaman'), 0)
```

Перевіряємо, чи відображається сторінка того самого користувача, що був введений у поле пошуку.

Appium:

```
    WebElement profile =
driver.findElement(AppiumBy.xpath("//android.widget.TextView[@co
ntent-desc=\"akaman3\"]"));
                if (profile.isDisplayed()) {
                    System.out.println("Profile page akaman3
successfully opened");
                } else {
                    System.out.println("No profile page
akaman3");
                }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/ViewingUserPhoto/akaman2'), 7)){
        println('Profile page akaman3 successfully opened')
    } else {
        println('No profile page akaman3')
```

```
        FailureHandling.STOP_ON_FAILURE
    }
```

5. Перегляд фото та відео:

Відкриваємо перший бокс з контентом на сторінці користувача.

Appium:

```
    WebElement OpeningPhoto =
driver.findElement(AppiumBy.accessibilityId("Фото Алексей в
строке 1, столбце 1"));
                OpeningPhoto.click();
```

Katalon:

```
    Mobile.tap(findTestObject('Object
Repository/ViewingUserPhoto/OpeningPhoto'), 0)
```

Перевірити коректність відображення контенту.

Appium:

```
    WebElement successful =
driver.findElement(AppiumBy.xpath("///android.widget.TextView[@c
ontent-desc=\"Публикации\"]"));
                if (successful.isDisplayed()) {
                    System.out.println("Opening any photo was
successful");
                } else {
                    System.out.println("Opening any photo did
not open");
                }
```

Katalon:

```
    if (Mobile.verifyElementVisible(findTestObject('Object
Repository/ViewingUserPhoto/OpeningPhoto2'), 7)){
        println('Opening any photo was successful')
    } else {
        println('Opening any photo did not open')
        FailureHandling.STOP_ON_FAILURE
    }
```

6. Поставити лайк:

Поставити лайк під цією, першою фотографією, переконуючись, що кнопка «Лайк» змінить свій колір на червоний.

Натискаємо на кнопку «Лайк».

Appium:

```
WebElement like = 
driver.findElement(AppiumBy.xpath("(//android.widget.ImageView[@
content-desc=\"Нравится\"])[1]"));
                like.click();
```

Katalon:

```
Mobile.tap(findTestObject('Object 
Repository/ViewingUserPhoto/like'), 0)
```

Перевірити, чи була зафіксована взаємодія кнопки «Лайк» контенту.

Appium:

```
WebElement from = 
driver.findElement(AppiumBy.xpath("//android.widget.FrameLayout[
@content-desc=\"Фото 2 из 2 от Алексей, Нравится: 36\"]"));
                if (from.isDisplayed()) {
                    System.out.println("Changing the color of 
the like button was successful");
                } else {
                    System.out.println("Changing the color of 
the like button failed");
                }
```

Katalon:

```
if (Mobile.verifyElementVisible(findTestObject('Object 
Repository/ViewingUserPhoto/like2'), 7)){
        println('Changing the color of the like button was 
successful')
} else {
        println('Changing the color of the like button 
failed')
        FailureHandling.STOP_ON_FAILURE
}
```

7. Вихід із профілю:

Повернутися на головну сторінку застосунку.

Натискаємо кнопку «Дім».

Appium:

```
WebElement home3 =
driver.findElement(AppiumBy.accessibilityId("Дом"));
                home3.click();
```

Katalon:

```
Mobile.tap(findTestObject('Object
Repository/ViewingUserPhoto/home'), 0)
```

Перевіряємо відображення на екрані головної сторінки.

Appium:

```
WebElement house3 =
driver.findElement(AppiumBy.accessibilityId("Дом"));
                if (house3.isDisplayed()) {
                    System.out.println("Return to home page
successful");
                } else {
                    System.out.println("Return to home page
failed");
                }
```

Katalon:

```
if (Mobile.verifyElementVisible(findTestObject('Object
Repository/ViewingUserPhoto/home2'), 7)){
          println('Return to home page successful')
     } else {
          println('Return to home page failed')
          FailureHandling.STOP_ON_FAILURE
     }
```

4.3 Аналіз зібраних даних

Під час дослідження використання високотехнологічних фреймворків Appium та Katalon для ефективного автоматизованого тестування мобільного застосунку Instagram на платформі Android, застосований науково-дослідний

підхід надав унікальну можливість вивчення та аналізу важливих та значущих даних, з метою розуміння функціональних можливостей та ефективності вибраних інструментів.

Вибірка досліджуваних фреймворків

При виборі фреймворків визначальними факторами стали їхні визнаність та широкий рівень популярності в сучасному програмному середовищі. Appium та Katalon виявилися ключовими учасниками нашого дослідження завдяки їхній широкій розповсюдженості та відмінним можливостям для ефективного вирішення завдань автоматизованого тестування. Їхні високі стандарти та впевненість у функціональності дозволяють їм вирізнятися в галузі тестування програмного забезпечення.

Підготовка тестового сценарію

Ключовим та необхідним етапом став фундаментальний аналіз та розробка детального тестового сценарію, який широко охоплює величезний спектр функціональностей та складових мобільного застосунку Instagram. Цей сценарій враховує різноманіття можливостей, що надає кожен фреймворк, та був ретельно розроблений для максимального висвітлення усіх аспектів та потенціалу кожного інструменту на подальших етапах проведеного дослідження.

Виконання тестування

З великою увагою та систематичністю був впроваджений докладно розроблений тестовий сценарій для кожного вибраного фреймворку. Під час виконання цього сценарію здійснювалось не лише збирання даних про надійність та час виконання тестів, але і вивчення інших критичних параметрів. Отриманий набір об'єктивних даних удосконалив можливість подальшого порівняння між фреймворками, забезпечуючи надійну основу для об'єктивного аналізу їхньої ефективності та функціональності.

4.4 Результати порівняння фреймворків для автоматизованого тестування мобільних застосунків за критеріями

Дослідження та порівняння фреймворків для автоматизованого тестування мобільних застосунків є важливою темою в сучасному інформаційному середовищі. Ключовим аспектом є вибір підходящого фреймворку для автоматизованого тестування, оскільки це може суттєво вплинути на якість та швидкість розробки мобільних застосунків. Порівняння різних фреймворків дозволяє розробникам та тестувальникам краще розуміти їхні переваги та обмеження, сприяючи обґрунтованому вибору на оптимальному етапі проекту.

Зараз будуть представлені результати мого власного дослідження, як кожен фреймворк показав себе в кожному з критеріїв.

Список критеріїв для порівняння:

1. Крос-платформенність та сумісність
2. Мова програмування та розширюваність
3. Інтерфейс користувача та підтримка UI-тестування
4. Легкість вивчення і використання
5. Швидкодія та стабільність
6. Можливості взаємодії з реальними пристроями та емуляторами
7. Зручність підтримки тестів
8. Можливості роботи з різними версіями Android
9. Надійність виконання тестів:
10. Підтримка спільноти та оновлення
11. Вартість та ліцензія

4.4.1 Результати дослідження фреймворку Appium

Appium - це крос-платформний фреймворк для автоматизованого тестування мобільних додатків. Розглянемо ключові характеристики, що були спостереженні під час дослідження фреймворку Appium:

1. Крос-платформенність та сумісність:

- Appium демонструє високу крос-платформенність, що дозволяє ефективно тестувати як Android, так і iOS додатки.

- Уніфіковане API та підтримка різних мов програмування забезпечують гнучкість для розробки тестів на обох мобільних платформах.

2. Мова програмування та розширюваність:
   - Appium визначається розширюваністю через підтримку різних мов, таких як Java, Python, Ruby, JavaScript і C#.
   - Це дозволяє розробникам та тестувальникам використовувати мову, з якою вони вже знайомі, що полегшує взаємодію та прискорює розробку тестів.
3. Інтерфейс користувача та підтримка UI-тестування:
   - Appium володіє гнучкістю та універсальністю в сфері UI-тестування, що спрощує розробку та підтримку тестів на обох платформах.
   - Можливість використовувати один тестовий скрипт для Android і iOS спрощує роботу тестувальників.
4. Легкість вивчення і використання:
   - Appium визначається легкістю вивчення завдяки простому API та підтримці різних мов програмування.
   - Зручний синтаксис та докладна документація полегшують освоєння платформи, особливо для тих, хто вже працював з Selenium.
5. Швидкодія та стабільність:
   - Appium вражає високою швидкодією та стабільністю на обох платформах, використовуючи внутрішні аспекти мобільних операційних систем.
6. Можливості взаємодії з реальними пристроями та емуляторами:
   - Appium гнучко взаємодіє з реальними пристроями та різними емуляторами для тестування на обох платформах.
7. Зручність підтримки тестів:

- Appium надає зручний підхід до підтримки тестів для крос-платформенних проектів, дозволяючи використовувати той самий код для Android та iOS.

8. Можливості роботи з різними версіями Android:
    - Appium володіє ефективною можливістю працювати з різними версіями операційної системи Android.
9. Надійність виконання тестів:
    - Appium відзначається високою надійністю виконання тестів завдяки крос-платформенній підтримці.
10. Підтримка спільноти та оновлення:
    - Appium користується активною підтримкою великої спільноти відкритого програмного забезпечення, що забезпечує регулярні оновлення та виправлення помилок.
11. Вартість та ліцензія:
    - Appium безкоштовний та відкритий, роблячи його економічно доступним для команд з обмеженим бюджетом.
    - Appium видається надзвичайно потужним, гнучким та економічно вигідним фреймворкам для автоматизованого тестування мобільних додатків, забезпечуючи високий рівень продуктивності та надійності.

Appium видається надзвичайно потужним, гнучким та економічно вигідним фреймворкам для автоматизованого тестування мобільних додатків, забезпечуючи високий рівень продуктивності та надійності.

4.4.2 Результати дослідження фреймворку Katalon

Katalon - це фреймворк для автоматизованого тестування, який пропонує комбінацію безкоштовного та платного функціоналу. Розглянемо ключові характеристики, що були спостереженні під час дослідження фреймворку Katalon:

1. Крос-платформенність та сумісність:

- Katalon пропонує крос-платформенність для тестування веб-додатків, мобільних додатків та API.
- Його сумісність включає підтримку різних браузерів для веб-тестування та підтримку Android і iOS для мобільного тестування.

2. Мова програмування та розширюваність:

- Katalon використовує Groovy для написання тестів, що може бути зручним для тих, хто вже володіє Java.
- Фреймворк також підтримує інтеграцію з Java та C#.

3. Інтерфейс користувача та підтримка UI-тестування:

- Katalon Studio надає графічний інтерфейс для створення та управління тестовими сценаріями.
- Забезпечує підтримку для UI-тестування веб-додатків та мобільних додатків.

4. Легкість вивчення і використання:

- Katalon Studio відомий своєю простотою використання, зокрема завдяки графічному інтерфейсу.
- Можливість використання Groovy спрощує синтаксис для написання тестів.

5. Швидкодія та стабільність:

- Швидкість та стабільність Katalon залежать від складності тестових сценаріїв та типу додатків, які тестуються.

6. Можливості взаємодії з реальними пристроями та емуляторами:

- Katalon дозволяє тестувати на реальних пристроях та емуляторах для мобільних додатків.

7. Зручність підтримки тестів:

- Графічний інтерфейс Katalon Studio робить підтримку тестів більш зручною для менеджерів та тестувальників.

8. Можливості роботи з різними версіями Android:

- Katalon зазвичай надає підтримку для різних версій Android, але слід перевірити документацію для конкретних деталей.

9. Надійність виконання тестів:

- Надійність Katalon залежить від правильної конфігурації та розробки тестів.

10. Підтримка спільноти та оновлення:

- Katalon має активну спільноту, але рівень підтримки може варіюватися в залежності від версії та проблеми.

11. Вартість та ліцензія:

- Katalon пропонує Freemium-модель, що означає базовий безкоштовний функціонал, але з можливістю розширення функціоналу за плату.

Загальною перевагою Katalon є його безкоштовний функціонал, простота використання та графічний інтерфейс для тестування. Однак важливо враховувати конкретні потреби та характеристики проекту при виборі між Appium та Katalon.

4.4.3 Підсумки дослідження

Appium та Katalon - це два різні фреймворки для автоматизованого тестування мобільних додатків, кожен з яких має свої особливості, переваги та недоліки.

Appium є потужним фреймворком для автоматизованого тестування мобільних додатків. Він виділяється крос-платформенність, гнучкістю у виборі мов програмування та ефективністю UI-тестування. Його швидкодія та стабільність роблять його ідеальним вибором для великих та складних проектів. Appium також вражає своєю гнучкістю у взаємодії з реальними пристроями та емуляторами, а також можливістю працювати з різними версіями Android. Легкість підтримки тестів та безкоштовність разом із підтримкою активної спільноти відкритого програмного забезпечення роблять Appium зручним та економічно вигідним вибором для розробників та тестувальників.

Katalon привертає увагу своєю доступністю та безкоштовністю базового функціоналу, роблячи його привабливим для початківців та проектів з обмеженим бюджетом. Фреймворк має розширену підтримку для різних типів тестування, включаючи API та веб-додатки, роблячи його універсальним інструментом. Інтуїтивний інтерфейс користувача спрощує автоматизацію, а можливість запису та відтворення тестів полегшує їх створення. Однак обмеження безкоштовної версії та можлива необхідність комерційної ліцензії для підтримки та оновлень можуть вплинути на його придатність для великих проектів.

Appium визначається гнучкістю та широкою підтримкою, Katalon приваблює доступністю та універсальністю. Вибір між ними залежить від конкретних потреб та умов проекту. Для більш зручного перегляду та розуміння різниці між цими фреймворками наведено таблицю 1 з їх можливостями.

Таблиця 1 — Детальна різниця між фреймворками Appium, Katalon

Критерії	Appium	Katalon
Крос-платформенність та сумісність	Висока (Android та iOS)	Обмежена (Android, обмежена на iOS у безкоштовній версії)
Мова програмування та розширюваність	Java, Python, Ruby, JavaScript, C#	Java, Groovy
Інтерфейс користувача та підтримка UI-тестування	Універсальний, гнучкий, підтримка обох платформ	Широкі можливості для UI-тестування, але можливості обмежені в безкоштовній версії для iOS
Легкість вивчення і використання	Легкий, зручний синтаксис, докладна документація	Середня, GUI для швидкої розробки та інтеграція з іншими інструментами
Швидкодія та стабільність	Висока	Висока
Можливості взаємодії з реальними пристроями та емуляторами	Гнучкість у взаємодії з реальними пристроями та емуляторами на обох платформах	Спеціалізована у роботі з реальними пристроями та емуляторами на обох платформах
Зручність підтримки тестів	Висока	Середня

Можливості роботи з різними версіями Android	Висока	Середня
Надійність виконання тестів	Висока	Середня
Підтримка спільноти та оновлення	Активна спільнота з частими оновленнями	Активна спільнота з частими оновленнями
Вартість та ліцензія	Безкоштовний та відкритий	Безкоштовний (Freemium)

4.5 Рекомендації щодо використання фреймворків

Підсумовуючи проведений аналіз, можна надати конкретні рекомендації для використання фреймворків в різних ситуаціях:

Для проектів, де найважливішою характеристикою є швидкодія та стабільність, рекомендується розглянути використання Appium або Espresso. Appium - це крос-платформний фреймворк, який дозволяє тестувати додатки на різних платформах, включаючи Android. Його популярність полягає в тому, що він підтримує різні мови програмування та може працювати з реальними пристроями та емуляторами. З іншого боку, якщо основним пріоритетом є зручність утримання та роботи з тестами, то оптимальним вибором може стати Katalon. Цей фреймворк надає зручний графічний інтерфейс для створення та управління тестовими сценаріями. Крім того, він підтримує запис та відтворення тестів, що полегшує процес створення та модифікації тестових сценаріїв.

Якщо акцент на інтерфейсі користувача та UI-тестуванні, то рекомендується розглянути використання Appium. Appium, надає гнучкість у роботі з різними платформами та мовами програмування, що дозволяє ефективно взаємодіяти з різними елементами UI на Android-пристроях.

Якщо команда веде розробку на Java або Kotlin, то важливо мати можливість використовувати одну мову програмування для тестування на обох

платформах (Android та iOS). Appium може стати оптимальним рішенням. Він дозволяє писати тести на різних мовах програмування та підтримує кросплатформеність.

Якщо ж основний фокус спрямований на кросплатформеність та максимальну гнучкість у виборі мови програмування, Appium залишається висок оцінним варіантом. Його підтримка різних мов програмування та можливість взаємодії з реальними пристроями робить його привабливим вибором для проектів з різними вимогами.

Враховуючи вищезазначені приклади, розробники можуть здійснювати обґрунтований вибір фреймворку відповідно до специфіки свого проекту та потреб бізнесу.

4.6 Висновок до розділу 4

У підсумку виконання тестових сценаріїв на фреймворка Appium та Katalon відзначається значний прогрес у напрямку автоматизації та забезпечення якості мобільних застосунків. Проведені порівняльні аналізи функціональності, продуктивності та зручності використання показують, що обидва фреймворки мають свої переваги та обмеження.

В ході тестування фреймворків було виявлено, що Appium відзначається великою гнучкістю та підтримкою різних мобільних платформ. Його можливість взаємодії з різними мовами програмування робить його універсальним рішенням для команд, які використовують різні технології розробки. Однак, враховуючи цю гнучкість, виникають труднощі у встановленні та конфігурації.

З іншого боку, Katalon вражає своєю простотою використання та інтегрованістю з Katalon Studio, що спрощує процес автоматизації для менших команд та початківців. Однак, його обмежена підтримка мов програмування та відсутність підтримки для деяких платформ, може стати перешкодою для проектів зі складним стеком технологій.

Результати дослідження вказують на те, що обираючи між Appium та Katalon, важливо враховувати конкретні потреби та обмеження проекту. Правильний вибір фреймворку допоможе забезпечити ефективне та стабільне автоматизоване тестування мобільних застосунків у вашому проекті.

ВИСНОВКИ

У процесі написання даної дипломної роботи був виконаний огляд сучасних мобільних платформ, типів застосунків та інструментів розробки.

Досліджені принципи автоматизованого тестування мобільних застосунків, загальні вимоги до інструментів тестування, визначені особливості ідентифікації елементів інтерфейсу та тестування користувацького введення.

Були визначені інструменти автоматизованого тестування для порівняльного дослідження: Appium, Katalon та Espresso. Було обрано мобільний застосунок – Android клієнт Instagram для дослідження.

Розроблені три тестові сценарії, кожен з яких містить перевірку основної функціональності.

Було виконане розгортання середовищ розробки для кожного з фреймворків, та створені відповідні проекти.

Дослідженні процедури експорту тестів, рефакторингу коду та виконання тестових сценаріїв. Аналіз результатів тестування дозволив сформулювати переваги та недоліки кожного інструменту.

Напрацьовані рекомендації визначенні у різних випадках кращого інструменту для тестування з досліджених.

Загалом, виконання дипломної роботи допомогло мені глибше зрозуміти принципи автоматизованого тестування мобільних застосунків та визначити критерії для ефективного вибору інструментів автоматизації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Масенко О.Є., магістрант, Кломоєць Г.П., канд. ф.-м. наук, доц. — науковий керівник. Порівняльне дослідження технологій тестування мобільних застосунків. *Молода наука-2023* : зб. наук. праць студентів, аспірантів, докторантів і молодих вчених. Запоріжжя : ЗНУ, 2023. Т. 5. С. 111–113.

2. Пасенко О.Є., магістрант, Кломоєць Г.П., канд. ф.-м. наук, доц. — науковий керівник. Порівняльне дослідження технологій тестування мобільних застосунків. Матеріали III всеукраїнської науково-практичної конференції за участю молодих науковців. Запоріжжя : ЗНУ, 2023. С. 126–127.

3. Android Programming: The Big Nerd Ranch Guide (5th Edition) | Big Nerd Ranch. *Big Nerd Ranch*. URL: https://bignerdranch.com/books/android-programming-the-big-nerd-ranch-guide-5th-edition/ (дата звернення: 21.06.2023).

4. Featured | Apple Developer Documentation. *Apple Developer Documentation*. URL: https://developer.apple.com/documentation/ (дата звернення: 10.12.2023).

5. iOS Programming: The Big Nerd Ranch Guide (7th Edition) | Big Nerd Ranch. *Big Nerd Ranch*. URL: https://bignerdranch.com/books/ios-programming-the-big-nerd-ranch-guide-7th-edition/ (дата звернення: 21.06.2023).

6. Windows Documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/en-us/windows/ (дата звернення: 21.06.2023).

7. Microsoft Launcher vs Samsung One UI: Which Launcher Is Better for You. *Guiding Tech*. URL: https://www.guidingtech.com/microsoft-launcher-vs-samsung-one-ui-better-launcher-comparison/ (дата звернення: 22.06.2023).

8. Hybrid vs. Native App Development: Pros and Cons of Each Option. *Cleveroad Inc. - Web and App development company*. URL: https://www.cleveroad.com/blog/native-vs-hybrid-app-development/ (дата звернення: 23.06.2023).

9. Learn web development | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Learn (дата звернення: 10.12.2023).

10. The biggest advantages and disadvantages of hybrid apps | Zudu. *Zudu*. URL: https://zudu.co.uk/blog/hybrid-apps-pros-and-cons/ (дата звернення: 25.06.2023).

11. Advantages and Disadvantages of Automated Testing - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-automated-testing/ (дата звернення: 12.09.2023).

12. 31 Top Automation Testing Tools In 2023 [Updated] | LambdaTest. *LambdaTest*. URL: https://www.lambdatest.com/blog/automation-testing-tools/ (дата звернення: 14.09.2023).

13. Your step-by-step mobile application testing process - Testlio. *Testlio*. URL: https://testlio.com/blog/step-step-mobile-application-testing-process/ (дата звернення: 20.09.2023).

14. How to approach Cross Platform Testing | BrowserStack. *BrowserStack*. URL: https://www.browserstack.com/guide/cross-platform-testing (дата звернення: 28.09.2023).

15. Different Operating Systems - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/different-operating-systems/ (дата звернення: 12.10.2023).

16. Интернет-приложения, собственные приложение и гибридные приложения – Сравнение типов приложений – AWS. *Amazon Web Services, Inc.* URL: https://aws.amazon.com/ru/compare/the-difference-between-web-apps-native-apps-and-hybrid-apps/ (дата звернення: 12.10.2023).

17. 5 Types of Programming Languages. *Coursera*. URL: https://www.coursera.org/articles/types-programming-language (дата звернення: 12.10.2023).

18. Visual Identification of Mobile App GUI Elements for Automated Robotic Testing. *Publishing Open Access research journals & papers | Hindawi*.

URL: https://www.hindawi.com/journals/cin/2022/4471455/ (дата звернення: 12.10.2023).

19. How to Build an Application to Test Motion Sensors in Android? - GeeksforGeeks. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/how-to-build-an-application-to-test-motion-sensors-in-android/ (дата звернення: 12.10.2023).

20. Daisy Raines. Sensor Check Android: The Best Apps and Methods in 2023. *[Official]Dr.Fone: Your One-Stop Complete Mobile Solution*. URL: https://drfone.wondershare.com/device-diagnostics/test-android-sensors.html (дата звернення: 13.10.2023).

21. 25 BEST Software Development & Programming Tools (2023). *Guru99*. URL: https://www.guru99.com/software-development-tools.html (дата звернення: 13.10.2023).

22. Appium Documentation - Appium Documentation. *Redirecting*. URL: https://appium.io/docs/en/2.2/ (дата звернення: 14.10.2023).

23. Katalon Studio | Best Codeless Test Automation Tools. *katalon.com*. URL: https://katalon.com/katalon-studio (дата звернення: 14.10.2023).

24. Espresso | Android Developers. *Android Developers*. URL: https://developer.android.com/training/testing/espresso (дата звернення: 14.10.2023).

25. Contributors to Wikimedia projects. Instagram - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Instagram (дата звернення: 21.10.2023).

26. Instagram Features. *About Instagram | Capture, Create & Share What You Love*. URL: https://about.instagram.com/features (дата звернення: 21.10.2023).

27. How to create Test Scenarios? (with Examples) | BrowserStack. *BrowserStack*. URL: https://www.browserstack.com/guide/how-to-create-test-scenarios (дата звернення: 23.10.2023).

28. How to Install Appium: Setting Up And Configuring For Mobile App Testing | LambdaTest. *LambdaTest*. URL: https://www.lambdatest.com/blog/install-appium/ (дата звернення: 23.10.2023).

29. Katalon Studio installation overview | Katalon Docs. URL: https://docs.katalon.com/docs/katalon-studio/get-started/katalon-studio-installation/katalon-studio-installation-overview (дата звернення: 27.10.2023).

30. Testing Business Applications Guide. *Oracle Help Center*. URL: https://docs.oracle.com/cd/F14158_13/books/TestGuide/setting-up-android-mobile-devices-for-automation-testing.html#setting-up-android-mobile-devices-for-automation-testing (дата звернення: 30.11.2023).

Декларація

академічної доброчесності

здобувача вищої освіти ЗНУ

Я__Пасенко Олексій Євгенійович_, студент___2__курсу,
форми здобуття освіти __денна__,
Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ
Спеціальності121 Інженерія програмного забезпечення,
адреса електронної пошти__ipz18bd-6@stu.zsea.edu.ua___,
підтверджую, що написана мною кваліфікаційна робота на тему
«_Порівняльне дослідження технологій автоматизованого тестування мобільних застосунків_»
відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений/ознайомлена;
- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;
- згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою Інтернет-системи, а також на архівування роботи в базі даних цієї системи.

Дата_____ Підпис_____ ПІБ (студент) __Пасенко О.Є._____
Дата_____ Підпис_____ ПІБ (науковий керівник) _Коломоєць Г. П._