# СТВОРЕННЯ СТРАТЕГІЙ ТЕСТУВАННЯ ДЛЯ ОПТИМАЛЬНОГО ПОЄДНАННЯ РУЧНОГО ТА АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ В СТАРТАП-ПРОЕКТАХ

**Мокрицька Ольга[1], Трухан Дмитро[2]**

[1] Кафедра систем автоматизованого проектування, Національний університет «Львівська політехніка», Україна, Львів, вул. С. Бандери 12, https://orcid.org/0000-0002-2887-9585 olha.v.mokrytska@lpnu.ua.

[2] Кафедра комп'ютерних наук, Національний лісотехнічний університет , Україна, Львів, вул. Генерала Чупринки, 103, https://orcid.org/0009-0000-8413-5532 23trukhan.d@nltu.lviv.ua

**Abstract** – The paper considers the creation of a testing strategy for a startup project that combines manual and automated testing. Automation is implemented using the AirTest IDE and the Python programming language. Particular attention is paid to the choice of tools to ensure high test coverage of the main project functions. Manual testing is used for more complex scenarios, while automation covers routine tasks. The strategy is aimed at improving the quality of the product with minimal resource consumption.

**Анотація** – У роботі розглянуто створення стратегії тестування для стартап-проекту, що поєднує ручне та автоматизоване тестування. Автоматизація реалізована за допомогою AirTest IDE та мови програмування Python. Особливу увагу приділено вибору інструментів для забезпечення високого покриття тестами основних функцій проекту. Ручне тестування використовується для складніших сценаріїв, тоді як автоматизація покриває рутинні завдання. Стратегія спрямована на підвищення якості продукту при мінімальних витратах ресурсів.

**Ключові слова** – тестування, автоматизоване тестування, ручне тестування, AirTest IDE, Python.

Ручне та автоматизоване тестування є двома основними підходами, що використовуються для забезпечення якості програмного забезпечення. У стартап-проектах, де часто зустрічаються обмежені ресурси та стислі терміни, важливо знайти баланс між цими підходами для досягнення оптимального результату. У цій роботі було розроблено стратегію, що поєднує автоматизовані та ручні підходи до тестування, спрямовану на ефективність та економію ресурсів.

Для автоматизації тестування було обрано інструмент **AirTest IDE** через його можливості легкої інтеграції з мобільними додатками, а також зручний інтерфейс для написання та налагодження тестових сценаріїв. Крім того, AirTest дозволяє створювати кросплатформні тести, що є важливим для стартапів, які мають обмежений бюджет, але потребують перевірки продукту на різних пристроях.

**Основними перевагами AirTest IDE є:**

- Візуальний редактор: дозволяє створювати тести без глибоких знань програмування, використовуючи запис та відтворення дій.
- Підтримка Python: для більш гнучких та складних тестів можна використовувати Python, який надає можливість легко модифікувати тестові сценарії.[3]
- Можливість інтеграції з іншими фреймворками: AirTest добре працює в парі з іншими інструментами, що дозволяє масштабувати систему автоматизації тестування залежно від потреб проекту.

Ручне тестування залишається невід'ємною частиною процесу, оскільки деякі функціональні або користувацькі сценарії можуть бути складними для автоматизації. Наприклад, тестування користувацького досвіду (UX), візуального відгуку інтерфейсу або нестандартних випадків, які не можуть бути легко автоматизовані. Ручне тестування також використовується для підтвердження нових функцій або складних інтеграцій перед їх автоматизацією.[1][2]

Для автоматизації було розроблено низку тестових сценаріїв, що перевіряють ключові функції продукту, зокрема:

- **Тестування основного функціоналу:** автоматизовані тести покривають повторювані завдання, такі як реєстрація користувача, вхід до системи, додавання та редагування записів.
- **Регресійне тестування:** після кожного оновлення програмного забезпечення автоматизовані тести перевіряють, чи працюють основні функції коректно після внесення змін.
- **Тестування на різних платформах:** AirTest IDE дозволяє запускати тести на різних пристроях і платформах, що є особливо важливим для стартапів, де необхідно перевіряти мобільні та десктопні версії продукту.[4]

**Приклад тестового сценарію на Python(частина):**

```
# -*- encoding=utf8 -*-
__author__ = "dimat"
from airtest.core.api import *
auto_setup(__file__)
from poco.drivers.ios import iosPoco
poco = iosPoco()
def Loading_Splash_Screen():
    sleep(1.0)
    assert_exists(Template(r"tpl1712052662382.png", record_pos=(-0.001, -0.001), resolution=(1334, 750)), "AA Logo is present!")
    sleep(1.0)
    assert_exists(Template(r"tpl1712051648871.png", record_pos=(-0.001, 0.05), resolution=(1334, 750)), "FP Logo is present!")
    sleep(1.0)
```

assert_exists(Template(r"tpl1712051912593.png", record_pos=(-0.001, -0.001), resolution=(1334, 750)), "Splash Screen is present!")

touch(Template(r"tpl1712051977606.png", record_pos=(0.001, 0.04), resolution=(1334, 750)))

Даний код демонструє ініціалізацію AirTest та Python скрипта.

Автоматизація виконується за умови прописання функцій, в тіло яких додається дія, координати та об’єкт з яким має працювати дана функція. На рисунку 1, можна побачити згенерований AirTest звіт, в якому пристуній час виконня скрипта, результат виконання та детальні кроки з додатковою інформації по кожному зробленому кроку.

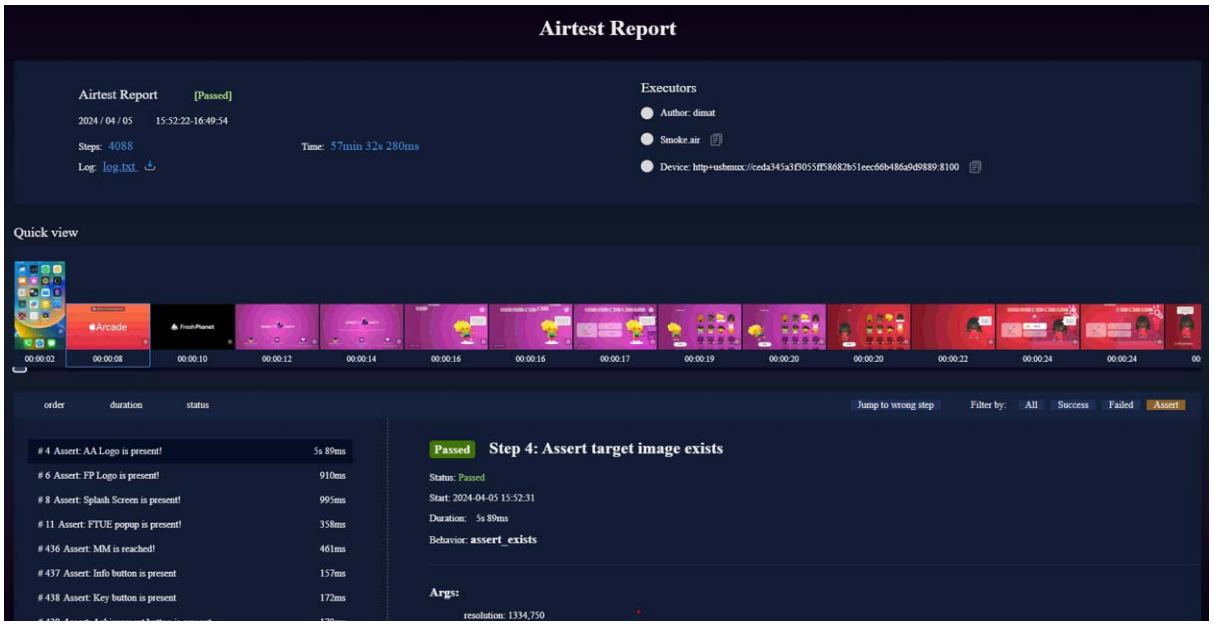


Рис.1 Результат виконання скрипту в AirTest IDE

**Висновок**. Використання AirTest IDE дозволило значно знизити час, необхідний для проведення рутинних тестів, таких як перевірка вхідних форм або основних функцій. Завдяки можливості повторного використання тестових сценаріїв, процес тестування став ефективнішим. Автоматизація регресійного тестування дозволила команді зосередитися на розробці нових функцій та їх тестуванні, не витрачаючи багато часу на перевірку старих частин продукту. У результаті застосування

автоматизованого підходу до тестування вдалося скоротити час виконання тестів на 40%, а також підвищити точність виявлення помилок на 25%. Це особливо важливо для стартапів, де кожна хвилина та кожен ресурс мають значення.

**Перелік посилань**

[1] Олейник, М. П., Іванов, Д. І. (2020). Підходи до автоматизації процесів тестування у сучасних системах. Наукові записки НаУКМА. Серія: Комп'ютерні науки, 16(2), 35-40.

[2] Карпов, Ю. Г. (2016). Основи автоматизованого тестування: принципи та підходи. Харків: Вид-во ХНУРЕ.

[3] Pycharm Documentation. (2023). Python Testing Tutorial. Retrieved from https://www.jetbrains.com/help/pycharm/pytest.html

[4] Airtest Documentation. (2023). Airtest IDE: Cross-platform testing tool. Retrieved from https://airtest.doc.io/en/IDE/