

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ДЛЯ АВТОМАТИЗАЦІЇ ФУНКЦІОНАЛЬНОГО ТЕСТУВАННЯ
МОБІЛЬНОГО ЗАСТОСУНКУ
RESEARCH AND IMPLEMENTATION OF SOFTWARE FOR AUTOMATING
THE FUNCTIONAL TESTING OF A MOBILE APPLICATION

Виконав: студент(ка) 2 курсу, групи КНТ-219м
Спеціальності 122 Комп'ютерні науки
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Білов О.І.

(прізвище та ініціали)

Керівник Пархоменко А.В.

(прізвище та ініціали)

Рецензент Морщавка С.В.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 122 Комп'ютерні науки
(код і найменування)
 Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 20 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Білова Олега Ігоровича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та реалізація програмного забезпечення для автоматизації функціонального тестування мобільного застосунку. Research and Implementation of Software for Automating the Functional Testing of a Mobile Application

керівник проєкту (роботи) Пархоменко Анжеліка Володимирівна, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” листопада 2020 року № 301

2. Строк подання студентом проєкту (роботи) 8 грудня 2020 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Аналіз та вибір інструментальних засобів для автоматизованого тестування мобільних застосунків. 3. Розробка методики та програмного забезпечення для автоматизованого тестування мобільного застосунку. 4. Інструкції до використання та експериментальне дослідження автоматизованих тестів. 5. Економіко-організаційна частина. 6. Охорона праці та безпека у надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Пархоменко А.В., доцент		
5 Організаційно-економічна частина	Пожуєва Т.О., професор		
6 Охорона праці та цивільна безпека	Коробко О.В., ст. викладач		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 4 ” вересня 2020 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Аналіз та вибір інструментарію розробки.	4–5 тижні	Розділ 2
4	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	6 тиждень	Розділ 3
5	Розробка архітектури та структури програми.	7–8 тижні	Розділ 3
6	Розробка та експериментальне дослідження програми.	9 тиждень	Розділ 4
7	Розробка організаційно-економічної частини	10 тиждень	Розділ 5
8	Розробка заходів з охорони праці та цивільної безпеки.	11 тиждень	Розділ 6
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14–15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

(підпис) Білов О.І.
(прізвище та ініціали)

Керівник проекту (роботи)

(підпис) Пархоменко А.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
112 с., 20 рис., 12 табл., 2 дод., 52 джерела.

МОБІЛЬНИЙ ЗАСТОСУНОК, АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ,
ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ, ANDROID, PYTHON.

Об'єкт дослідження – методи тестування програмного забезпечення.

Предмет дослідження – методи та інструментальні засоби автоматизації тестування мобільних застосунків.

Мета роботи – розробка методики автоматизованого тестування мобільних застосунків та розробка програмного забезпечення для автоматизації процесу функціонального тестування мобільного застосунку.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, метод тестування «чорної скриньки» та функціональне тестування, середовище розробки PyCharm, мова програмування Python, персональний комп'ютер з інструментарієм Appium, Android SDK та Selenium.

Наукова новизна роботи полягає в тому, що розроблено методику автоматизованого тестування мобільних застосунків, в якій отримав подальший розвиток підхід до функціонального тестування програмного забезпечення за методом «чорної скриньки».

Практична значимість роботи полягає у тому, що було розроблено автоматизовані функціональні тести для проведення тестування мобільного застосунку, які дозволили скоротити час тестування на 70-85%.

Висновки. Розроблено методику автоматизованого тестування, сценарії функціонального тестування та на основі цих сценаріїв програмне забезпечення для проведення тестування мобільного застосунку, що дозволяє автоматизувати процес тестування.

Галузь використання – розробка мобільних застосунків.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 112 pages, 20 figures, 12 tables, 2 appendixes, 52 sources.

MOBILE APPLICATION, TEST AUTOMATION, TEST CASES, FUNCTIONAL TESTING, ANDROID, PYTHON.

The object of study is a methods of software testing.

The subject of study is a methods and algorithms for automation of mobile applications testing.

The goal of the master thesis is to create a method of automated testing of mobile applications and to develop software to automate the process of functional testing of mobile applications.

Materials, methods and tools: object-oriented programming, black box testing method and functional testing, PyCharm development environment, Python programming language, personal computer with Appium tools, Android SDK and Selenium.

Scientific novelty lies in the methods of automated testing of mobile applications is developed, which further developed the functional testing of software by the method of "black box".

The practical significance of the work is that automated functional tests have been developed for testing mobile applications, which have reduced testing time by 70-85%.

Conclusions. The technique of automated testing, functional testing scenarios and on the basis of these scenarios software for testing a mobile application, which allows to automate the testing process, have been developed.

The field of application is a development of mobile applications.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	9
Вступ	10
1 Аналіз предметної області	12
1.1 Класифікація методів тестування	12
1.1.1 Аналіз методів автоматизованого тестування	16
1.2 Особливості тестування мобільних застосунків	20
1.3 Автоматизація тестування мобільних застосунків	22
1.3.1 Категорії автоматизації функціонального тестування	26
1.3.2 Емулятори для автоматизації тестування мобільних застосунків ..	27
1.4 Висновки до розділу 1	28
2 Аналіз та вибір інструментальних засобів для автоматизованого тестування мобільних застосунків	30
2.1 Інструментальні засоби для взаємодії з мобільним застосунком	31
2.1.1 Вибір драйверу	32
2.1.2 Вибір надбудови	33
2.2 Вибір інструментальних засобів для написання та виконання тестів ...	36
2.2.1 Вибір мови програмування	36
2.2.2 Вибір середовища для написання та виконання тестів	39
2.3 Висновки до розділу 2	42
3 Розробка методики та програмного забезпечення для автоматизованого тестування мобільного застосунку	43
3.1 Розробка методики автоматизованого тестування мобільних застосунків	43
3.2 Розробка сценаріїв тестування на основі діаграми переходів між станами тестованого мобільного застосунку	45
3.3 Розробка архітектури взаємодії програмного забезпечення автоматизованих тестів	46

3.4 Структура програмного забезпечення автоматизованих тестів для мобільного застосунку	47
3.5 Використані методи та функції при створенні автоматизованих тестів для мобільного застосунку	50
3.6 Висновки до розділу 3	52
4 Інструкції до використання та експериментальне дослідження автоматизованих тестів	53
4.1 Призначення та умови використання програмного продукту	53
4.2 Алгоритм роботи з програмним забезпеченням для автоматизації функціональних тестів	53
4.3 Вхідні та вихідні дані	54
4.4 Налаштування емулятора та серверу для запуску автоматизованих тестів	55
4.5 Порівняння результатів виконання автоматизованих і ручних тестів ..	58
4.6 Висновки до розділу 4	59
5 Економіко-організаційна частина	60
5.1 Планування розробки програмного продукту	60
5.2 Визначення термінів розробки	60
5.3 Визначення витрат на розробку програми	62
5.4 Розрахунок основної заробітної плати	63
5.5 Розрахунок додаткової заробітної плати.....	63
5.6 Відрахування на єдиний соціальний внесок	64
5.7 Визначення витрат на матеріали	64
5.8 Витрати на спеціальне обладнання	65
5.9 Розрахунок накладних витрат.....	69
5.10 Розрахунок економічної ефективності	70
6 Охорона праці та безпека у надзвичайних ситуаціях.....	72
6.1 Аналіз потенційних небезпек	72
6.2 Заходи із забезпечення безпеки	73
6.3 Заходи з виробничої санітарії та гігієни праці.....	81

6.4 Заходи безпеки у надзвичайних ситуаціях.....	85
6.4.1 Заходи з пожежної безпеки	85
6.4.2 Заходи з цивільного захисту	86
Висновки.....	89
Перелік джерел посилання	91
Додаток А Текст програми	97
Додаток Б Слайди презентації	104

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
GUI	– графічний інтерфейс користувача;
IDE	– Integrated Development Environment,
SDK	– Software Development Kit;
ЕОМ	– електронно-обчислювальна машина.
ОС	– операційна система;
ПЗ	– програмне забезпечення;
ПК	– персональний комп'ютер;

ВСТУП

Ринок мобільних застосунків зростає з кожним роком [1] - [2]. На теперішній час існує безліч застосунків для мобільних пристроїв у різних сферах діяльності: фінанси, розваги, покупки, соціальні мережі та інші. І все більше компаній роблять ключовою складовою свого бізнесу саме мобільно орієнтовані застосунки. Дослідження проведені за 2019 рік свідчать, що в середньому людина проводить близько 4 годин у мобільних застосунках [2]. Користувачі чекають надійності, простоти використання, продуктивності та безпеки, але нажаль якість випускаємих мобільних застосунків нижча за очікувану. В дослідженні показано, що відсутність якості пов'язано з дуже швидким процесом розробки, в якому тестування програмного забезпечення ігнорується або виконується поверхнево [1], [3]. Крім того, саме час витрачений на тестування відіграє ключову роль у випуску продукту на ринку, та у реалізації нових версій продукту.

Тестування програмного забезпечення мобільних застосунків – це процес дослідження, випробування програмного продукту, який має на меті перевірку відповідності між реальною поведінкою програми і її очікуваною поведінкою на кінцевому наборі тестів [4].

Щоб задовольнити зростаючий попит на мобільні застосунки, команда розроблюваного застосунку повинна приділяти більше зусиль і уваги процесам розробки програмного забезпечення, зокрема, тестування програмного забезпечення.

Сьогодні, підходи та засоби тестування активно розвиваються, що підтверджується в роботах [5] - [7]. Оскільки тестування один з найважливіших етапів контролю якості в процесі розробки програмного забезпечення. Автоматизоване тестування є його складовою частиною. Воно використовує програмні засоби для виконання тестів і перевірки результатів виконання, що допомагає скоротити час тестування і спростити його процес. Тому, саме автоматизація тестування, розуміння того, як автоматизоване

тестування мобільних застосунків здійснюється на практиці, грає стратегічну роль у цьому процесі розробки програмного продукту. Таким чином розробка методики автоматизованого тестування та впровадження їх у процес розробки програмного продукту для мобільних застосунків є актуальним завданням.

Метою дипломної кваліфікаційної роботи магістра є розробка методики автоматизованого тестування мобільних застосунків та розробка програмного забезпечення для автоматизації процесу функціонального тестування мобільного застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Класифікація методів тестування

Дослідження ресурсів показало, що існуючі методи тестування можна розподілити за групами, узагальнена класифікація зображені на рисунку 1.1 [4] - [7].

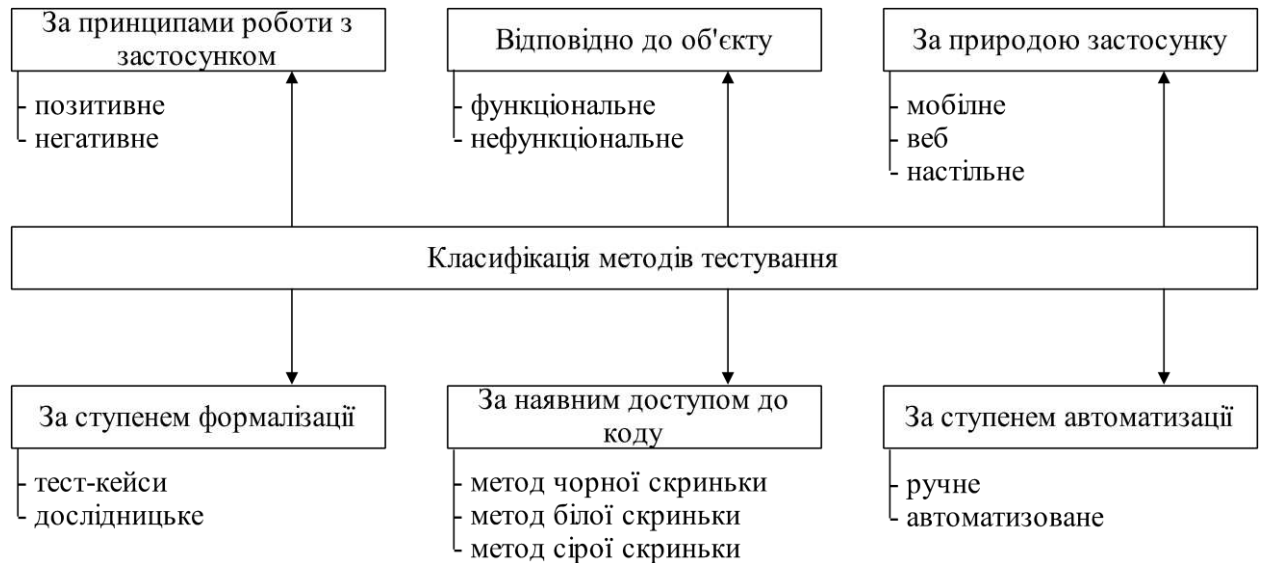


Рисунок 1.1 – Класифікація методів тестування

Позитивне тестування необхідно для контролю роботи програми в ситуації, коли дії виконуються строго по інструкції, без яких би то не було помилок, завідомо неправдивих ситуацій, введення невірних даних і т.і. Для того щоб прискорити тестування кілька позитивних тест-кейсів можна об'єднати (наприклад, перед відправкою заповнити всі поля форми коректними значеннями). Іноді це може ускладнити діагностику помилки, але суттєва економія часу компенсує цей ризик [8].

Негативне тестування передбачає відпрацювання тих сценаріїв, які можуть статися з системою, якщо, користувач припуститься помилки при введенні даних. В даному випадку, система повинна відповісти на запит користувача повідомленням про помилку. Хоча негативних тест-кейсів набагато більше, ніж позитивних, негативні не варто об'єднувати, тому що

подібне рішення може привести до невірної трактування поведінки програми та пропуску дефектів [8].

За природою застосунки можна класифікувати на: настільні, мобільні, вебзастосунки [9].

Тестування мобільних застосунків вимагає підвищеної уваги до тестування сумісності, оптимізації продуктивності, автоматизації тестування з використанням емуляторів мобільних пристроїв.

Тестування на основі тест-кейсів – це формалізований підхід, тестування проводиться за допомогою заздалегідь підготовлених тест-кейсів або тестових планів, а також іншої документації. Найпоширеніший метод тестування, за допомогою якого досягається найбільша повнота тестування програми за рахунок серйозної систематизації процесу, впровадження метрик і широкого комплексу вироблених і перевірених на практиці порад [7].

Дослідницьке тестування – частково формалізований підхід, в рамках якого тестувальник виконує роботу з застосунком згідно заданого сценарію, який, в свою чергу, допрацьовується в процесі виконання з метою найбільш досконалого вивчення програми. Основним фактором успіху при виконанні дослідницького тестування є саме робота за певним сценарієм, а не виконання розрізнених бездумних операцій.

Функціональне тестування - це тестування функцій та можливостей системи, а також її взаємодія за специфічними системами. Розділяють 3 рівня тестування:

- компонентне: цілеспрямоване тестування є: виявити функціональні помилки в роботі окремо взятого ізольованого модуля;
- інтеграційне: на даному етапі оцінюється здатність модулів взаємодіяти з іншим згідно вимог;
- системне: системне тестування оцінює роботу всієї системи в цілому [10].

Нефункціональне тестування набагато складніше функціонального, оскільки в даному випадку інженер за якістю оцінює не те «що» робить

система, а «як» вона робить. Нефункціональне тестування оцінює наступні характеристики:

- зручність та простоту використання;
- дружність інтерфейсу;
- наскільки інтуїтивно зрозумілий функціонал.

До нефункціонального можна віднести наступні види тестування:

- навантажувальне тестування: це тестування, при якому тестується введення систем при максимально допустимому значенні навантаження;
- тестування зручності використання: це тестування систем на відповідність вимогам до зручності;
- тестування надійності: це тестування систем на відповідність вимогам до надійності;
- тестування зручності використання систем;
- тестування інтерфейсу користувача: охоплює виключно інтерфейс користувача, але не оцінює його з усіх точок зору;
- тестування безпеки: даний вид тестування оцінює, наскільки система безпечна для користувача та іншої системи;
- тестування локалізації: тестування адаптації програм до різних мов;
- тестування сумісності: це тестування систем введення в непривичному оточенні інших систем;
- стрес-тестування: це тестування систем у нестандартних для неї ситуаціях (наприклад, введення систем при обриві зв'язку);
- тестування стабільності: це тестування систем на предмет того, як довго вона може працювати без збоїв[10].

Регресійне тестування - це повторна перевірка функціональності, який був раніше вже протестований, до введення в систему змін, які можуть впливати на стан функціоналу, що існує на той момент [11].

Як показало дослідження, існують інструменти тестування мобільних

застосунків. Умовно їх можна поділити на інструменти для ручного та автоматизованого тестування.

Сьогодні багато фахівців підтримують думку про те, що неможливо відмовитись від ручного тестування [1], [3]. Але, обійтись без автоматизації тестування на сьогоднішній день дуже важко.

Переваги ручного тестування мобільних застосунків:

- це більш економічно вигідно в короткостроковій перспективі;
- ручне тестування більш гнучке;
- найкраще моделювання дій користувача.

Недоліки ручного тестування мобільних застосунків:

- ручні тестові приклади важко використовувати повторно;
- менш ефективне виконання певного постійного завдання;
- процес тестування повільний;
- деякі типи тестових випадків не можуть бути виконані вручну (тестування навантаження).

Переваги автоматизованого тестування застосунків:

- процес тестування займає мало часу;
- економічність в довгостроковій перспективі використання;
- автоматизовані тестові випадки легко використовувати повторно;
- єдине рішення для деяких видів тестування (тестування продуктивності);
- результати випробувань легкодоступні.

Недоліки автоматизованого тестування застосунків:

- у деяких мобільних засобів тестування є обмеження;
- процес написання тестів займає багато часу;
- автоматизоване тестування найменш ефективне у визначенні зручності користування[10].

Таким чином, хоча автоматизоване тестування і має недоліки, пов'язанні з часовими, ресурсними витратами на їх створення та виконання,

але з точки зору швидкості виконання тестів, продуктивності, точності та можливості повторного використання розроблених тестів, автоматизоване тестування надає суттєву перевагу порівняно з ручним тестуванням.

1.1.1 Аналіз методів автоматизованого тестування

Існує два основних підходи до автоматизації тестування (рис.1.2):

- тестування на рівні коду (біла скринька);
- тестування графічного користувацького інтерфейсу (GUI-тестування), чорна скринька [1].

До першого типу належить, зокрема, модульне тестування. До другого – імітація дій користувача за допомогою спеціальних тестових фреймворків. Найбільш поширеною формою автоматизації є тестування застосунків через графічний користувацький інтерфейс. В роботі розглянуто один з таких підходів.

Популярність такого виду тестування пояснюється двома факторами: по-перше, застосунок тестується тим же способом, яким його буде використовувати людина; по-друге, можна тестувати застосунок, не маючи при цьому доступу до вихідного коду.

Однією з головних проблем автоматизованого тестування є його трудомісткість, незважаючи на те, що воно дозволяє усунути частину рутинних операцій і прискорити виконання тестів, великі ресурси можуть витрачатися на оновлення самих тестів. Це відноситься до обох видів автоматизації. При рефакторінгу програм часто буває необхідно оновити і модульні тести, і зміна коду тестів може зайняти стільки ж часу, скільки і зміна основного коду. З іншого боку, при зміні інтерфейсу застосунку необхідно заново переписати всі тести, які пов'язані з оновленими вікнами, що при великій кількості тестів може відняти значні ресурси.

Другою проблемою є час виконання автоматизованих тестів. Тести GUI на великих проєктах зазвичай запускаються звечора, аби до ранку

результат їх проходження був готовим. Це сповільнює вихід нових версій та не дає швидко виправити помилки, якщо вони раптом з'являються в продукті.

До більш дієвих аспектів тестування, що забезпечує вищий щабель автоматизації, відносяться багатфункціональні критерії [12].

Тестування за багатфункціональними критеріями, передбачає здійснення процесу тестування за принципом «чорної скриньки» (рис. 1.2). При цьому невідома структура програми, вихідні коди до проєкту недоступні, однак відома специфікація програмного продукту.



Рисунок 1.2 – Методи автоматизованого тестування

До більш відомих критеріїв функціонального тестування відносяться критерії тестування функцій і тестування специфікацій.

Тестування функцій - один з найбільш застосовуваних на практиці критеріїв. Відповідно до критерію тестування функцій необхідно перевірити кожну функцію (спосіб), що реалізовується програмним модулем (класом).

При тестуванні специфікації необхідно вибудувати комплект тестів, що забезпечує перевірку кожного пункту вимог хоча б один раз. Специфікація вимог може містити сотні і тисячі вимог, при цьому тестування специфікації повинно перевірити їх всі. Повинні бути перевірені не тільки всі вимоги до кожного конкретного методу, але також всі вимоги до класів і системи в цілому.

До приватних функціональних критеріїв належать: тестування класів еквівалентності, тестування граничних значень, тестування на основі діаграм причинно-наслідкових зв'язків, тестування функцій і тестування пунктів специфікацій. Так як вичерпне тестування нетривіальною програми неможливо, то необхідно вибрати оптимальну підмножину даних з області визначення функції, що тестується [11].

Тестування класів еквівалентності – одна з методик визначення «оптимального» тестового комплекту. Вона полягає в розбитті вхідної області визначення і вихідний області значень на кінцеве число класів еквівалентності. Розрізняють два типи класів еквівалентності: правильні класи еквівалентності, що включають коректні (правильні) дані і неправильні класи еквівалентності, що охоплюють помилкові дані [10].

Граничні умови – це ситуації, що виникають як безпосередньо на кордонах класів еквівалентності, так і на значеннях вище або нижче даних кордонів. Тестування програм, здійснюється як на граничних значеннях, так і поблизу заданих меж. Як показує практика, тестування класів еквівалентності і тестування граничних значень використовується як єдиний комбінаторний критерій. Основним недоліком критеріїв еквівалентних класів і граничних значень є те, що вони не досліджують всіх можливих комбінації безлічі вхідних значень [10].

Метод діаграм переходів (метод функціональних діаграм) дозволяє формувати репрезентативні комплекти тестових даних і виконувати систематичну генерацію високо результативних тестів.

Інтенсивність виявлення помилок на одиницю витрат і надійність тісно пов'язані з часом тестування і, тому, є гарантією якості продукту (рис. 1.3). Чим більше ми витрачаємо часу в процесі тестування, тим менше залишається в продукті критичних помилок. Однак, як відомо повне тестування не можна досягти, тому трудовитрати, пов'язані з отриманням програмного продукту, а також з надлишком якості, яке не затребуване замовником програми, повинні мати межі. Визначення моменту завершення тестування - дуже відповідальне завдання тестувальника і менеджера проєкту [13] - [14].

Рух до зменшення числа помилок що залишились, або, до якості продукту, призводить до використання різних способів тестування в процесі створення ПЗ. На рисунку 1.3, блок В наведено витрати на тестування в залежності від вдосконалення застосовуваного інструментарію і методів тестування [13].

Залежність ефективності впровадження перерахованих способів або їх можливості до виявлення відповідних класів помилок порівняна на рисунку 1.3 з витратами. Графік показує, що з часом, по мірі виявлення найбільш серйозних помилок і недоліків, ефективність низьковитратних методів падає разом з кількістю виявлених помилок.

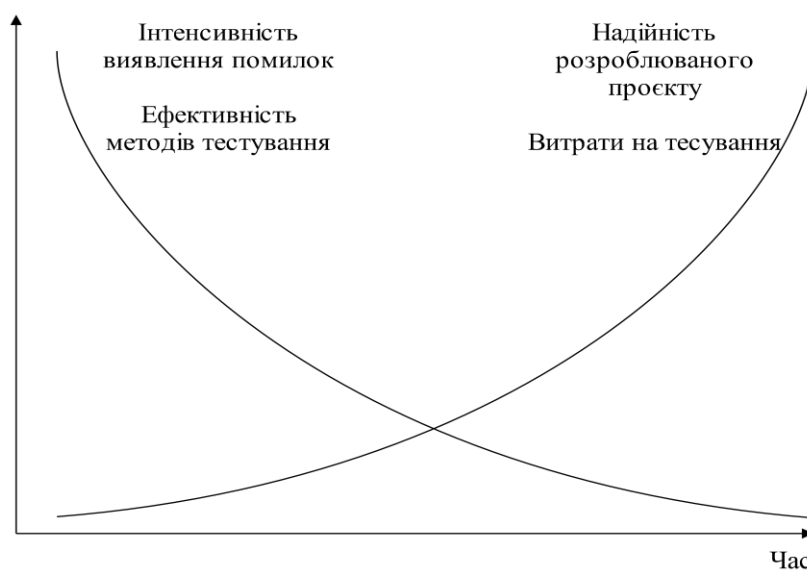


Рисунок 1.3 – Методи тестування на шкалі часу

З цього дослідження можна зробити висновок, що всі методи тестування мають право на існування, але вони мають свою область застосунку, де вони добре виявляють помилки, тоді як поза цієї області їх ефективність падає.

1.2 Особливості тестування мобільних застосунків

Тестування застосунків на мобільних пристроях в цілому відповідає загальним принципам тестування, але також присутній ряд особливостей, які характерні саме для тестування мобільних застосунків [15].

Для того щоб зрозуміти особливості тестування мобільних застосунків, необхідно враховувати чинники, які відрізняють мобільний застосунок від десктопного, а саме: специфічні і різноманітні операційні системи для мобільних платформ, різні конфігурації комплектуючих, функціональність таких пристроїв як комунікатори і т.ін.

У зв'язку з цими факторами підхід до тестування застосунків на мобільних пристроях відрізняється від десктопного. З'являється велика кількість додаткових нюансів і вимог, які необхідно протестувати.

Основні моменти, на які необхідно звертати особливу увагу саме при тестуванні застосунків на мобільних пристроях.

Розмір екрану і сенсорний інтерфейс:

- розміри всіх елементів графічного інтерфейсу користувача;
- перевірка можливості використання всіх активних елементів (кнопок, посилань і т.д.);
- швидкість відгуку активних елементів повинна бути досить високою;
- перевірка того, що багаторазове швидке натискання на кнопку не призведе до екстреного завершення програми;
- підтримка одночасного натискання кількох кнопок;
- підтримка горизонтального і вертикального положення;

- перевірка використання в застосунку спеціальних жестів (подвійний натиск, прокрутка і т.ін.).

Ресурси телефону:

- перевірка обробки ситуацій нестачі пам'яті для функціонування операційної системи, під час роботи програми в активному і фоновому режимах;

- недолік місця для установки або роботи програми;
- установка на карту пам'яті;
- перевірка роботи батареї (акумулятора) пристрої при запусненому застосунку, роботи у фоновому режимі, при включеному Інтернет, без підключення до мережі і т.д.

Різні дозволи екрану і версії ОС:

- необхідно перевірити роботу програми на пристроях з різними дозволами екрану, на екранах з високою роздільною здатністю (наприклад, ретина-екран) елементи інтерфейсу і текст відображаються дрібніше, при роботі застосунка на пристрої з екраном більш низького дозволу елементи інтерфейсу можуть стати занадто великими;

- необхідно переконатися, що програма не може бути встановлено на непідтримувані пристрої, при цьому обов'язково тестування програми на всіх заявлених підтримуваних пристроях.

Реакція програми на зовнішні переривання:

- вхідні та вихідні SMS і MMS;
- вхідні та вихідні дзвінки;
- нагадування, будильник і т.д;
- відключення і підключення Wi-Fi;
- часто виникають проблеми з переходом в онлайн-режим після офлайн режиму;

- відключення і підключення карти пам'яті;
- відключення / підключення мобільного пристрою до зарядного

пристрою;

- робота з фізичною клавіатурою (якщо в списку підтримуваних моделей є такі);
- робота в фоновому режимі;
- робота застосунка після виходу із сплячого режиму;
- сумісність з іншими застосунками.

Користувач мобільного пристрою очікує, що програми прості, інтуїтивно зрозумілі, працюють завжди і всюди без збоїв. Якщо очікування не виправдовуються, то користувач встановлює аналогічний застосунок від іншого розробника, яких в сфері мобільних розробок завжди достатньо. Тому якість програми є одним з головних факторів його популярності [16].

1.3 Автоматизація тестування мобільних застосунків

Автоматизоване тестування - це тестування програмного забезпечення з використанням засобів автоматизації: скрипти, віртуальні машини, емулятори, спеціалізовані програми та ін. [9]. Автоматизація тестування насамперед потрібна для прискорення та повторного застосування тестів.

Важливим моментом при створенні тестових систем та написанні тестів є наявність плану тестування, що визначений міжнародним стандартом IEEE 829- 1983 [17].

В ньому повинні бути наступні розділи:

- що буде тестуватися (тестові вимоги, варіанти тестів);
- якими методами буде тестуватися система та критерії, за якими буде визначатися успіх тестів;
- план робіт та ресурси (тестувальники, техніка).

Також слід визначити певні критерії вдалого/невдалого завершення тестів, ризики і плани керування та конфігурації середовища та керування та т.п. Коли зрозуміло, що і навіщо потрібно тестувати, і є план дій, саме час задуматися про те, як це зробити ефективніше, швидше і якісніше. Сучасне

програмне забезпечення – це складний об’єкт, і вручну з ним справлятися важко і дорого. До того ж при «ручному» тестуванні результати кожного виконання тестів пропадають, і їх важко повторити. Для того щоб збільшити обсяг перевірок і підвищити якість тестування, забезпечити можливість повторного використання тестів при внесенні змін у програмне забезпечення застосовують засоби автоматизації тестування.

Сьогодні Україна відстає від решти комп’ютерного світу щодо застосування засобів автоматизації тестування, оскільки існує неправильне ставлення до тестування багато керівників вважають, що розробник може написати програму без помилок. Інший фактор це бажання заощадити, робота спеціаліста тестувальника який володіє засобами автоматизації тестування обходиться дорожче. Крім того, обмеження по термінах – автоматизація тестування вимагає значних тимчасових витрат і має сенс тільки в тому випадку, якщо проєкт як мінімум середньостроковий.

Життєвий цикл тесту зображено на рисунку 1.4 [13].

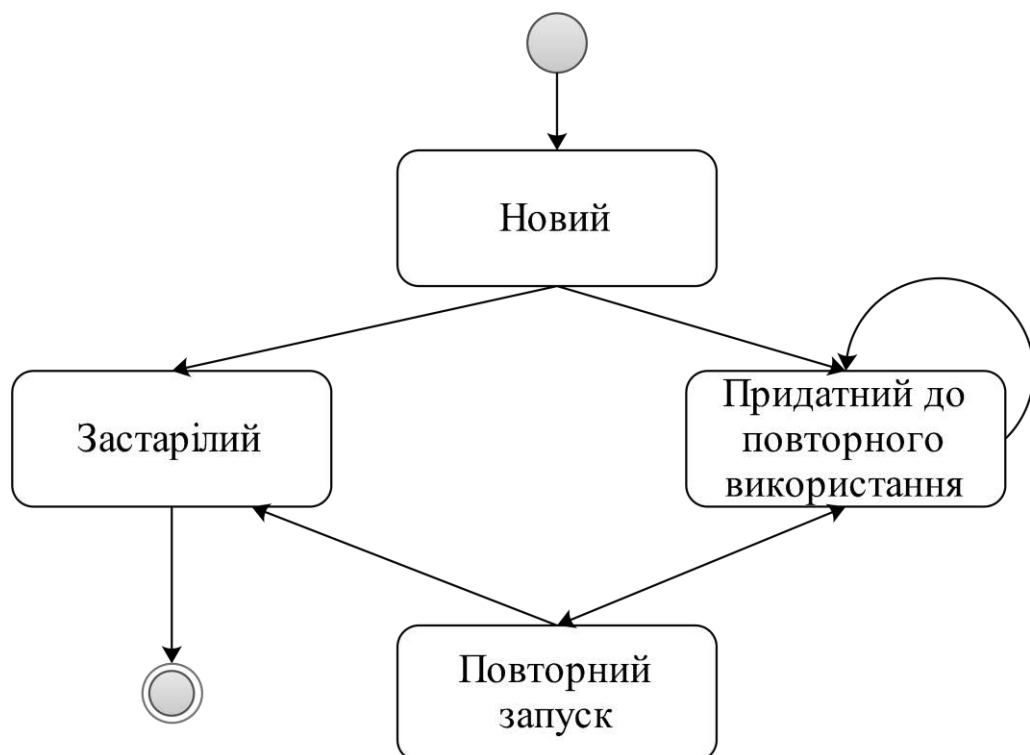


Рисунок 1.4 – Життєвий цикл тесту

До зміни існуючих тестів можуть привести три наступні види діяльності програмістів: створення нових тестів, виконання тестів, зміна коду.

При зміні вхідних даних існуючого тесту будемо вважати, що старий тест припиняє існування, і створюється новий тест. Зміна вихідних даних без зміни траєкторії і / або вхідних даних неможливо [7].

Переваги автоматизації тестування:

- повторюваність - всі написані тести завжди будуть виконуватися одноманітно, тобто виключений «людський фактор»;
- швидке виконання - автоматизованому скрипту не потрібно звертатися з інструкціями і документаціями, це сильно економить час виконання;
- менші витрати на підтримку - коли автоматичні скрипти вже написані, на їх підтримку і аналіз результатів потрібно, як правило, менший час ніж на проведення того ж обсягу тестування вручну;
- звіти - автоматично розсилаються і зберігаються звіти про результати тестування;
- виконання без втручання - під час виконання тестів інженер-тестувальник може займатися іншими корисними справами, або тести можуть виконуватися в неробочий час (цей метод краще, тому що навантаження на локальні мережі вночі знижена).

Недоліки автоматизації тестування:

- повторюваність - всі написані тести завжди будуть виконуватися одноманітно (це одночасно є і недоліком, так як тестувальник, виконуючи тест вручну, може звернути увагу на деякі деталі і, провівши кілька додаткових операцій, знайти дефект, а скрипт цього зробити не може);
- витрати на підтримку - незважаючи на те, що в разі автоматизованих тестів вони менше, ніж витрати на ручне тестування того ж функціоналу - вони все ж є (чим частіше змінюється застосунок, тим вони вищі);

- великі витрати на розробку - розробка автоматизованих тестів це складний процес, так як фактично йде розробка програми, яка тестує іншу програму (в складних автоматизованих тестах також є фреймворки, утиліти, бібліотеки та інше, а все це потрібно тестувати і налагоджувати, а це вимагає часу);

- вартість інструменту для автоматизації - в разі, якщо використовується ліцензійне ПЗ, його вартість може бути досить висока (вільно поширювані інструменти як правило відрізняються більш скромним функціоналом і меншим зручністю роботи);

- пропуск дрібних помилок - автоматичний скрипт може пропускати дрібні помилки, на перевірку яких він не запрограмований (це можуть бути неточності в позиціонуванні вікон, помилки в написах, які не перевіряються, помилки контролів і форм з якими не здійснюється взаємодія під час виконання скрипта) [18].

Для більш ефективного використання автоматизації тестування краще розробити окремі тест кейси перевіряючи:

- базові операції створення / читання / зміни / видалення сутностей (наприклад: створення, видалення, перегляд і зміна даних про користувача);

- типові сценарії використання застосунка, або окремі дії (приклад: користувач заходить на поштову сайт, гортає листи, переглядає нові, пише і відправляє лист, виходить з сайту, все це end-to-end сценарій, який перевіряє сукупність дій);

- інтерфейси, роботи з файлами і інші моменти, незручні для тестування вручну (приклад: система створює певний xml файл, структуру якого необхідно перевірити) [10].

Це і є функціональність, від автоматизації тестування якої можна отримати найбільшу віддачу.

1.3.1 Категорії автоматизації функціонального тестування

Для тестування мобільних застосунків рекомендується послідовно розробляти окремі тестові сценарії (test case) нижче перерахованих категорій.

Функціональні тести необхідні для тестування інтерфейсу стандартних операцій по створенню, читанню, зміни, видалення елементів. Обов'язкові дії в даній категорії:

- додати перевірки можливості додавання кількох різнотипних елементів, з різних класів тестових даних;
- додати перевірки можливості редагування доданих даних;
- додати перевірки можливості видалення даних;
- додати перевірки успішного оновлення даних.

Функціональні позитивні тести необхідні для перевірки коректності завершення дозволених (валідних) операцій, що входять в тести (наприклад, перевірка того, що після збереження коректно введеної інформації виводиться повідомлення про успіх операції. Обов'язкові дії в даній категорії:

- додати перевірки наявності ключових елементів екрану (все кнопки і написи на своїх місцях);
- додати перевірки фактичного виконання операцій створення, читання, зміни та видалення елементів (додані елементи дійсно додалися, після редагування дані змінилися, після видалення дані дійсно видалені, після поновлення наявні на формі дані залишаються незмінними і т.п. ;
- додати перевірки наявності повідомлень про успіх операцій створення, читання, зміни та видалення елементів.

Функціональні негативні тести необхідні для перевірки коректної обробки неправильно введених даних (наприклад, перевірка того, що при спробі введення в текстове поле занадто довгою рядки виводиться повідомлення про помилку). Обов'язкові дії в даній категорії:

- додати перевірки коректності обробки неприпустимих або некоректних даних для операцій створення, читання, зміни та видалення

елементів (при таких даних програма не повинна допускати їх збереження, редагування або поновлення в бази даних, а також не допускати видалення даних, з якими є зв'язку, без попередньої обробки цих зв'язків і т.п.);

- додати перевірки наявності повідомлень про некоректні дії користувача або невірних вихідних даних [10].

1.3.2 Емулятори для автоматизації тестування мобільних застосунків

Для автоматизації тестування використовують емулятори. Фактично, емулятор – це оригінальна заміна пристрою, де є можливість налаштувати аналогічне середовище, таке як в ОС оригінального пристрою [19]. Таким чином, краще використовувати мобільні емуляторів для тестування мобільного застосунку надають перевагу оскільки дозволяють не маючи оригінального фізичного пристрою протестувати як застосунок буде себе вести у різних сценаріях використання.

Переваги використання емуляторів для тестування мобільного застосунку:

- просте налаштування;
- швидка дія;
- допомагає перевіряти і тестувати поведінку мобільного застосунку;
- економічно вигідно.

Недоліки використання симуляторів для тестування мобільного застосунку:

- апаратне обладнання не враховується;
- можливі помилкові спрацьовування;
- отримання неповних даних про результати моделювання, що створює певні труднощі для повного аналізу результатів тестувань.

1.4 Висновки до розділу 1

В цьому розділі було проведено дослідження методів та видів тестування та розроблена класифікація видів тестування.

Було розглянуто особливості тестування застосунків на мобільних пристроях, яке в цілому відповідає загальним принципам тестування, але присутній ряд особливостей, які характерні саме для тестування мобільних застосунків.

Проаналізовано ручне та автоматизоване тестування, та особливості застосування цих способів в процесі тестування. Крім того, проаналізовано проблеми автоматизованого тестування мобільних застосунків. Однією з головних є його трудомісткість: попри те, що воно дозволяє усунути частину рутинних операцій і прискорити виконання тестів, великі ресурси можуть витрачатися на створення та оновлення самих тестів. З іншого боку, при зміні інтерфейсу програми необхідно заново переписати всі тести, які пов'язані з оновленими вікнами, що при великій кількості тестів може відняти значні ресурси. Однак, хоча автоматизоване тестування і має недоліки, пов'язані з часовими, ресурсними витратами на їх створення та виконання, але з точки зору швидкості виконання тестів, продуктивності, точності та можливості повторного використання розроблених тестів, автоматизоване тестування надає суттєву перевагу порівняно з ручним тестуванням.

Дослідження показало, що на сьогодні неможливо повністю відмовитись від ручного тестування але і виникає велика необхідність застосовувати методи автоматизованого тестування в процесі розробки. Найбільш дієвим аспектом тестування, що забезпечує вищий щабель автоматизації є функціональне тестування. Тестування за функціональними критеріями, передбачає здійснення процесу тестування за методом «чорної скриньки».

Таким чином, розробка методики та програмного забезпечення автоматизованого функціонального тестування мобільних застосунків є актуальним завданням.

Метою дипломної кваліфікаційної роботи магістра є розробка методики автоматизованого тестування мобільних застосунків та розробка програмного забезпечення для автоматизації процесу функціонального тестування мобільного застосунку.

Для досягнення цієї мети необхідно вирішити наступні завдання:

- аналіз методів тестування;
- порівняльний аналіз та вибір інструментів автоматизованого тестування мобільних застосунків;
- розробка методики автоматизації тестування мобільних застосунків;
- розробка функціональних тестів для автоматизації тестування мобільного застосунку.

2 АНАЛІЗ ТА ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ МОБІЛЬНИХ ЗАСТОСУНКІВ

Ще нещодавно тестування програм проводилося вручну або самими програмістами, або користувачами, що неможна назвати системним підходом і до того ж це не дозволяло оцінювати якість коду. Трохи пізніше тестування виділилося в окрему галузь знань у складі розробки програмного забезпечення, але швидко прийшло розуміння того, що тестування вручну неефективне, оскільки вимагає великих трудових ресурсів і багато часу.

Для автоматизації потрібно замінити тестувальника на інструменти, які вміють взаємодіяти з одним або декількома інтерфейсами програми. Таким чином, необхідні інструментальні засоби для виконання та формування набору тестів [20].

Перші засоби автоматизації тестування практично представляли собою бібліотеки, які можна було використовувати для написання тестів, що вимагало від тестувальника вміння програмувати на рівні розробника.

На ринку засобів автоматизованого тестування сьогодні представлено багато продуктів [21] - [22]. Сучасні засоби автоматизованого тестування дозволяють створювати автоматизовані тести з мінімальною участю людини (на основі дій користувача автоматично генерується сценарій тестування), проте вони не можуть гарантувати якість, яку дає метод програмування тестів.

Усі інструменти називаються стеком автоматизованого тестування. Загальна схема взаємодії інструментальних засобів при створенні автоматизованих тестів зображена на рисунку 2.1.

Як зображено на рисунку 2.1 інструменти для автоматизації тестів можна поділити на:

- інструментальні засоби для взаємодії з мобільним застосунком (драйвери, надстройки);
- засоби для написання та виконання тестів (фреймворки).

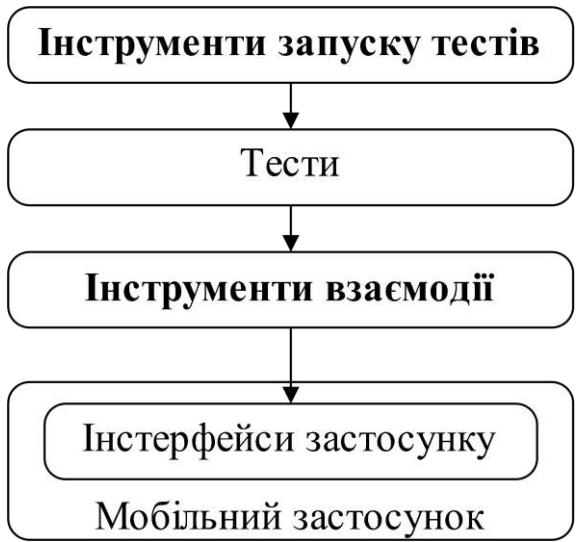


Рисунок 2.1 – Загальна схема взаємодії інструментальних засобів автоматизованого тестування

2.1 Інструментальні засоби для взаємодії з мобільним застосунком

Як показало дослідження, для взаємодії з інтерфейсами мобільного застосунку інструментальні засоби можна поділити на драйвери та надбудови. На рисунку 2.2 зображені найбільш відомі інструменти серед розробників автоматизованих тестів [20].

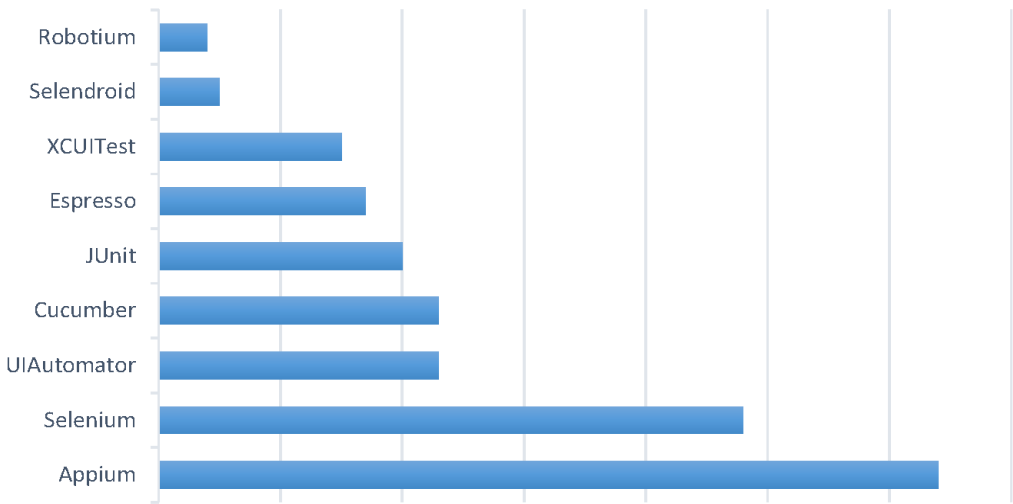


Рисунок 2.2 – Популярні інструменти для розробки автоматизованих тестів

2.1.1 Вибір драйверу

Утиліти автотестування, як і інші програми, можуть взаємодіяти з застосунком тільки через програмний інтерфейс. Для роботи через інші інтерфейси існують спеціальні програми – драйвери.

Драйвер – програма, яка надає API для одного з інтерфейсів програми [23]. Для кожного інтерфейсу. Наприклад, коли драйвер отримує для GUI команду натиснути на кнопку, він сприймає її через API і відсилає в тестований застосунок, де ця команда перетворюється в клік по графічній кнопці «Меню» [20]. Для взаємодії з API застосунка драйвери не потрібні або майже не потрібні – взаємодія програмна. А ось при роботі з іншими інтерфейсами без них не обійтися.

Найбільш складними зазвичай є драйвери для GUI, оскільки цей інтерфейс дуже відрізняється від звичайного для програми спілкування кодом. При цьому в автоматизованому тестуванні мобільних застосунків GUI вважається найбільш актуальним, тому в інтеграційному тестуванні використовувати найчастіше доводиться саме його. Найбільш популярні драйвера для GUI в мобільному тестуванні – UIAutomator і Espresso для Android [24], XCUITest – для iOS [25]. В таблиці 2.1 наведені результати порівняння драйверів.

Таблиця 2.1 – Порівняння драйверів мобільного тестування

Інструмент	Android	iOs	Мови	Доступ до коду	>1 застосунку	Реальні пристрої
UIAutomator	+	-	Java, Kotlin	-	+	+
Espresso	+	-	Java, Kotlin	+	-	+
XCUITest	-	+	ObjectiveC Swift	-	-	+
Robotium	+	-	Java	-	-	+
Selendroid	+	-	Більшість	-	-	+

У мобільному тестуванні драйверів трохи, і часто вони розробляються тими ж компаніями, що і операційні системи. Для Android є два офіційних драйвера: UIAutomator, який на сьогоднішній день має версію 2.0, і Espresso. Обидва вони входять в Android Testing Support Library, розробляються компанією Google і добре документовані. Крім них, існують проекти Robotium і Selendroid, які розробляються сторонніми компаніями.

В цілому, якщо потрібно тестувати тільки застосунок, без урахування його взаємодії з системою, і є бажання і можливість працювати з вихідними кодами, краще використовувати Espresso. До того ж в ньому реалізовані корисні функції на кшталт автоматичної синхронізації тестів з UI застосунки, і можна не писати різні команди очікування.

Якщо потрібно протестувати застосунок в зв'язці з іншими або з функціями системи, а доступ є лише до .apk, вибирайте UIAutomator.

Selendroid, і Robotium розвиваються набагато менш активно, ніж драйвери від Google, і їхня аудиторія значно вужче. Для Robotium тести потрібно писати на Java, а Selendroid підтримує протокол WebDriver, що дає можливість використовувати практично будь-який популярна мова програмування [21]. Обидва інструменти можуть звертатися тільки до одного застосунку, не вимагають доступу до вихідного коду і підтримують роботу на емуляторах і реальних пристроях.

2.1.2 Вибір надбудови

Коли функціоналу драйвера не вистачає або він незручний і складний, над ним з'являється ще один рівень, який я буду називати надбудовою [26]. Надбудова – програма, яка взаємодіє з застосунком через один або кілька драйверів, підвищуючи зручність їх використання або розширюючи їх можливості.

У надбудови можуть бути функції перераховані нижче.

Модифікація поведінки (без зміни API). Наприклад, додаткове

протоколювання, валідація даних, очікування виконання дії протягом певного часу.

Підвищення зручності і / або рівня абстракції API через використання синтаксичного цукру – зручних назв функцій, більш коротких звернень до них, уніфікованого стилю написання тестів; неявне управління драйвером, коли, наприклад, він ініціалізується автоматично, без необхідності прописувати кожен таку дію вручну; спрощення складних команд на кшталт вибору події з календаря або роботи зі списками, які прокручуються; реалізацію альтернативних стилів програмування, таких як процедурний стиль або fluent.

Уніфікація API драйверів. Тут надбудова надає єдиний інтерфейс для роботи відразу з декількома драйверами. Це дозволяє, наприклад, використовувати один і той самий код для тестів на iOS і Android, як у популярній надбудові Appium.

Є ряд інструментів, доступні на ринку, таких як [27]: Appium, Calabash та інші. Порівняльна таблиця 2.2 наводить основні переваги та недоліки найбільш відомих на сьогодні надбудов.

Таблиця 2.2- Порівняння надбудов для автоматизації тестування

Інструмент	Android	iOs	Мови	Доступ до коду	>1 застосунку	Реальні пристрої
Calabash	+	-	Ruby JRuby	-	-	+
Appium	+	+	Більшість	-	+	+
Earl Grey	-	+	ObjectiveC Swift	+	-	+
WebDriver Agent	-	+	ObjectiveC Swift	-	+	+

Провівши аналіз доступних драйверів для тестування мобільних застосунків, для подальшої роботи було обрано Appium Driver. Порівняльну характеристика драйверів наведено в таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика драйверів для тестування мобільних застосунків Appium – відомий інструмент автоматизації мобільних застосунків (Android та iOS), з можливістю написання тестів на великій кількості мов програмування / фреймворками.

Позитивні особливості Appium [28]:

- порівняно легко встановити, налаштувати, запустити перший тест;
- підтримується Android та iOS;
- підтримує майже всі мови програмування;
- багато способів ідентифікації елементів;
- синтаксис дуже схожий на Selenium, завдяки чому можна дуже швидко почати писати тести, не треба вивчати новий API;
- підтримує автоматизацію гібридних, нативних та вебзастосунків;
- відкритий код (безкоштовно);
- дозволяє писати DeepTests (тобто тести за структурою поточної Activity та ініціалізації доступних на ній елементів, з якими треба працювати і змінюють список перевірок в залежності від того що зараз представлено на екрані – а не просто діють по строго певних кроків).

Негативні особливості Appium:

- не завжди знаходяться елементи з першого разу, а якщо елемент вже знайдено не завжди є можливість читати властивості знайдених елементів, даний недолік компенсується написанням функцій обгортки;
- немає можливості розпізнавати елементи по картинці – тобто для автоматизації мобільних ігор Appium не підходить.

Великою перевагою Appium є те, що він дає доступ до тестування одночасно різних застосунків та браузерів на застосунку, тобто при наявності в мобільному застосунку зовнішніх джерел (наприклад, логін та реєстрація через Facebook або Gmail) є можливість доступу та, як наслідку,

покриття автоматизованими тестами і цього функціоналу.

Арріум дозволяє створювати та зберігати безліч варіантів пристроїв для запуску автоматизованого тестування.

2.2 Вибір інструментальних засобів для написання та виконання тестів

З іншого боку тестів знаходиться фреймворк запуску. В цій статті я буду коротко називати його “фреймворк”. Фреймворк – це програма для формування, запуску і збору результатів запуску набору тестів [29].

До завдань фреймворка входять:

- формування, угруповання і упорядкування набору тестів;
- розпаралелювання набору (опціонально);
- запуск тестів;
- збір результатів їх виконання;
- формування звітів про виконання (опціонально).

Можна помітити, що ці функції не пов’язані з тестуванням тільки мобільних застосунків. Справа в тому, що фреймворк не повинен забезпечувати взаємодію тестів і застосунку – він працює тільки з тестами, і тип програми не має значення [20].

Якщо драйвери і надбудови знаходяться між тестами і застосунком, то фреймворк знаходиться над тестами, організовуючи їх запуск. Звичайно, в деяких фреймворках є власні драйвери для роботи з застосунками, але це зовсім необов’язкова умова.

2.2.1 Вибір мови програмування

Важливу роль також відіграє і мова програмування, що використовується в автоматизації. Трійка найбільш використовуваних мов програмування виглядає так: Python, Java, Ruby [30].

Java – це мова програмування загального призначення, яким володіє корпорація Oracle. Java побудована на принципах об'єктно-орієнтованого програмування. Мова Java дотримується принцип WORA («Write Once, Run Anywhere»), який дає багато переваг для крос-платформного тестування [31].

Java використовується для підтримки внутрішніх корпоративних систем в багатьох великих корпораціях. Програми, написані на мові Java, працюють більш ніж на 3 мільярди пристроїв. Незважаючи на те, що для юніт тестування найпопулярнішою платформою є JUnit, багато платформ для автоматизації тестування з відкритим вихідним кодом були розроблені на мові Java. Автоматизоване тестування в браузері для вебсайту або вебзастосунки можна виконати за допомогою JUnit з Selenium WebDriver.

Python також використовується в автоматизації тестування, пропонує відкритий вихідний код, використовується в машинному навчанні та багато іншого. Основною перевагою, яке Python має перед іншими мовами програмування для автоматизації тестування, є легкість в освоєнні завдяки читабельності мови [32].

Згідно з дослідженням Stack Overflow Developer Survey (2019) [33], колосальні 73,1% проголосували за Python як найбільш затребуваний мову програмування, що вказує на популярність мови Python.

Дана мова програмування високорівнева, плюс має відмінну динамічну систему. Python найкраще підходить для автоматизації тестування.

Створений Microsoft, C # також є одним з домінуючих мов в сфері автоматизації. Він дотримується концепції ООП і є одним з найпоширеніших мов, що використовують .NET Framework [34]. 67% респондентів в Stack Overflow Developer Survey (2019) вважають C # найбільш підходящим мовою для автоматизації, програмування і багато чого іншого. C # в сфері автоматизованого тестування добре підходить для застосунків на операційних системах Android, Windows і iOS. Остання версія C # - 8.0 [33].

Таблиця 2.3 – Порівняння мов програмування

Мова	Переваги	Недоліки
Java	PageFactory, спрощує код для автоматизації та є можливість писати дуже прості для розуміння тести. У Java існує велика ком'юніті, нею володіють багато розробників і на ній вже написана величезна кількість інструментів.	Код читається складніше, ніж, наприклад, в Python. Повідомлення про помилки, може бути складно зрозуміти.
Python	Використовується для вирішення практично будь-якої задачі на будь-якій платформі. Бібліотеки Selenium і Appium та безліч інших засобів полегшують роботу тестувальникам автоматизаторів зокрема в крос-браузерному тестуванні. Python дозволяє домогтися багато чого з меншою кількістю коду. Python легше вивчити, має велике ком'юніті і відкритий вихідний код.	Незвичний синтаксис, якщо переходити на Python з іншої мови. Великі затрати ресурсів комп'ютера.
C#	Розроблено багато платформ для автоматизованого тестування на Selenium або для тестування в браузерах. Також існує широкий ряд тестових середовищ, через що багато автоматизаторів використовують C# для написання тестових випадків для крос-браузерного тестування	Microsoft засоби розробки .NET не завжди безкоштовні, і операції командного рядка можуть проходити з проблемами.
Ruby	У Ruby інтернет співтовариство користувачів велике але не настільки як в Java та Python. Код програми виходить менший за рахунок оптимізованого синтаксису. Мовою Ruby можна програмувати в платформі Selenium.	Автоматизація веб, Selenium. При взаємодії з іншими засобами розробник може зіткнутися зі складнощами.

Ця мова розробки повільно і неухильно набирає обертів в області автоматизованого тестування. Завдяки потужним можливостям мови і його сумісності з Selenium WebDriver, багато автоматизаторів схильні використовувати C # для тестування автоматизації та крос-браузерного тестування. Шаблон проєктування Page Object Model (POM) дозволяє тестувальникам написати тестовий код ефективним і легко підтримуваним [35].

Ruby також є мовою програмування, за допомогою якого можна писати автоматизовані тести. Дана мова зараз набирає популярність в сфері автоматизації та автоматизованому тестуванні в браузерах. Він має відкритий вихідний код, є досить простим і продуктивним. Ruby, на зразок Python, дуже простий у вивченні і подальшій реалізації. Потужним мовою для програмування і автоматизації його робить те, що він має простий і зручний для людини синтаксис і гнучку об'єктно-орієнтовану архітектуру [34].

Підводячи підсумки можна сказати, що Python і Java є сьогодні кращими мовами для автоматизації. На Ruby можна швидко почати писати повноцінний фреймворк для тестування. Даний вид тестування обходиться дешевше і набагато швидше, ніж ручне тестування. У реальності відбувається так, що чим краще фахівець програмує, тим правильніше і точніше зможе автоматизувати.

2.2.2 Вибір середовища для написання та виконання тестів

PyCharm це одна з кращих IDE з відкритим вихідним кодом, орієнтованих на мову Python. Якщо займатися розробкою програм самостійно або в невеликому стартапі, ця IDE прекрасно підійде [36].

PyCharm має кілька варіантів ліцензії: є відкритою та безкоштовною, а є і платні версії. Можна вибрати ту, яка найкраще відповідатиме поставленим вимогам. PyCharm має багатий функціонал, що підтримує Python у всіх аспектах програмування [37].

У цій IDE можна знайти високоякісне автодоповнення коду, просунутий відладчик, перевірку коду і т.ін. PyCharm підтримує вебпрограмування, а також код, написаний іншими мовами, що мають відношення до баз даних.

PyCharm дозволяє розширювати наявний в ній функціонал за допомогою зовнішніх плагінів. Якщо працювати над великим проєктом, у PyCharm в будь-який час можна задіяти інтеграцію системи контролю версій [36].

Sublime Text, написаний інженером з Google з мрією про кращий текстовий редактор, є вельми популярним редактором коду. Доступний на всіх платформах, Sublime Text має вбудовану підтримку редагування Python-коду, а також багатий набір розширень, званих пакетами, які розширюють можливості синтаксису і редагування [38].

Встановити додатковий Python-пакет може бути непросто - все пакети Sublime Text написані на Python, тому для установки пакетів спільноти часто може знадобитися виконати Python-скрипт безпосередньо в редакторі.

Eclipse відкрите та безкоштовне середовище розробки. Доступне для Linux, Windows і OS X, Eclipse є безкоштовною та відкритою для розробки на Java. Існує безліч розширень і аддонів, які роблять Eclipse корисним для різного роду завдань [39].

Одним з таких розширень є PyDev, що надає інтерактивну консоль Python і можливості для налагодження і автодоповнення коду. Встановити його просто: запустіть Eclipse, виберіть Help → Eclipse Marketplace, потім знайдіть PyDev. Натисніть «Install» і при необхідності перезапустити Eclipse.

Visual Studio Code це хороший редактор коду з відмінним функціоналом [36]. Якщо працювати над маленьким проєктом, варто подумати про використання VS Code. Цей редактор має гарний користувацький інтерфейс, а також такі корисні функції як підсвічування синтаксису і автодоповнення.

Таблиця 2.4 – Переваги та недоліки середовищ програмування Python

Середовище	Переваги	Недоліки
PyCharm	Підтримка бібліотек Selenium і Appium та безліч інших засобів полегшують роботу тестувальників автоматизаторів. Велике інтернет ком'юніті. Можна редагувати, запускати і налагоджувати Python-код.	Може повільно завантажуватися, а настройки за замовчуванням, деколи доведеться виправляти для існуючих проєктів.
Sublime	Велика кількість користувачів. Як редактор коду, Sublime Text швидкий, легкий і має хорошу підтримку.	Sublime Text не є безкоштовним, є пробний період скільки завгодно. Установка розширень може бути складною. В редакторі немає підтримки налагодження і запуску коду.
PyDev (Eclipse)	Надає всі переваги розробки великого середовища Eclipse.	Для новачків Python у Eclipse може бути тяжким, оскільки IDE велике і вимагає більше досвіду для повноцінного використання.
VS Code	Завдяки Electron, VS Code доступний на всіх платформах з дивно великою функціональністю. Крім того, вихідні коди можна знайти у відкритому доступі.	Замішаний Electron, значить VS Code не нативний застосунок. Продукти Microsoft є платними.

Технологія Intellisense робить код застосунку дуже швидкодіючим. У порівнянні з іншими IDE VS Code має більший час завантаження, хоча компанія-виробник заявляє, що він легкий і дуже швидкий.

Visual Studio поставляється з вбудованим терміналом, але через брак багатьох корисних функцій я б не радив використовувати його для великих проєктів.

Було проведено аналіз наявних фреймворків для розробки тестів для

мобільних застосунків, для подальшої роботи було обрано PyCharm, за рахунок підтримки бібліотек Selenium і Appium та безліч інших засобів полегшують роботу тестувальників автоматизаторів, а також через велике інтернет ком'юніті.

2.3 Висновки до розділу 2

Як показало дослідження інструменти для автоматизації тестів можна поділити на: інструментальні засоби для взаємодії з мобільним застосунком (драйвери, надстройки); засоби для написання та виконання тестів (фреймворки).

Для розробки автоматизованих тестів та прискорення їх тестування серед проаналізованих було обрано наступні засоби.

Мова програмування для написання тестів було обрано Python через свою популярність та інтернет спілук. Тестовий фреймворк для написання тестів на мові Python було обрано PyCharm.

Драйвер для запуску тестів Appium через свою кросплатформність, він дає доступ до тестування одночасно різних застосунків та браузерів на застосунку, тобто при наявності в мобільному застосунку зовнішніх джерел (наприклад, логін та реєстрація через Facebook або Gmail) є можливість доступу та, як наслідку, покриття автоматизованими тестами і цього функціоналу.

UIAutomator, було обрано як драйве взаємодії з інтерфейсом оскільки він дозволяє працювати не тільки з самим застосунком, але і протестувати застосунок в зв'язці з іншими застосунками та з функціями системи. Повністю підтримує метод тестування «чорної скриньки».

3 РОЗРОБКА МЕТОДИКИ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Розробка методики автоматизованого тестування мобільних застосунків

На основі проведеного дослідження методів та інструментальних засобів була розроблена методика автоматизованого тестування мобільних застосунків (рис.3.1), яка передбачає використання інструментальних засобів для прискорення процесу тестування.

Методика була розроблена з використанням методу чорної скриньки являє собою сукупність кроків, які необхідно виконати, аби етап тестування проєкту виконувався швидше. Методика складається з кроків перерахованих нижче.

Крок 1. Створення сценаріїв тестування мобільного застосунку. На цьому кроці необхідно проробити існуючу документацію проєкту та базуючись на цьому розробити сценарії тестів.

Крок 2. Налаштування серверу та емулятору. Проводиться налаштування серверу та емулятору для тестування на відповідних платформах.

Крок 3. Пошук елементів екрану мобільного застосунку. Для написання тестів необхідно знайти всі елементи для проведення успішного автотесту.

Крок 3. Опис знайдених елементів в проєкті тестування.

Крок 4. Групування елементів в окремі класи – об'єкти екранів.

Крок 5. Використання анотацій кроків при програмуванні дій тестів.

Крок 6. Попередня ініціалізація об'єктів екранів безпосередньо перед початком тесту або тестового сценарію.

Крок 7. Запуск автоматизованих тестів.

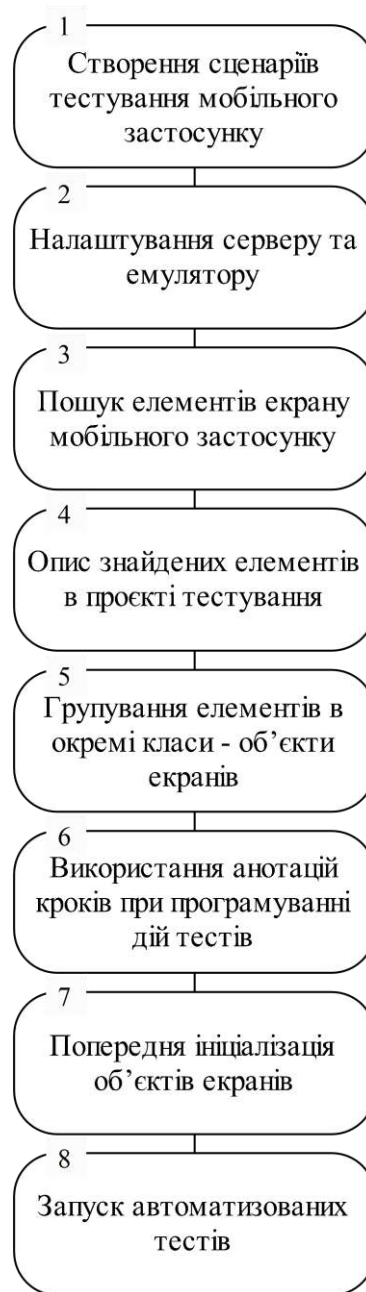


Рисунок 3.1 – Методика прискорення автоматизованого тестування мобільних застосунків

Таким чином, розроблено методику за допомогою якої можна досягти прискорення проведення тестування мобільних застосунків завдяки використанню інструментарії для автоматизації тестування. Данна методика може бути використана при тестуванні будь-яких мобільних застосунків.

3.2 Розробка сценаріїв тестування на основі діаграми переходів між станами тестованого мобільного застосунку

З метою кращого розуміння функціональності застосунку, що необхідно тестувати, та подальшого створення сценарію тестування та написання автоматизованих тестів біло розроблено діаграму переходів між станами. Діаграма переходів між станами зображена на рисунку 3.2, вона демонструє логіку взаємодії з інтерфейсом застосунку, а саме логіку взаємодії з інтерфейсом на етапі входу та реєстрації користувача в застосунку.

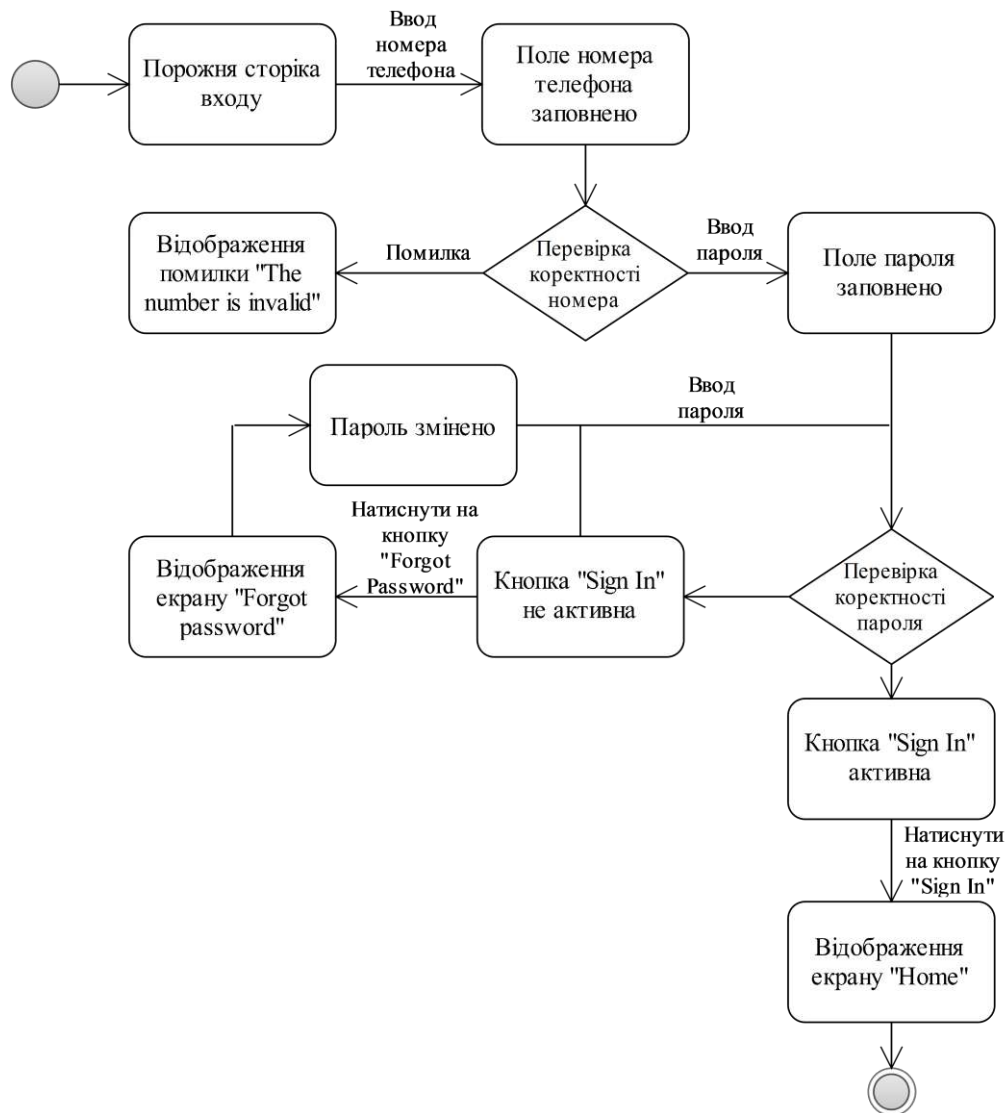


Рисунок 3.2 – Діаграма переходів між станами тестованого застосунку

Базуючись на рисунку 3.2 були розроблені тестові сценарії для перевірки мобільного застосунку, визначені необхідні параметри вводу та очікувані відповіді застосунку на них. Таким чином, основі цієї діаграми розроблена таблиця (рис.3.3) з позитивні та негативні тестовими сценаріями, для автоматизації тестування.

1	A	B	C	D	E	F	G
1	Section (Screen)	Description	PreConditions	Steps to Reproduce	Expected result	iOS build 1.5 2020-09-21	Android dev 0.7.5-2020-09-21
15	Логін	Логін (позитивний тест-кейс)	1. Користувач знаходиться на екрані Login	1. Ввести в поле Phone Number валидний номер 2. Ввести в поле Password валидний пароль 3. Натиснути кнопку Sign In	1. Користувач заповнює 2. Відкривається екран Home->Daily	passed	passed
16		Логін з вводом невалидного номера телефону	1. Користувач знаходиться на екрані Login	1. Ввести в поле Phone Number невалидний номер телефону (наприклад, менше 9 цифр) 2. Натиснути кнопку Date на клавіатурі	1. Під полем Code и Phone Number з'явиться повідомлення "The number is invalid. Please check it once again!" 2. Кнопка "Sign In" закривається	passed	passed
17		Логін з вводом невалидного пароля	1. Користувач знаходиться на екрані Login	1. Ввести в поле Password невалидний пароль 2. Натиснути кнопку Date на клавіатурі	1. З'явиться повідомлення "Wrong phone number or password" 2. Кнопка "Sign In" закривається	passed	passed
18		Логін через Facebook	1. Користувач знаходиться на екрані Login, коли раніше провів реєстрацію через FB	1. Натиснути на кнопку FB 2. В появившомуся поп-апе натиснути кнопку "Продолжить" 3. На екрані FB в поп-апе натиснути кнопку "Открыть"	1. Користувач перенаправлений на екран логіна, з'явиться логгер, після чого користувач перенаправлений на екран Home->Daily	passed	passed
19		Логін через AppleID	1. Користувач знаходиться на екрані Login	1. Натиснути на кнопку Apple 2. В появившомуся поп-апе натиснути кнопку "Продолжить с код-переводом" 3. На екрані пароля ввести "1111"	1. Користувач перенаправлений на екран логіна, з'явиться логгер, після чого користувач перенаправлений на екран Home->Daily	passed	passed
20		Логін через Google	1. Користувач знаходиться на екрані Login	1. Натиснути на кнопку Google 2. В появившомуся поп-апе натиснути кнопку "Продолжить" 3. На екрані Google натиснути на акк	1. Користувач перенаправлений на екран логіна, з'явиться логгер, після чого користувач перенаправлений на екран Home->Daily	passed	passed
21		Логін через Twitter	1. Користувач знаходиться на екрані Login	1. Натиснути на кнопку Twitter 2. В появившомуся поп-апе натиснути на кнопку Open 3. Натиснути на кнопку Authenticate app	1. Користувач перенаправлений на екран логіна, з'явиться логгер, після чого користувач перенаправлений на екран Home->Daily	passed	passed
22		Отображення кнопок	1. Користувач раніше був зареєстрований з кимсь-то з кнопок	1. Відкрити аплі 2. Обрати уважність на кнопки на екрані логіна 3. Заповнити 4. Подивитись на екран	1. На екрані Login під полем логіна и пароля з'явиться повідомлення "Wrong phone number or password"	passed	passed

Рисунок 3.3 – Розроблена таблиця з тестовими сценаріями для тестування мобільного застосунку

3.3 Розробка архітектури взаємодії програмного забезпечення автоматизованих тестів

На рисунку 3.4 зображено архітектуру взаємодії засобів та програмного забезпечення для виконання автоматизованих тестів.

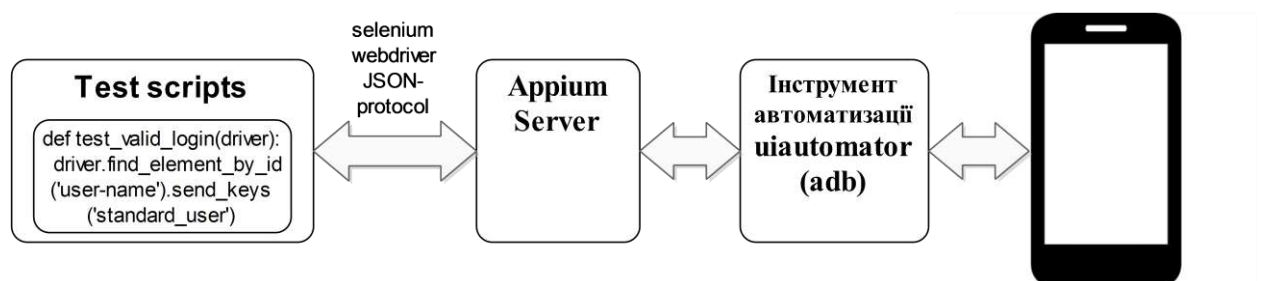


Рисунок 3.4 – Архітектура взаємодії програмних засобів автоматизованого тестування мобільних застосунків

Appium надає клієнтські бібліотеки, щоб взаємодіяти з REST API. Цей метод викликає HTTP-запит, і отримує відповідь від кінцевої точки.

Драйвер виконає тестовий скрипт, який приходить в JSON-форматі від AppiumDriver сервера, щоб отримати екран. Назад Appium поверне XML-документ, що представляють ієрархію UI-елементів. Структура документа може відрізнятися, в залежності від платформи.

Як представлено на рисунку вище, під час виконання тесту, Appium відправляє команди до UIAutomator. На мобільному пристрої bootstrap.jar грає роль TCP-сервера, який можна використовувати, щоб відправляти команди. Команди, в свою чергу виконуються на Android-пристрої засобами UIAutomator. Після чого, Appium отримує HTTP-запит у форматі JSON від клієнта (в даному випадку тестованого мобільного застосунку) и перетворює їх у залежності від платформи, на якій працює.

3.4 Структура програмного забезпечення автоматизованих тестів для мобільного застосунку

Для доступу та можливості взаємодії з окремими елементами на сторінках мобільного застосунку необхідно знати їх id (індивідуальний ідентифікатор елементу в обраному застосунку) або XPath (або XML Path Language — це мова запитів, для вибору вузлів з XML документів).

Для того, щоб отримати ідентифікатори використовується способів через Appium inspector із відкритим файлом .apk, чи .app відповідно. Appium Inspector було використано для пошук елементів оскільки:

- підтримує метод тестування «чорної скриньки», тобто не потрібно розбиратися в коді розробника, шукати файли окремих сторінок;
- візуально видно, на якій саме сторінці застосунку обираються елементи в даний момент, та при кліку на певний елемент видно усі атрибути обраного елементу (рис. 3.5);

- якщо розробником застосунку не було присвоєно ідентифікатор певного елемента, Appium Inspector визначає XPath елемента (рис 3.6). Використовуючи пошук елементів через інструмент розробника, в разі відсутності ідентифікатору елемент залишається невизначеним.

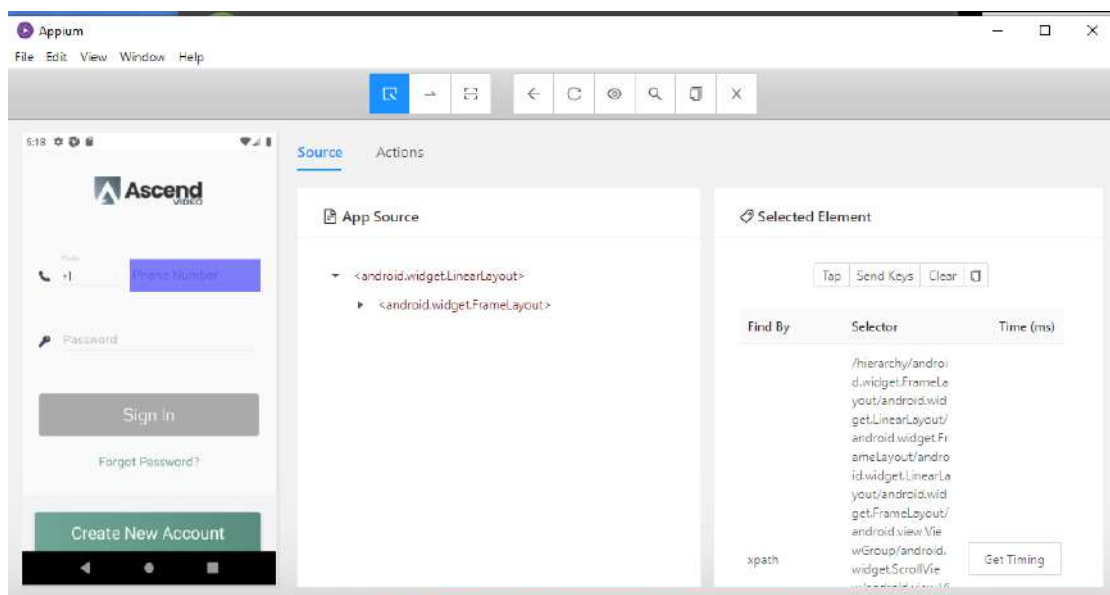


Рисунок 3.6 – Визначення XPath елемента для якого відсутній ідентифікатор у Appium Inspector

Після знаходження елементів, вони були описані в засобі створення автоматизованих тестів.

Проект створено в середовищі розробки PyCharm, структура проекту зображена на рисунку 3.7. В папці Test є дві вкладені папки: Android та iOS, в яких знаходяться готові автоматизовані тести, позитивні та негативні, для тестування логіну (Login_test_success.py, Login_test_failed.py), автоматизовані тести для негативних та позитивних сценаріїв реєстрації (Registration_test_failed.py, Registration_test_siccess.py), головної сторінки (Home_screen_test.py) та основної функції застосунку створення груп (Create_group.py), приклади в додатку А, файл main.py був створений для опису усіх методів, констант, опису елементів, виклику драйверу.

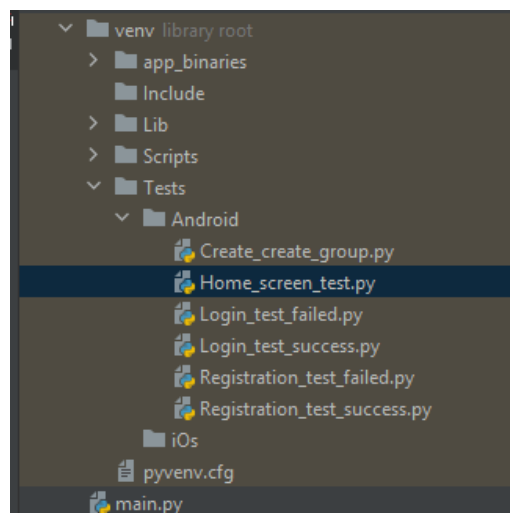


Рисунок 3.7 – Структура проекту автоматизованих тестів для мобільного застосунку

Для опису елементів кожного окремого екрану створюється клас, в конструкторі якого кожному елементу екрану надається ідентифікатор, що відповідає даному елементу застосунку, далі, при написанні тестів, сторінка ініціалізується, на етапі відкриття. Ініціалізація екрану потрібна для того, щоб тест при запуску мав доступ до всіх його елементів, а не шукав елемент при кожному його виклику.

3.5 Використані методи та функції при створенні автоматизованих тестів для мобільного застосунку

Для структуризації коду були створені методи для кроків тестових сценаріїв. Крім того така структуризація також необхідна для того, аби в разі падіння тесту можна було одразу зрозуміти, на якому кроці та за яких умов це сталось. А тест, з попереднім створенням тестових кроків наведено в додатку А.

Для опису кроків створюється окремий метод (для кожного сценарію окремий), в якому також створюються методи (назва методу - назва кроку) з описом дій, які необхідно зробити для того, аби виконати крок. Також необхідно було виконати ініціалізацію екранів, а саме елементів які знаходяться на ній. Ініціалізація об'єкта відбувається у методі, що називається конструктором. Далі, при описі кроків тесту, створюється об'єкт потрібного екрану, за рахунок чого всі елементи на екрані, що описані в конструкторі, знаходяться в один крок.

Таким чином зрозуміло, що попередня ініціалізація сторінок пришвидшує виконання автоматизованого тесту. Наприклад, для кроку “ввести номер телефону”, драйверу необхідно:

- знайти поле вводу телефону.
- натиснути (кликнути) на поле вводу.
- очистити поле, якщо воно не було порожнім.
- ввести номер телефону.

За наявності попередньої ініціалізації елементів екрану пункт “знайти поле вводу телефону” стає неактуальним, адже всі елементи на даному екрані були знайдені раніше. Як видно на прикладі, один тестовий сценарій містить чотири кроки, на початку яких треба знайти елемент. Попередня ініціалізація допомагає зекономити час на першому кроці.

В таблиці 3.1 наведені прикладі функцій та методів, які були використані при розробці автоматизованих тестів.

Таблиця 3.1 – Функції та методи автоматизованого тестування

Метод, функція	Опис
find_element_by_id()	Знайти елемент за його ідентифікатором.
find_element_by_path()	Знайти елемент за його xpath ідентифікатором.
driver.click()	Зробити тап по необхідному елементу екрану.
implicitly_wait()	Очікування.
driver.quit()	Закрити та завершити роботу застосунку.
driver.back()	Повернутися до попереднього стану екрану.
assert()	Перевіряє отриманий результат тесту.
TouchAction()	Метод для виконання дій користувача, наприклад, такого як проведення пальцем по екрану.
findLoginScreenElements()	Метод ініціалізація всіх елементів екрану входу.
testValidLogin()	Метод позитивного тесту входу до застосунк.
testInvalidLoginTel()	Метод негативного тесту входу до застосунк.

Для роботи з пристроєм також необхідно створити екземпляр «driver». Конструктор даного класу очікує на вході два параметри: Desired capabilities – JSON-об'єкт (набір пар ключ-значення), відправлений клієнтом серверу; Desired capabilities описують особливості створюваної сесії (ім'я пристрою / емулятора, операційну систему (ОС), версію ОС, що запускається і т.д.).

Приклад capabilities:

```
desired_capabilities = {
    "platformName": "Android",
    "platformVersion": "9.0",
    "deviceName": "Android Emulator",
    "app":
"/Users/DoneIT/PycharmProjects/Project2/venv/app_binaries/Ascend-
devTest.apk" }
```

Сам драйвер пристрою оголошується як:

```
driver = webdriver.Remote('http://127.0.0.1:4723/wd/hub',  
desired_capabilities=desired_capabilities).
```

3.6 Висновки до розділу 3

В цьому розділі була розроблена методика автоматизованого тестування та представлено її детальний опис. Розроблені діаграми переходів та на основі цієї діаграми сценарії тестування у вигляді таблиць.

Розроблена архітектура взаємодії інструментальних засобів та описано послідовність їх взаємодії. Розроблена структура проєкту.

Наведено методи та функції які були використані при розробці автоматизованих тестів для проведення тестування мобільного застосунку.

4 ІНСТРУКЦІЇ ДО ВИКОРИСТАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ АВТОМАТИЗОВАНИХ ТЕСТІВ

4.1 Призначення та умови використання програмного продукту

Програмне забезпечення це набір автоматизованих тестів, які призначений для проведення функціонального тестування мобільного застосунку. Розроблюване програмне забезпечення покликане допомогти тестувальнику у процесі тестування мобільного застосунку на мобільній платформі Android та iOS.

Програмний продукт може бути використаний на будь-яких пристроях з відповідним встановленим мобільним застосунком, але з необхідністю внесення основних характеристик цього пристрою в програму.

4.2 Алгоритм роботи з програмним забезпеченням для автоматизації функціональних тестів

Необхідні кроки для виконання автоматизованих тестів на емуляторі було описано у вигляді алгоритму, який зображено на рисунку 4.1. Алгоритм складається з кроків описаних нижче.

Крок 1. Запуск серверу Appium.

Крок 2. Запуск емулятору Android або IOS мобільного пристрою. В середовищі Android SDK запускається емулятор за допомогою засобу UIAutomator.

Крок 3. Встановлення застосунку на емуляторі мобільного пристрою. Встановлюється остання версія застосунку, для пристроїв Android це файл з розширенням *.apk, що необхідно протестувати.

Крок 4. Запуск тесту в середовищі PyCharm. Цей крок передбачає відкриття проекту в середовищі PyCharm та запуск його на виконання.

Крок 5. Отримання результатів виконання тесту. Після завершення виконання тесту в середовищі PyCharm в області повідомлень користувач

отримає всю необхідну інформацію по виконаному тесту, за необхідністю результати можна зберігати у файл.



Рисунок 4.1 – Алгоритм роботи автоматизованих функціональних тестів на емуляторі

Таким чином розроблений алгоритм запуску автоматизованих функціональних тестів на емуляторі представляє собою послідовність кроків, які необхідних виконати для запуску автоматизованих тестів.

4.3 Вхідні та вихідні дані

Вхідними даними може бути опис пристрою на якому необхідно провести тестування, якщо цього пристрою немає в списку.

У якості вихідних даних користувач отримає список повідомлень про виконанні тести або посилки що виникли у ході виконання програми.

4.4 Налаштування емулятора та серверу для запуску автоматизованих тестів

Для роботи з програмним забезпеченням, необхідно виконати ряд інструкцій з його запуску та налаштувань.

Спочатку потрібно впевнитись, що на комп'ютері встановлене все необхідне програмне забезпечення для роботи а саме: Java SDK, Android Studio, Android SDK Tools (adb, UIAutomator), Appium, PyCharm. У випадку якщо відсутній якийсь інструмент, то можна скористатись інструкціями з встановлення [40].

Далі необхідно запустити сервер Appium та вказати параметри за замовчанням (рис.4.2). За замовчанням Appium використовує localhost з портом 4723.

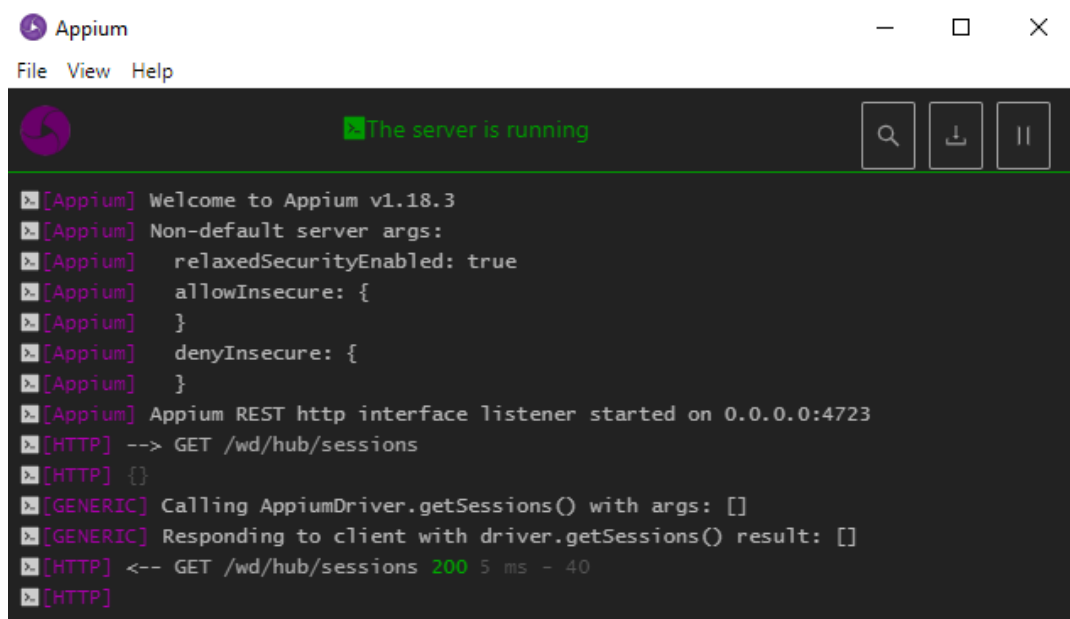


Рисунок 4.2 – Запуск серверу Appium

Автоматизований тест необхідно запустити або на емуляторі або на реальному пристрої. В цьому проєкті робота саме з Android емулятором тести необхідно проводити на багатьох пристроях, які мати реально фізично немає можливості. Для запуску емулятора UIAutomator необхідно встановити

Android Studio. Після чого відкриваємо та переходимо в Tools — AVD Manager та додаємо необхідний пристрій до списку та запускаємо його (рис. 4.3). Далі необхідно встановити застосунок, що тестується, на емуляторі мобільного пристрою. Для цього просто треба перетягти файл застосунку з розширенням *.apk на екран емулятора, після чого розпочнеться встановлення та іконка застосунку з'явиться в емуляторі пристрою..

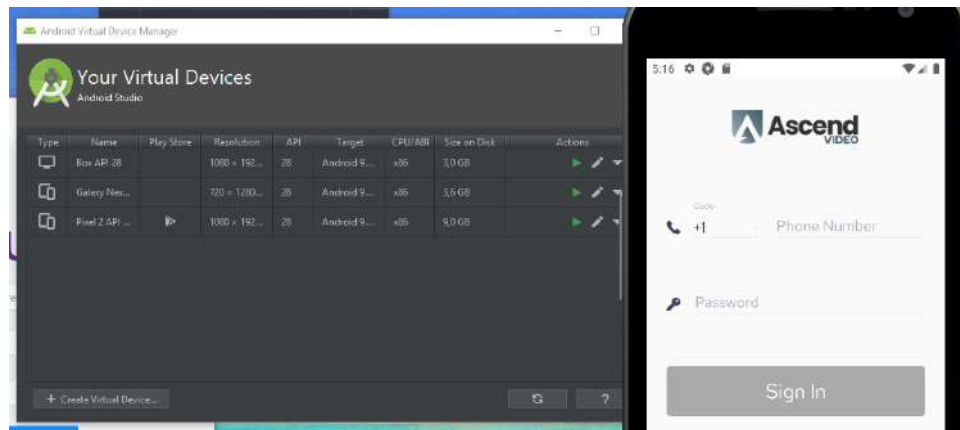


Рисунок 4.3 – Запуск тестової програми на емуляторі

Далі необхідно відкрити термінал і виконати команду `adb devices`. Вона покаже список підключених пристроїв, в якому повинен бути емулятор, якщо його немає в списку значить потрібно виконати пошук помилки.

Для того, щоб керувати елементами застосунку, необхідно звертатися до елементів через будь-які локатори. Для їх отримання необхідно відкрити Appium Inspector. Після чого всі capabilities визначені в попередньому розділі перенести в Inspector (рис. 4.4).

Після чого можна розпочати сесію Appium Inspector (рис.4.5). Інспектор транслює те, що відображається на пристрої або емуляторі. З права знаходиться опис всіх елементів поточного екрану.

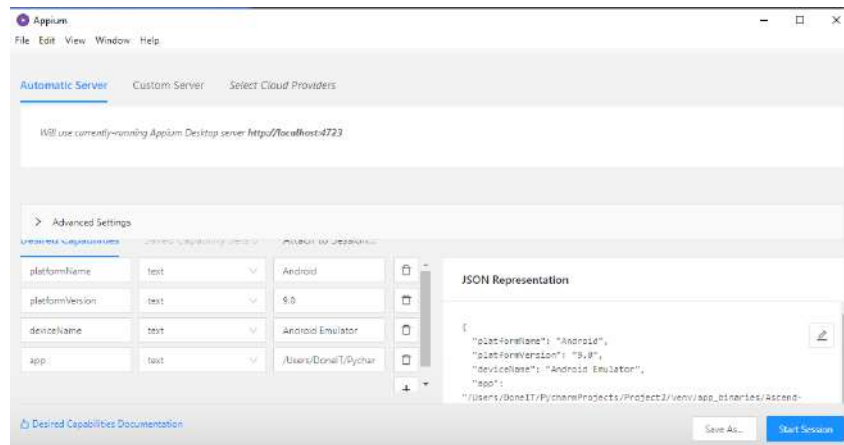


Рисунок 4.4 – Налаштування Appium Inspector для тестованого пристрою

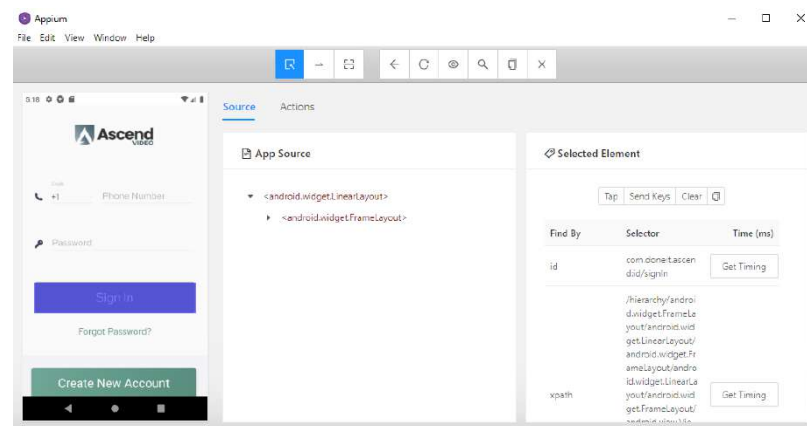


Рисунок 4.5 – Сесія тестованог мобільного застосунку в Appium Inspector

Наступним і останнім кроком є запуск тестів в середовищі розробки. Для цього запускаємо середовище PyCharm. Та відкриваємо в ньому проєкт під назвою Project (рис. 4.6). Обираємо необхідний тест та натискаємо Run.

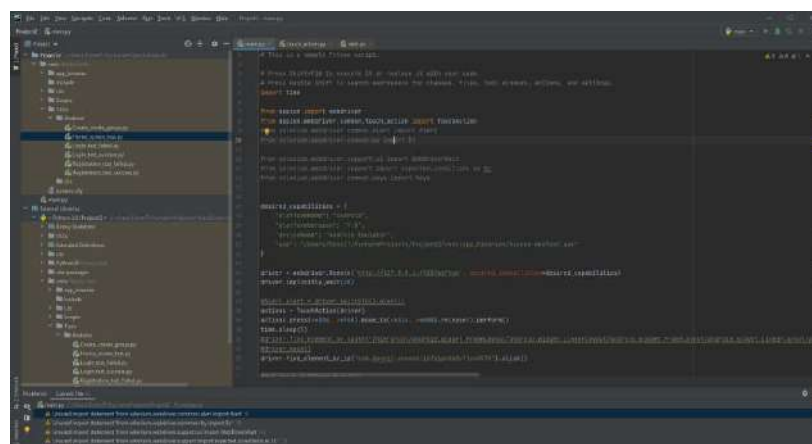


Рисунок 4.7 – Запуск автоматизованих тестів в середовищі PyCharm

4.5 Порівняння результатів виконання автоматизованих і ручних тестів

Для визначення ефективності впровадження автоматизованих тестів в процес тестування програмного подку було виконано порівняння тих самих тестів але у режимі ручного тестування.

Нижче наведені данні виконання експериментальних досліджень часових характеристик ручного та автоматизованого тестування:

- автоматизовані тести виконувались на комп'ютері з процесором Intel core i5-4430 CPU 3.00 GHz , ОЗУ: 8 Гб;
- кількість виконуємих тестів: до 100;
- середа розробки тестів: PyCharm.

100 тестів було виконано за 35 хвилин. Реальний час якщо б тести виконувала людина близько 8 годин (1 робочий день).

Загальні результати порівняння виконаного ручного та автоматизованого тестування наведено на рисунку 4.8.

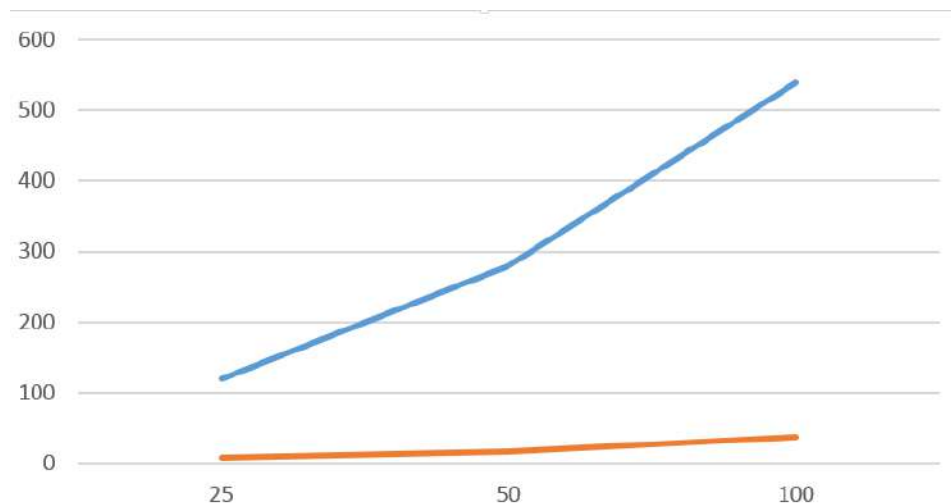


Рисунок 4.8 – Порівняння часу виконання автоматизованих тестів та тестів в ручному режимі

Як видно на рисунку, автоматизоване тестування вимагає менше часу для виконання тестів ніж це робити в ручну приблизно на 70-85%. Крім того,

чим більше тестів тим часовий відрив збільшується між ручним та автоматизованим тестуванням, оскільки у процесі тестування людина втомлюється та втрачає увагу, тож чим більше тестів необхідно провести тим більше з кожним тестом витрачається часу на зосередження.

4.6 Висновки до розділу 4

На основі розробленої методики був розроблено алгоритм налаштування запуску інструментарію та програмного забезпечення виконання автоматизованих тестів. Були описані кроки цього алгоритму. Приведені експериментальні дослідження, які порівнюють виконання одних тих самих тестів у ручному режимі та з використанням створених автоматизованих тестів.

За результатами проведеного експериментального тестування методики автоматизованого тестування мобільних застосунків, можна зробити висновок, що впровадження автоматизованих тестів в процес розробки дає змогу прискорити тестування програмного продукту на 70-85%. Отже, завдання розробки методики а також завдання створення автоматизованих тестів на основі методики, виконані.

5 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

5.1 Планування розробки програмного продукту

Програмне забезпечення, розроблене в межах магістерської роботи, призначене для автоматизації функціонального тестування мобільного застосунку. Мобільний застосунок – це великий проєкт, робота над яким запланована протягом року. Розробка даного програмного продукту є важливою, оскільки вона дозволяє скоротити час роботи тестувальника, який у випадку відсутності цієї програми повинен буде працювати над тестуванням даного мобільного застосунку весь свій час. В результаті впровадження програми тестувальник буде працювати над даним мобільним застосунком на 25 % свого часу менше. Зарплатня тестувальника складає 16 000 грн. на місяць. У розробці беруть участь інженер-програміст протягом 3 місяців і консультант протягом 1 місяця.

5.2 Визначення термінів розробки

Весь проєкт можна поділити на декілька етапів. Для кожного етапу визначається трудомісткість, тривалість робіт, кількість виконавців. У розробці беруть участь інженер-програміст протягом 3 місяців і консультант протягом 1 місяця. Розробка починається 7 вересня 2020 року і повинна закінчитися 7 грудня 2020 року. Розробка триває 65 робочих днів.

Тривалість робіт можна обчислити за формулою (5.1):

$$T_{\text{ц}} = \frac{Q}{R}, \quad (5.1)$$

де $T_{\text{ц}}$ – тривалість циклу, днів;

Q – трудомісткість, людино-днів;

R – кількість виконавців, чол.

Виконаємо обчислення:

$$T_{\text{ц}} = \frac{65}{1} + \frac{44}{2} = 87.$$

Результати характеристики робіт з розробки програми зведено в таблицю 5.1.

Таблиця 5.1 – Характеристика робіт з розробки програми

Найменування виду робіт	Трудомісткість		Виконавці		Тривалість, днів
	люд.- дні	вага до підсумку, %	спеціаль- ність	кількість люд.	
1	2	3	4	5	6
Дослідження предметної області	5	5,75	Програміст	1	5
Розробка специ- фікації проекту	2	2,3	Програміст	1	2
Розробка і затвердження технічного завдання	6	6,89	Консультант, програміст	2	3
Розробка основних алгоритмів	38	43,68	Консультант, програміст	2	19
Проектування модулів системи	5	5,75	Програміст	1	5

Продовження таблиці 5.1

1	2	3	4	5	6
Розробка модулів системи	5	5,75	Програміст	1	5
Тестування модулів системи	5	5,75	Програміст	1	5
Інтеграція модулів системи	5	5,75	Програміст	1	5
Розробка графічного інтерфейсу користувача	2	2,3	Програміст	1	2
Тестування системи	6	6,89	Програміст	1	6
Створення документації	8	9,19	Програміст	1	8
Всього:	87	100		2	65

5.3 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складається калькуляція кошторисної вартості робіт, яка включає такі статті:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок;
- витрати на спеціальне обладнання;
- матеріали і комплектуючі вироби;

- накладні витрати;
- податки.

5.4 Розрахунок основної заробітної плати

Витрати по основній заробітній платі складаються з фонду зарплати всіх категорій працівників, зайнятих у розробці програми. Розрахунок зарплати ведеться на підставі даних про трудомісткість, представлених в таблиці 5.2.

Таблиця 5.2 – Розрахунок основної заробітної плати

Посада виконавця	Чисельність, чол.	Місячний оклад, грн.	Кількість місяців	Сума заробітної плати, грн.
Програміст	1	14000	3	42000
Консультант	1	17000	1	17000
Підсумок	2			59000

5.5 Розрахунок додаткової заробітної плати

Додаткову заробітну плату обчислюють з розрахунку 10% від основної плати працівника та розраховують за формулою (5.2):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 0,1. \quad (5.2)$$

Виходячи з фонду основної заробітної плати, що дорівнює 59000 грн., обчислюємо фонд додаткової заробітної плати:

$$ЗП_{\text{дод}} = 59000 \cdot 0,1 = 5900.$$

5.6 Відрахування на єдиний соціальний внесок

Відрахування на єдиний соціальний внесок становлять 22% і беруться від основної та додаткової заробітної плати. Вони розраховуються за формулою (5.3):

$$OT = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22. \quad (5.3)$$

Виходячи з фонду основної заробітної плати, що дорівнює 59000 грн., та фонду додаткової заробітної плати, обчислюємо:

$$OT = (59000 + 5900) \cdot 0,22 = 14278 \text{ грн.}$$

5.7 Визначення витрат на матеріали

Використовується 2 найменування матеріалів: картридж для принтеру – 90 грн. і папір А4 – 110 грн. (1 упаковка). Витрати на матеріали розраховуються за формулою (5.4):

$$M = \sum_{I=1}^n (\Pi_i \cdot N_i \cdot (1 + K_{\text{т.з.}}) - \Pi_{i0} \cdot N_{i0}), \quad (5.4)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі, грн.;

$K_{\text{т.з.}}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

Π_i – ціна i -го найменування матеріалу, напівфабрикату і комплектуючого, грн.;

N_i – потреба в i -му матеріалі, напівфабрикатів і комплектуючих;

Π_{i0} – ціна зворотних відходів i -го найменування матеріалу, грн.;

N_{i0} – кількість зворотних відходів i -го найменування;

n – кількість найменувань матеріалів, напівфабрикатів та комплектуючих.

Виконаємо відповідні обчислення:

$$\Pi_{i0} = 0; N_{i0} = 0; K_{\text{т.з.}} = 0,05;$$

$$M = (1 + 0.05) (90 + 110) = 210,00 \text{ грн.}$$

5.8 Витрати на спеціальне обладнання

У цю статтю входять витрати на придбання, транспортування, монтаж і налагодження нестандартного обладнання.

Практично, в даному випадку, в цій статті, враховуються витрати на оплату машинного часу електронно-обчислювальної машини (ЕОМ) для написання і налагодження даної програми. Для чого необхідно скласти схему «витрат на утримання та експлуатацію устаткування» виходячи з якої визначити вартість одного машино - години роботи персонального комп'ютера (ПК), після множення якої на машинний час, який пішов на написання і налагодження програми, отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначаються за формулою (5.5):

$$A = \Phi_6 \cdot H_a, \quad (5.5)$$

де Φ_6 – балансова вартість обчислювальної техніки, грн.;

H_a – норма амортизаційних відрахувань на повне відновлення обчислювальної техніки, %.

Балансова вартість обчислювальної техніки становить 15000 грн. одержуємо:

$$A = 15000 \cdot 0,25 = 3750,00 \text{ грн.}$$

Статтю «Експлуатація обладнання» розраховують підсумовуванням витрат на електроенергію і допоміжні матеріали за формулою (5.6):

$$C_e = N_n \cdot \Phi_{\text{еф}} \cdot K_{\text{зв}} \cdot K_{\text{зм}} \cdot C_e, \quad (5.6)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\text{еф}}$ – річний ефективний фонд часу роботи ЕОМ, машино-годину;

$K_{\text{зв}}$ – Середній коефіцієнт завантаження за часом;

$K_{\text{зм}}$ – коефіцієнт завантаження по потужності;

C_e – ціна одного кВт * год електроенергії, грн. / (кВт * год).

Номінальна потужність ЕОМ – 0,4 кВт. Річний ефективний фонд часу роботи ЕОМ становить 1800 годин. Середні коефіцієнти завантаження за часом і за потужністю рівні відповідно 0,9 і 0,6. Ціна однієї кіловат-години електроенергії становить 2,11 грн. Отримаємо:

$$C_e = 0,4 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 820,37 \text{ грн.}$$

Зарплати обслуговуючого персоналу розраховуються за формулою (5.7):

$$ЗП_{\text{обс}} = ФЗП_p \cdot (1 + K_{\text{відр}}) \cdot \frac{t_{\text{обсл}}}{\Phi_{\text{еф.обс}}}, \quad (5.7)$$

де $ФЗП_p$ – річний фонд заробітної плати (основної й додаткової) обслуговуючих робітників, грн.;

$K_{\text{відр.}}$ – коефіцієнт, що враховує відрахування на єдиний соціальний внесок;

$t_{\text{обсл.}}$ – час протягом року, необхідний на технічне обслуговування ЕОМ, год/рік;

$\Phi_{\text{еф.обсл.}}$ – річний ефективний фонд часу обслуговуючого персоналу, год/рік.

Місячна основна заробітна плата обслуговуючого персоналу становить 3723,00 грн., а річний фонд основної заробітної плати відповідно дорівнює 44676,00 грн.

Річний ефективний фонд робочого часу обслуговуючого персональний комп'ютер працівника дорівнює 1750 годин / рік. На обслуговування одного персонального комп'ютера по 1 годині на місяць, що на рік становить 12 годин. Отримаємо:

$$ЗП_{\text{обсл}} = 44676 \cdot (1 + 0,22) \cdot \frac{12}{1750} = 373,74 \text{ грн.}$$

Стаття «Поточний ремонт обладнання» приймається рівною 3% від балансової вартості обладнання і складає 450,00 грн.

Стаття «Інші витрати» приймається рівною п'яти відсоткам від суми всіх попередніх статей витрат на утримання та експлуатацію устаткування. Сума всіх попередніх статей дорівнює 5394,11 грн., 5% від даної суми становлять 269,70 грн.

Розраховані статті витрат на утримання та експлуатацію устаткування внесені в таблиці 5.3.

Таблиця 5.3 – Кошторис витрат на утримання та експлуатацію обладнання

Найменування статей витрат	Сума, грн
1	2
Амортизація обладнання	3750,00
Експлуатація обладнання (крім витрат на поточний ремонт)	820.37

Продовження таблиці 5.3

1	2
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	373,74
Поточний ремонт обладнання	450,00
Інші витрати	269,70
Підсумок	663,81

Витрати на оплату машинного часу ЕОМ для написання і налагодження даної програми визначаються за формулою (5.8):

$$C_{\text{мо}} = P_{\text{екс}} \cdot t_{\text{мо}}, \quad (5.8)$$

де $C_{\text{мо}}$ – витрати на оплату машинного часу, грн.;

$P_{\text{екс}}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино-год.;

$t_{\text{мо}}$ – машинний час цифрової ЕОМ для написання і налагодження даного програмного продукту, машино-год.

Експлуатаційні витрати на одну годину машинного часу використовуваної ЕОМ розраховують діленням суми витрат по кошторису «Витрати на утримання та експлуатацію устаткування (ЕОМ)» на річний ефективний фонд часу роботи ЕОМ. Річний ефективний фонд часу роботи ЕОМ дорівнює 1800 годин. У результаті експлуатаційні витрати на одну годину машинного часу рівні:

$$P_{\text{екс}} = \frac{5663,81}{1800} = 3,14 \text{ грн./машино – год.}$$

ЕОМ експлуатується 65 робочих днів в одну зміну (8 годин), що становить в сумі 520 годин. Таким чином, витрати на оплату машинного часу складуть:

$$C_{\text{мо}} = 3,14 \cdot 520 = 1632,8 \text{ грн.}$$

5.9 Розрахунок накладних витрат

Результати визначення витрат на розробку програми у вигляді калькуляції кошторисної вартості робіт наведено в таблиці 5.4. У таблиці розрахована підсумкова сума вартості робіт, а так само питома вага кожної статті витрат у відсотках від підсумкової суми.

Таблиця 5.4 – Калькуляція кошторисної вартості робіт з розробки програми

№	Найменування статей витрат	Сума, грн	Питома вага до підсумку, %
1	Основна заробітна плата	59000,00	67,88
2	Додаткова заробітна плата	5900,00	6,79
3	Відрахування на єдиний соціальний внесок та інше	14278,00	16,42
4	Матеріали та комплектуючі	210,00	0,24
5	Витрати на спец. обладнання	1632,80	1,88
6	Накладні витрати	5900,00	6,79
7	Підсумок	86920,80	100,00

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати тощо), утримання та експлуатацію будівель. Накладні витрати включаються у вартість розробки програми непрямым шляхом – у

відсотках до основної заробітної плати розробників. У даному випадку накладні витрати становлять 10 % до основної заробітної плати розробників, це складає 5900 грн.

5.10 Розрахунок економічної ефективності

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою (5.9):

$$E = (T_{M_1} - T_{M_2}) \cdot B_{\text{екс.КТЗ}} + E_n^a \cdot \frac{\Phi_{\text{б.КТЗ}} \cdot (T_{M_1} - T_{M_2})}{\Phi_{\text{еф.КТЗ}}} - \frac{S_{\text{рп}}}{T_c} + E_{\text{зп}}, \quad (5.9)$$

де T_{M_1} , T_{M_2} – машинний час, необхідний для рішення поставлених завдань, відповідно в старому та у новому варіантах, машино-год./рік;

$B_{\text{екс.КТЗ}}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино-год.;

E_n^a – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{\text{б.КТЗ}}$ – балансова вартість ЕОМ, грн.;

$\Phi_{\text{еф.КТЗ}}$ – річний ефективний фонд часу роботи ЕОМ, машино-год.;

$S_{\text{рп}}$ – сумарні витрати на розробку програми, грн.;

T_c – термін служби впроваджуваної програми до її морального зношування, років;

$E_{\text{зп}}$ – економія на заробітній платі (до впровадження даного програмного забезпечення дані функції виконувались тестувальником, який працював 1800 годин на рік (з оплатою за місяць 16 000 гривень), впровадження програми дозволить скоротити 25 % часу його роботи).

Виконаємо відповідні обчислення:

$$E = (1800 - 1350) \cdot 3,14 + 0,15 \cdot (15000 \cdot (1800 - 1350)) / 1800 - 86920,80/5 + 16000 \cdot (1 + 0,22) \cdot 1,1 \cdot 3 = 49007,34 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E і строк їхньої окупності $T_{\text{ок}}$ розраховуються за формулами (5.10) і (5.11):

$$E_{\phi} = \frac{E}{S_{\text{рп}}}, \quad (5.10)$$

$$T_{\text{ок}} = \frac{S_{\text{рп}}}{E}. \quad (5.11)$$

Виконаємо відповідні обчислення:

$$E_{\phi} = 49007,34 / 86920,80 = 0,56,$$

$$T_{\text{ок}} = 86920,80 / 49007,34 = 1,77 \text{ року.}$$

Зведемо отримані техніко-економічні показники в таблицю 5.5.

Таблиця 5.5 – Техніко-економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	86920,80	грн.
Річна економія	49007,34	грн.
Строк окупності програми	1,77	рік

Строк окупності програми 1,77 року, що значно менше нормативного строку окупності для соціальної сфери (6,67 років). Річна економія при використанні програми становить 49007,34 гривні. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

6 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

В дипломній кваліфікаційній роботі магістра проводиться дослідження та реалізація програмного забезпечення для автоматизації функціонального тестування мобільного застосунку.

При роботі передбачається використання ЕОМ із спеціалізованим програмним забезпеченням. Тому необхідно виконати вимоги щодо охорони праці для збереження здоров'я робітників в обчислювальному центрі.

6.1 Аналіз потенційних небезпек

При роботі з комп'ютером, на людину впливає комплекс негативних факторів обумовлених характером виробничого процесу. До таких потенційно небезпечних факторів можна віднести:

- ураження електричним струмом при контакті зі струмопровідними частинами, які мають порушену ізоляцію або відсутнє заземлення (при ввімкненому/вимкненому електроприладі або освітленні в приміщенні);
- отримання фізичних травм через недбале поводження з канцелярськими приладами або їх використання не за прямим призначенням;
- зниження працездатності та погіршення загального здоров'я працівника через порушення мікроклімату в приміщенні;
- порушення гостроти зору при недостатньому рівні освітлення робочого місця, зорова втома при тривалій роботі з комп'ютером і документами;
- зниження імунітету організму людини, яка тривалий час (більше 4 годин на добу) працює за комп'ютером через вплив електромагнітних випромінювань при роботі з комп'ютером;
- зниження працездатності та погіршення загального самопочуття через перевтому, викликану понаднормовими навантаженнями на індивіда фактичної тривалості робочого часу та інтенсивності протікання робіт;

- виникнення пожежі внаслідок короткого замикання або порушення правил техніки електробезпеки працівником на робочому місці;
- негативні наслідки через неадекватні дії працівників в умовах надзвичайної ситуації.

Таким чином були визначені фактори, які несуть потенційну небезпеку та негативно впливають на здоров'я працівників.

6.2 Заходи із забезпечення безпеки

Оскільки основним потенційно небезпечним фактором є ураження електричним струмом, то були виділені основні причини можливого ураження електричним струмом:

- ураження електричним струмом при несправності обладнання, що не залежать від людини та брак захисних засобів;
- ураження електричним струмом при необережному поводженні або недбалому користуванні електричним обладнанням з боку працівника.

Електробезпека в приміщенні забезпечується:

- конструкцією установок;
- технічними засобами та способами захисту;
- організаційними та технічними заходами [41].

Для захисту від ураження електричним струмом, в тому числі й випадків недбалого ставлення обслуговуючого персоналу, передбачено виконання конструктивних заходів:

- належне ізолювання струмопровідних частин обладнання, яке є джерелом небезпеки, а також прокладення останнього в місця з обмеженим доступом для людей (для електроприладів першого класу передбачено використання подвійної ізоляції IP44, яка передбачає захист від шару пилу до 1 мм і бризок);
- заземлення металевих частин обладнання, яке у випадку пошкодження ізоляції може опинитися під небезпечним рівнем напруги;

– передбачити пристрій, який централізовано вимикає усі електричні схеми від мережевого живлення [42].

Технічні способи та засоби захисту, які забезпечують електробезпеку, встановлюються в залежності від умов оточуючого середовища. Приміщенню, в якому встановлено персональні комп'ютери, з урахуванням умов оточуючого середовища присвоєно ступінь небезпеки та ураження електричним струмом категорії приміщення без підвищеної небезпеки [41].

Виключно важливе значення, для попередження виникнення електротравм, має правильна організація обслуговування діючих електроустановок, проведення ремонтних, монтажних і профілактичних робіт. Під правильною організацією розуміють суворе виконання ряду організаційних і технічних заходів, встановлених чинними вимогами [41], [43]. Рекомендовано встановлення автоматичних вимикачів та пристроїв заземлення контуру.

Для забезпечення безпечних умов роботи з діючими електроустановками необхідно виконати наступні організаційні заходи регламентовані відповідно до:

- призначення осіб, відповідальних за організацію та виконання робіт;
- оформлення робіт із використанням нарядів – допусків або усним розпорядженням, які виконуються в порядку поточної експлуатації;
- видача допусків на виконання робіт після перевірки підготовки робочого місця;
- нагляд за виконанням робіт виконується особою, яка працює на даному робочому місці;
- оформлення перерв в роботі, переводів на інші робочі місця та закінчення робіт [43].

Для забезпечення електробезпеки, згідно з ПУЕ, роботи пов'язані з виконанням наряду – допуску або усного розпорядження, при підготовці робочого місця необхідно виконати наступні технічні заходи:

– виконати відключення та вжити заходів щодо перешкоджання подачі напруги на робоче місце, внаслідок помилкового або випадкового ввімкнення комутаційних апаратів;

– на приводах з ручним керуванням, і на ключах із дистанційним керуванням комутаційних апаратів мають бути розміщені знаки попереджувального характеру;

– необхідно перевірити наявність напруги в струмопровідних частинах, які мають бути заземлені для захисту працівників від ураження електричним струмом;

– використати заземлюючі ножі, а в тих місцях де вони відсутні, мають бути переносні заземлювачі;

– розмістити вказівники та плакати «Заземлено», які носять попереджувальний та рекомендаційний характер, а при необхідності відгородити робоче місце та струмопровідні частини, які залишилися під напругою.

Для уникнення уражень електричним струмом, серед працівників та обслуговуючого персоналу, необхідно виконати розрахунок системи захисного заземлення, яке використовується в електромережах з напругою до 1000 В [43]. Згідно з завданням, вихідні дані для розрахунку захисного заземлення приведені в таблиці 6.1.

Визначаємо необхідний опір штучного заземлювача R_u , Ом, якщо передбачається використання також природного заземлювача R_e , за формулою (6.1), Ом:

$$R_u = \frac{R_e R_3}{R_e - R_3}, \quad (6.1)$$

де R_e – опір розтіканню струму природних заземлювачів, Ом;

R_3 – розрахунковий нормований опір заземлювального пристрою (ЗП), Ом.

Таблиця 6.1 – Вихідні дані для розрахунку

Назва параметру	Позначення, од.	Значення
Напруга в мережі	U , В	380
Розміри приміщення	L , В, м	13·12
Опір природного заземлювача	$R_{\text{с}}$, Ом	18
Довжина вертикального стрижня	$l_{\text{в}}$, м	5
Діаметр вертикального стрижня	d , м	0,018
Ширина смуги	$b_{\text{с}}$, м	0,045
Питомий опір ґрунту	$\rho_{\text{вим}}$, Ом · м	120
Коефіцієнт сезонності ґрунту	Ψ	1,3
Розрахунковий опір заземлення, при $U < 1000 \text{ В}$	$R_{\text{з}}$, Ом	4

Виконаємо відповідні розрахунки:

$$R_{\text{з}} = \frac{18 \cdot 4}{18 - 4} = 5,1 \text{ Ом.}$$

Визначимо розрахунковий питомий опір ґрунту за формулою (6.2):

$$\rho = \rho_{\text{вим}} \cdot \Psi, \text{ Ом} \cdot \text{м}, \quad (6.2)$$

де ρ – розрахунковий питомий опір ґрунту, Ом·м;

$\rho_{\text{вим}}$ – питомий опір землі, отриманий у результаті вимірів, Ом·м; $\rho_{\text{вим}} = 120 \text{ Ом} \cdot \text{м}$;

Ψ – коефіцієнт сезонності, що враховує промерзання чи висихання ґрунту;
 $\Psi = 1,3$.

Розрахуємо:

$$\rho = 120 \cdot 1,3 = 156 \text{ Ом} \cdot \text{м.}$$

Обчислимо опір розтіканню струму одиночного вертикального заземлювача R_B , Ом. Для стрижневого заземлювача круглого перерізу, заглибленого в ґрунт, розрахункова формула (6.3) має вигляд:

$$R_g = \frac{\rho}{2 \cdot \pi \cdot l_B} \left(\ln \frac{2 \cdot l_B}{d} + \frac{1}{2} \cdot \ln \frac{4 \cdot t_B + l_B}{5 \cdot t_B - l_B} \right), \quad (6.3)$$

де ρ – розрахунковий питомий опір ґрунту, Ом·м; $\rho = 156$ Ом·м;

l_B – довжина вертикального стрижня, м; $l_B = 5$ м;

d – діаметр перерізу стрижня, м; $d = 0,018$ м;

t_B – відстань від поверхні ґрунту до середини довжини вертикального стрижня, яка обчислюється за формулою (6.4), м:

$$t_B = 0,8 + \frac{1}{2} l_B, \quad (6.4)$$

Підставимо значення та розрахуємо:

$$t_B = 0,8 + \frac{1}{2} l_B = 0,8 + \frac{1}{2} \cdot 5 = 0,8 + 2,5 = 3,3 \text{ м.}$$

Тоді розрахуємо R_B :

$$R_g = \frac{156}{2 \cdot 3,14 \cdot 5} \left(\ln \frac{2 \cdot 5}{0,018} + \frac{1}{2} \cdot \ln \frac{4 \cdot 3,3 + 5}{5 \cdot 3,3 - 5} \right) = 32,52 \text{ Ом.}$$

Розрахуємо наближену (мінімальну) кількість вертикальних стрижнів за формулою (6.5), шт.:

$$n' = \frac{R_g}{R_u}, \quad (6.5)$$

де R_b – опір розтікання струму одиночного вертикального заземлювача, Ом;

$$R_g = 32,52 \text{ Ом};$$

R_u – необхідний опір штучного заземлювача, Ом; $R_u = 5,1 \text{ Ом}$.

Розрахуємо:

$$n' = \frac{32,52}{5,1} = 6,38 \text{ шт.}$$

Якщо виходити з розмірів контуру (6.6), шт.:

$$n = \frac{P}{a}, \quad (6.6)$$

де P – периметр прямокутника (приміщення), який визначається за формулою (6.7), м:

$$P = 2 \cdot (L + B), \quad (6.7)$$

де L – довжина приміщення, м;

B – ширина приміщення, м.

Розрахуємо:

$$P = 2 \cdot (13 + 12) = 50 \text{ м.}$$

a – відстань між стрижнями, м; обумовлена зі співвідношення та обчислюється за формулою (6.8):

$$a = k \cdot l_b, \quad (6.8)$$

де l_b – довжина вертикального стрижня, м;

k – коефіцієнт кратності, який дорівнює 1, 2, 3 (для заглиблених стаціонарних заземлювачів $k=1$).

Розрахуємо:

$$a = 1 \cdot 5 = 5 \text{ м.}$$

Тоді:

$$n = \frac{50}{5} = 10 \text{ шт.}$$

Приймаємо $n = 10$ шт.

Отриману кількість стрижнів округляють до більшого довідкового значення, але не менше n' .

Довжина горизонтальної смуги при конфігурації групового заземлювача – контуру, що обчислюється за формулою (6.9):

$$l_z = 1,05 \cdot a \cdot n, \quad (6.9)$$

де n – кількість вертикальних стрижнів; $n = 10$ шт.

a – відстань між вертикальними стрижнями, м; $a=5$ м.

Розрахуємо:

$$l_z = 1,05 \cdot 10 \cdot 5 = 52,5 \text{ м.}$$

Обчислити опір розтіканню струму горизонтальної з'єднуючої смуги R_e , Ом. У випадку горизонтального смугового заземлювача розрахунок виконується за формулою (6.10), Ом:

$$R_r = \frac{\rho}{2 \cdot \pi \cdot l_r} \cdot \ln \frac{2 \cdot l_r^2}{b_c \cdot t_r}, \quad (6.10)$$

де ρ – розрахунковий питомий опір ґрунту, Ом·м; $\rho = 156$ Ом·м;

l_r – довжина горизонтальної смуги, м; $l_r = 52,5$ м;

b_c – ширина смуги, м; $b_c = 0,045$ м;

t_r – відстань від поверхні ґрунту до середини ширини горизонтальної смуги, яка обчислюється за формулою (6.11):

$$t_r = 0,8 + \frac{1}{2} b_c. \quad (6.11)$$

Розрахуємо:

$$t_r = 0,8 + \frac{1}{2} \cdot 0,045 = 0,82 \text{ м.}$$

Тоді розрахуємо опір розтіканню струму горизонтальної з'єднуючої смуги R_r , Ом.:

$$R_r = \frac{156}{2 \cdot 3,14 \cdot 52,5} \cdot \ln \frac{2 \cdot 52,5^2}{0,045 \cdot 0,82} = 5,63 \text{ Ом.}$$

Розрахувати еквівалентний опір розтіканню струму групового заземлювача за формулою (6.12):

$$R_{rp} = \frac{R_b \cdot R_r}{R_b \cdot \eta_r + R_r \cdot \eta_b \cdot n}, \text{ Ом,} \quad (6.12)$$

де R_b – опір розтікання струму одиночного вертикального заземлювача, Ом;

$R_g = 32,52$ Ом;

R_r – опір розтікання струму горизонтальної смуги, Ом; $R_z = 5,63$ Ом;

η_b, η_r – коефіцієнти використання вертикальних стрижнів і горизонтальної смуги, Ом; $\eta_b = 0,57$; $\eta_r = 0,34$;

n – кількість вертикальних стрижнів; $n = 10$ шт.

$$R_{rp} = \frac{32,52 \cdot 5,63}{32,52 \cdot 0,34 + 5,63 \cdot 0,57 \cdot 10} = 4,24 \text{ Ом.}$$

Отриманий опір розтіканню струму групового заземлювача не повинен перевищувати необхідний опір, визначений вище за формулою (6.1):

$$R_{rp} \leq R_u.$$

Після підстановки отримаємо:

$$4,24 \text{ Ом} \leq 5,10 \text{ Ом.}$$

Умову виконано, рішення завдання вірне.

Таким чином, були розглянуті потенційні небезпеки ураження електричним струмом, при роботі з електроприладами та запропоновано заходи щодо забезпечення техніки електробезпеки при роботі з електроприладами.

6.3 Заходи з виробничої санітарії та гігієни праці

При організації праці пов'язаної з ВДТ ЕОМ і ПЕОМ, для збереження здоров'я працівників, попередження професійних захворювань та підтримки працездатності необхідно дотримуватися вимог санітарії та гігієни регламентованих [44]. Деякі з них приведено в таблиці 6.2.

Таблиця 6.2 – Санітарно – гігієнічні вимоги в приміщеннях з ЕОМ

Критерій оцінки	Значення
Площа робочого місця для одного користувача ЕОМ	$S_{pm} \geq 6 \text{ м}^2$
Об'єм робочого місця для одного користувача ЕОМ	$V_{pm} \geq 20 \text{ м}^3$
Температура повітря в приміщенні (холодно)	22°C – 24°C
Температура повітря в приміщенні (тепло)	23°C – 25°C
Відносна вологість повітря (холодно / тепло)	40 – 60 %
Швидкість руху повітря	$v_b \leq 0,1 \text{ м/с}$
Рівень шуму	$L_{ш} \leq 60 \text{ дБА}$
Рівень освітленості робочого місця	300 – 500 лк
Рівень електромагнітного випромінювання	$L_{EB} \geq 25 \text{ В/м}$

Для збереження здоров'я працівників дотримуються режиму праці та відпочинку регламентованого [44]. Згідно з ним, протягом робочого дня необхідно робити перерви, при появі ознак втоми та зниження працездатності.

При виконанні робіт протягом робочого дня, працівники усіх категорій трудової діяльності залежно від основної роботи, пов'язаної з ВДТ ЕОМ і ПЕОМ (складає не менше 50% часу робочого дня), повинні дотримуватися наступних правил праці та відпочинку:

- тривалість безперервної роботи за комп'ютером не повинна перевищувати 2 години;
- тривалість перерви для інженера – програміста має складати не менше 15 хвилин кожні 2 години робочого часу.

Також, для зниження нервово – емоційного навантаження, втоми зорового апарату та попередження втоми кістково – м'язового апарату, внаслідок прошення режиму праці та відпочинку, необхідно виконувати фізичні вправи.

Для забезпечення виробничої санітарії та гігієни праці, необхідно виконати ряд організаційних та технічних заходів відповідно до [44].

Організаційні заходи забезпечення санітарних норм та правил передбачають:

- спостереження за дотриманням вимог охорони стосовно праці жінок та неповнолітніх осіб;
- організацію проходження попереднього та періодичного медичного огляду кожним працівником, який працює з ВДТ ЕОМ і ПОЕМ;
- організацію заходів щодо поліпшення санітарних умов праці на виробництві.

Для забезпечення виробничої санітарії та гігієни праці, передбачено виконання деяких технічних (організаційно – технічних) заходів.

Монотонність роботи може викликати захворювання невротичного характеру, порушення кістково – м'язового апарату в результаті тривалих статичних навантажень та порушення в роботі зап'ястка. Для попередження можливих проблем зі здоров'ям у працівників необхідно виконати заходи організаційно – технічного характеру:

- дотримуватися режиму праці та відпочинку;
- змінювати тип робіт пов'язаних із роботою на ПЕОМ.

Окрім того, для покращення ергономічності робочого місця необхідно виконати організаційно – технічні заходи щодо правильного облаштування робочих місць, регламентованого [44] :

- ширина і глибина робочого простору складає не менше 700 і 400 мм, а висота робочої поверхні не перевищує 700 – 750 мм;
- оптимальними параметрами для столу є довжина і ширина складають 1300 і 650 мм (для письмових робіт менше 40 і 600 мм);
- повинний бути передбачений простір для ніг під робочою поверхнею висота, ширина та глибина яких не менше 600, 500 і 400 мм;
- робоче крісло має задовольняти регламентованим вимогам і мати такі характеристики: регульований допуск висоти залежно від зросту складає

400 – 550 мм, має невелику опуклість поверхні та невеликий вигин назад [42]

Для забезпечення оптимальних показників мікроклімату наведених в таблиці 2, необхідно дотримуватися вимог регламентованих [45]. Застосовують технічні заходи до таких показників мікроклімату: вентиляція, кондиціонування повітря та температура оточуючого середовища.

Для систем вентиляції та кондиціонування повітря передбачені вимоги:

- місце встановлення має бути таким, щоб потік повітря не потрапляв на працівників;

- різниця температур повітря на поверхні робочого місця та рівні голови не повинна перевищувати 5°C;

- температура повітря в приміщенні підтримується на рівні 22°C – 24°C (холодно) і 23°C – 25°C (тепло) шляхом встановлення водної опалювальної системи;

- швидкість обміну повітря не повинна перевищувати 0,1 м/с.

Рівень шуму в приміщенні складає 65 дБА внаслідок роботи комп'ютерів, периферійної техніки та розмов самих працівників. Такий рівень шуму призводить до зниження ефективності праці, підвищеної втоми та погіршення психологічного стану працівників. Для зниження рівня шуму до регламентованого [46] необхідно виконати організаційно – технічні заходи:

- необхідно встановити периферійну оргтехніку в окремих приміщеннях або використовувати шумоізоляційні екрани для робочих місць;

- виконати пасивну звукоізоляцію шляхом облицювання огорожуваних конструкцій матеріалами з високим рівнем поглинання звуку, активну – шляхом використання глушників шуму або їх комбінацію.

Для забезпечення оптимального рівня освітлення робочого місця передбачається:

- встановлення світильників місцевого освітлення;

– використання системи рівномірного освітлення з використанням люмінесцентних ламп ЛБ [47], освітленість повинна становити 300 лк

Таким чином, був розроблений комплекс організаційних і технічних заходів для забезпечення санітарно – гігієнічних вимог регламентованих при роботі з ВДТ ЕОМ і ПЕОМ, забезпечує зниження ймовірності виникнення у працівників професійних та хронічних захворювань пов'язаних з працею.

6.4 Заходи безпеки у надзвичайних ситуаціях

6.4.1 Заходи з пожежної безпеки

Заходи з протипожежної безпеки розроблені для виробничого приміщення обладнаного ПК з ВДТ розроблені згідно [48].

Виходячи з аналізу речовин та матеріалів, які використовуються при роботі у приміщенні, обладнаному ПК з ВДТ:

- згідно [49] у приміщенні, обладнаному ПК з ВДТ можлива пожежа класу – А та Е;

- згідно [50] воно належить до категорії «Д» з пожежної безпеки – простір у приміщенні, у якому перебувають тверді горючі речовини та матеріали.

Оскільки приміщення належить до категорії «Д» з пожежної безпеки, тому відповідно до [49] воно має II ступінь вогнестійкості.

У разі виникнення пожежі у приміщенні обладнаному ПК з ВДТ для евакуації персоналу відповідно до вимог [49] передбачені виходи, по обидві сторони приміщення, з одного боку вікно (на пожежні сходи), а з іншого – вхідні двері. Згідно існуючих норм, відстань від найбільш віддаленого робочого місця до найближчого евакуаційного виходу не обмежується.

Обладнання, силові та освітленні мережі приміщення, обладнаного ПК з ВДТ відповідають вимогам пожежної безпеки, оскільки виконані відповідно до вимог [51], та мають ступінь захисту ізоляції обладнання IP44

яка відповідає класу пожежанебезпечної зони П-Па до якої належить приміщення.

Для виявлення початкової стадії загоряння, швидкого й точного оповіщення служби пожежної охорони про час і місце виникнення пожежі встановлені системи автоматичної пожежної сигналізації (АПС). Вони можуть самостійно пускати в хід установки пожежогасіння, коли пожежа не досягла великих розмірів. Системи АПС складаються з пожежних сповіщувачів, ліній зв'язку й прийомних пультав (станцій). Згідно вимог [42] в приміщенні обладнаному ПК з ВДТ встановлена система пожежної й охоронної сигналізації «Фотон-ОФІС», яка забезпечує виявлення теплових і димових ознак пожежі і місця виникнення пожежі з точністю до місця розміщення датчика.

6.4.2 Заходи з цивільного захисту

Непідготовленість та неадекватні дії персоналу можуть призвести до значних втрат у разі виникнення надзвичайної ситуації. Для зниження дії несприятливих факторів і наслідків, викликаних НС, використовують захисні укриття. Захисні укриття, як один із напрямів інженерного захисту населення та територій, які призначені для укриття населення під час розвитку небезпечних процесів та явищ. Згідно з призначенням, визначають наступні види захисних споруд: фортифікаційні, військові та цивільної оборони.

З метою захисту цивільного населення, техніки та матеріальних цінностей від сучасних засобів ураження, небезпечних природних і техногенних явищ було розроблено різні по конструкції, захисним властивостям і термінам будівництва захисні споруди цивільного захисту [52].

Захисні споруди, для як найшвидшого прийому постраждалих приводяться в режим готовності протягом 12 годин, на атомних станціях і хімічно – небезпечних об'єктах сховища знаходяться в постійній готовності.

Визначають основні види захисних споруд: найпростіші, протирадіаційні та сховища.

Сховища забезпечують захист населення від прогнозованого впливу вражаючих факторів, при застосуванні як звичайних засобів (без урахування прямого потрапляння), бактеріальних засобів, отруйних речовин, так і ядерної зброї. Окрім захисту від перелічених вражаючих факторів, які наведено вище, сховища забезпечують захист від природних впливів таких як катастрофічне затоплення, аварії із застосування хімічно небезпечних речовин, радіоактивних речовин внаслідок руйнування енергетичних установок, високих температур і продуктів горіння внаслідок виникнення пожеж. Системи життєзабезпечення сховища забезпечують безперервний захист протягом 48 годин. Постачання повітря до сховищ здійснюється в два режими: чистої вентиляції та фільтровентиляції.

Постачання повітря до сховищ, які дислоковані поблизу атомних станцій та хімічно – небезпечних об'єктів, районах можливого затоплення та пожеж, окрім наведених режимів, включає також два додаткових режими: повної ізоляції і часткової ізоляції. Сховища, як захисні споруди, можна класифікувати по такими критеріями: захисні властивості, кількість місць або місткість, місце розташування, час побудови, матеріал конструкцій, спосіб енергопостачання, забезпечення фільтровентиляційним обладнанням, характер використання в мирний час. За місткістю, сховища бувають: малі (до 150 чоловік), середні (150-600 чоловік), великі (600-5000 чоловік і більше). При збільшенні місткості сховища, вартість його будівництва на душу населення знижується.

За місцем розташування сховища бувають: окремі, споруджені зовні, споруди заглиблені повністю/частково в землю, вбудовані, розміщені в підвалах і на перших поверхах будівель та споруд, в гірничих виробках і природних порожнинах, в підземних спорудах міського будівництва.

За часом побудови, сховища бувають: побудовані в мирний час і побудовані в умовах НС.

Сховища відрізняються за матеріалом побудови: лісоматеріали, комплексні, кам'яні/блокові, тканинні та тканекаркасні, металеві та залізобетонні. В свою чергу сховища побудовані із залізобетону, можна класифікувати, як монолітні залізобетонні, збірні та змішані притулки.

Протирадіаційні укриття призначені для захисту населення від впливу іонізуючого випромінювання внаслідок радіоактивного забруднення місцевості. Системи життєзабезпечення забезпечують безперервний захист встановленої кількості осіб до 48 годин. Протирадіаційні укриття можна класифікувати за наступними властивостями: захисні властивості, місткість укриття, фонд приміщень і шляхи постачання повітря.

Найпростішим укриттям є споруда, яка забезпечує лише частковий захист населення від ударної хвилі у повітрі, світлового випромінювання та уламків зруйнованих будівель. Найпростіше укриття може знизити вплив проникаючої радіації та іонізуючих випромінювань на радіоактивно забрудненій місцевості, захистити від негоди та інших несприятливих умов.

До найпростіших укриттів можна віднести: відкриті/перекриті щілини, траншеї, підвали/підпілля з лісоматеріалів, землянки, навіси, цокольні/перші поверхи будівель і заглиблені приміщення.

На облаштування щілин та траншей відводиться до 12 годин. Також існує варіант перетворення останніх у протирадіаційне укриття до 48 годин і у притулок. Місткість найпростішого укриття складає від 10 до 40 чоловік. Усі роботи по будівництву укриттів тісно пов'язані з планами будівництва та евакуації населення.

Нарощування фонду захисних споруд проходить в двох напрямках: комплексне освоєння підземного простору міст: пристосування підвальних приміщень у знову споруджуваних і існуючих будівлях, споруд заглиблених у землю, метрополітенів, підземних гірничих виробок, печер та інших підземних порожнин і пристосування приміщень на цокольних і наземних поверххах в існуючих і знову споруджених будинках [52].

ВИСНОВКИ

В кваліфікаційні дипломній роботі магістра було проведено дослідження методів та видів тестування та розроблена класифікація видів тестування. Було розглянуто особливості тестування застосунків на мобільних пристроях.

Проаналізовано ручне та автоматизоване тестування, та особливості застосування цих способів в процесі тестування. Крім того, проаналізовано проблеми автоматизованого тестування мобільних застосунків. Однак, хоча автоматизоване тестування і має недоліки, пов'язанні з часовими, ресурсними витратами на їх створення та виконання, але з точки зору швидкості виконання тестів, продуктивності, точності та можливості повторного використання розроблених тестів, автоматизоване тестування надає суттєву перевагу порівняно з ручним тестуванням.

Дослідження показало, що на сьогодні неможливо повністю відмовитись від ручного тестування але і виникає велика необхідність застосовувати методи автоматизованого тестування в процесі розробки. Найбільш дієвим аспектом тестування, що забезпечує вищий щабель автоматизації є функціональне тестування. Тестування за функціональними критеріями, передбачає здійснення процесу тестування за методом «чорної скриньки».

Як показало дослідження інструменти для автоматизації тестів можна поділити на: інструментальні засоби для взаємодії з мобільним застосунком (драйвери, надстройки); засоби для написання та виконання тестів (фреймворки).

Для розробки автоматизованих тестів та прискорення їх тестування серед проаналізованих було обрано наступні засоби. Мова програмування для написання тестів було обрано Python через свою популярність та інтернет спілук. Тестовий фреймворк для написання тестів на мові Python було обрано PyCharm. Драйвер для запуску тестів Appium через свою кросплатформність,

він дає доступ до тестування одночасно різних застосунків та браузерів на застосунку, тобто при наявності в мобільному застосунку зовнішніх джерел (наприклад, логін та реєстрація через Facebook або Gmail) є можливість доступу та, як наслідку, покриття автоматизованими тестами і цього функціоналу.

UIAutomator, було обрано як драйве взаємодії з інтерфейсом оскільки він дозволяю працювати не тільки з самим застосунком, але і протестувати застосунок в зв'язці з іншими застосунками та з функціями системи. Повністю підтримує метод тестування «чорної скриньки».

Була розроблена методика автоматизованого тестування та представлено її детальний опис. Розроблені діаграми переходів та на основі цієї діаграми сценарії тестування у вигляді таблиць.

Розроблена архітектура взаємодії інструментальних засобів та описано послідовність їх взаємодії. Розроблена структура проєкту. Наведено методи та функції які були використані при розробці автоматизованих тестів для проведення тестування мобільного застосунку.

На основі розробленої методики був розроблено алгоритм налаштування запуску інструментарії та програмного забезпечення виконання автоматизованих тестів. Були описані кроки цього алгоритму. Приведені експериментальні дослідження, які порівнюють виконання одних тих самих тестів у ручному режимі та з використанням створених автоматизованих тестів.

За результатами проведеного експериментального тестування методики автоматизованого тестування мобільних застосунків, можна зробити висновок, що впровадження автоматизованих тестів в процес розробки дає змогу прискорити тестування програмного продукту на 70-85%. Отже, завдання розробки методики а також завдання створення автоматизованих тестів на основі методики, виконані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Méndez-Porras, A. Automated Testing of Mobile Applications: A Systematic Map and Review / A. Méndez-Porras, C. Quesada-López, M. Jenkins // Proceeding of the Ibero American Conference on Software Engineering. – 2015. – P.362-375.
2. Итоги развития мобильного рынка в 2019 году [Электрон. ресурс]. – Режим доступа: <https://www.shopolog.ru/metodichka/analytics/itogi-razvitiya-mobil-nogo-rynka-v-2019-godu/>.
3. Uddin, A. Importance of Software Testing in the Process of Software Development / A. Uddin, A. Anand // International Journal for Scientific Research & Development. – Vol. 6, Issue 12. – 2019. – P.141-145. |
4. Тестирование программного обеспечения [Электрон. ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Тестирование_программного_обеспечения.
5. Anwar, N. Review Paper on Various Software Testing Techniques & Strategies / N. Anwar, S. Kar // Global Journal of Computer Science and Technology: Software and Data Engineering. – Volume 19, Issue 2. – 2019. – P.43-49.
6. Уиттакер, Дж. Как тестируют в Google / Дж. Уиттакер, Дж. Арбон, Дж. Каролло. – СПб.: Питер, 2014. – 320 с.
7. Куликов С., Тестирование программного обеспечения. Базовый курс. ЕРАМ Systems, 2015 [Электрон. ресурс]. – Режим доступа: http://svyatoslav.biz/software_testing_book/
8. Позитивное и негативное тестирование [Электрон. ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/positive-negative-testing/>
9. Скроботов, А.А. Тестирование и контроль качества программного обеспечения / А.А. Скроботов, Т.Ю. Чернышева // Всероссийская научно-практическая конференция для студентов и учащихся

молодежи «Прогрессивные технологии и экономика в машиностроении». – 2019. – С. 161-163. [Электрон. ресурс]. – Режим доступа: http://earchive.tpu.ru/bitstream/11683/53097/1/conference_tpu-2019-C57_p161-163.pdf

10. Савин, Р.И. Тестирование ДотКом, или Пособие по жестокому обращению с багами в интернет-стартапах / Р.И. Савин. – М.: Дело, 2007. – 312 с.

11. Блэк, Р. Ключевые процессы тестирования / Р. Блэк. – Лори, 2014. – 566 с.

12. Особенности тестирования приложений на мобильных устройствах [Электрон. ресурс]. - Режим доступа: <https://training.qatestlab.com/blog/technical-articles/testing-mobile-devices/>

13. Котляров В., Тренинг Основы тестирования программного обеспечения [Электрон. ресурс]. – Режим доступа: <https://intuit.ru/studies/courses/48/48/lecture/1438?page=2>

14. Криспин, Л. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд. : Пер. с англ / Л. Криспин, Д. Грегори. – М. : ООО “И.Д. Вильямс”, 2010. – 464 с.

15. Балыков, Е. Тестирование программных средств / Е. Балыков, В. Царев // RSDN Magazine – 2006. – 4 [Электрон. ресурс]. - Режим доступа: <http://www.rsdn.ru/article/testing/SoftwareTesting.xml>.

16. Хашими, С. Разработка приложений для Android / С. Хашими, М. Коматинени. – СПб.: Питер, 2011. – 736 с.

17. Software test documentation : IEEE 829- 1983. – [Effective from 1983-02-18]. – IEEE Standart, 1983. – 21p.

18. Тестирование и качество ПО [Электрон. ресурс]. – Режим доступа: <http://software-testing.ru/>

19. Форум про программирование для Android [Электрон. ресурс]. – Режим доступа: <http://www.cyberforum.ru/android-dev/>

20. Путеводитель по инструментам автотестирования мобильных

приложений [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/company/badoo/blog/347986/>

21. Which is the best tool for mobile application testing: Appium, Selendroid or Robotium? [Electronic source]. – Access mode: <https://www.quora.com/Which-is-the-best-tool-for-mobile-application-testing-Appium-Selendroid-or-Robotium>

22. Инструменты для автоматизированного тестирования мобильных приложений от А до Z [Электрон. ресурс]. – Режим доступа: <https://qaat.ru/instrumenty-dlya-avtomatizirovannogo-testirovaniya-mobilnyx-prilozhenij-ot-a-do-z/>

23. Драйвер [Электрон. ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80>

24. Тестирование на Android: UIAutomator против Espresso [Электрон. ресурс]. – Режим доступа: <http://www.ohandroid.com/android-uiautomator-espresso.html>

25. Mobile Web Automated Testing for iOS with XCUITest [Electronic source]. – Access mode: <https://medium.com/@eugeneberezintest/mobile-web-automated-testing-for-ios-with-xcuitest-c0639e0293f8>

26. Путеводитель по инструментам автотестирования мобильных приложений [Электрон. ресурс]. – Режим доступа: https://www.software-testing.ru/library/testing/mobile-testing/2735-mobile-autotesting#cl_wrapper

27. Инструменты автоматизации приложений для Android и iPhone [Электрон. ресурс]. – Режим доступа: <https://askentire.net/q/instrumenty-avtomatizacii-prilozhenii-dlya-android-i-iphone-36085369171>

28. Appium Overview [Electronic source]. – Access mode: <https://automationlab0000.wordpress.com/2018/09/10/appium/>

29. Фреймворк [Электрон. ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA>

30. Обзор языков программирования для автоматизированного тестирования [Электрон. ресурс]. – Режим доступа: <https://training.qatestlab.com/blog/technical-articles/overview-of-programming-languages-for-automated-testing/>

31. Плюсы и минусы программирования на Java [Электрон. ресурс]. – Режим доступа: <https://medium.com/nuances-of-programming/плюсы-и-минусы-программирования-на-java-2861f4c2a0d5>

32. Преимущества и недостатки языка Python Java [Электрон. ресурс]. – Режим доступа: <https://blog.ithillel.ua/articles/preimushchestva-i-nedostatki-yazyka-python>

33. Developer Survey Results 2019 [Electronic source]. – Access mode: <https://insights.stackoverflow.com/survey/2019>

34. 7 важных языков программирования для автоматизации тестирования в 2020 году [Электрон. ресурс]. – Режим доступа: <https://www.kv.by/post/1059518-7-yazykov-programmirovaniya-dlya-avtomatizacii-testirovaniya-v-2020-godu>

35. Паттерн Page Object Model (POM) в Selenium Webdriver [Электрон. ресурс]. – Режим доступа: <http://autoqa.org/advanced-webdriver/patterns-page-object-model-pom-selenium.html>

36. 5 лучших IDE с открытым исходным кодом для Python-разработчиков [Электрон. ресурс]. – Режим доступа: <https://techrocks.ru/2019/03/14/5-best-open-source-ides-for-python/>

37. Почему PyCharm – отличный выбор для автоматизации [Электрон. ресурс]. – Режим доступа: <https://software-testing.ru/library/testing/testing-automation/3099-why-python-is-great-for-test-automation>

38. Лучшие IDE и редакторы кода для Python [Электрон. ресурс]. – Режим доступа: <https://tproger.ru/translations/python-ide/>

39. Python: три лучших открытых IDE [Электрон. ресурс]. – Режим доступа: <https://www.itweek.ru/foss/article/detail.php?ID=179508>

40. Настройка автоматического тестирования гибридного приложения [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/452742/>

41. Правила улаштування електроустановок : ПУЕ-2017. – На заміну ПУЕ-86. - [Чинний від 2017-08-21]. – К. : Міненерговугілля України, 2017. – 617 с.

42. Грибан, В. Охорона праці / В. Грибан, А. Негодченко. К. : Центр навчальної літератури, 2017. – 280 с.

43. Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом: ДСТУ Б В.2.5-82:2016. – [Чинний від 2017-04-01]. – Київ, ДП УкрНДНЦ, 2016. – 109 с. (Національний стандарт України).

44. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями: НПАОП 0.00-7.15-18. – [Чинні від 2018-04-25]. – Харків: Форт, 2018. – 48 с. (Державні санітарні правила і норми).

45. Санітарні норми мікроклімату виробничих приміщень: ДСН 3.3.6.042-99. – [Чинні від 1999-12-01]. – К.: МОЗ України, 2000. – 24 с. (Державні санітарні правила і норми).

46. Санітарні норми виробничого шуму, ультразвуку та інфразвуку: ДСН 3.3.6.037-99. – [Чинні від 1999-12-01]. – К.: МОЗ України, 1999. – 24 с. (Державні санітарні правила і норми).

47. Природне та штучне освітлення: ДБН В.2.5-28:2018. - [Чинні від 2019-03-01]. - К: Мінрегіон України, 2018. – 133 с. (Державні будівельні норми України).

48. Правила пожежної безпеки в Україні. - [Чинні від 2015-03-05]. – К.: МВСУ, 2015. – 124 с. (Наказ МВС України).

49. Пожежна безпека об'єктів будівництва. Загальні вимоги: ДБН В.1.1-7:2016. - [Чинні від 2017-06-01]. – Київ: Міністерство регіонального розвитку, будівництва та житлово-комунального господарства України, 2017. – 39 с. (Державні будівельні норми України).

50. Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою : ДСТУ Б В.1.1-36:2016. – На заміну НАПБ Б.03.002-2007. - [Чинний від 2017-01-01]. – К. : Мінрегіонбуд України, 2016. – 66 с. (Національний стандарт України).

51. Правила будови електроустановок. Електрообладнання спеціальних установок: НПАОП 40.1-1.32-01. - [Чинний від 2001-06-21]. - Харків: Індустрія, 2010. – 118 с. (Державний нормативний акт про охорону праці).

52. Сусло, С.Т. Цивільний захист: Навч. посібник / С.Т. Сусло, В.М. Заплатинський, Г.М. Харамда. За ред.. проф.. М.О. Біляковича. – К.: Арістей, 2007.– 386 с

ДОДАТОК А
Текст програми

A.1 Login_test_failed.py

```
def login_test_invalid_number(driver):
    driver.find_element_by_id('number').send_keys('+1782645
4739')
    driver.find_element_by_id('password').send_keys('pass12
34')
    driver.find_element_by_id('.btn_action').click()
    assert driver.find_element_by_id('.error-
button').is_displayed()

def login_test_invalid_password(driver):
    driver.find_element_by_id('number').send_keys('+1782645
4788')
    driver.find_element_by_id('password').send_keys('pass11
11')
    driver.find_element_by_id('.btn_action').click()
    assert driver.find_element_by_id('.error-
button').is_displayed()
```

A.2 Login_test_success.py

```
def login_test_valid_login(driver):
    driver.find_element_by_id('number').send_keys('+1782645
4788')
    driver.find_element_by_id('password').send_keys('pass12
34')
    driver.find_element_by_id('.btn_action').click()
    assert
    driver.find_element_by_id('Home_titel').is_displayed()
```

A.3 main.py

```
import time

from appium import webdriver
from appium.webdriver.common.touch_action import
TouchAction
from selenium.webdriver.common.alert import Alert
from selenium.webdriver.common.by import By

from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import
expected_conditions as EC
from selenium.webdriver.common.keys import Keys
```

```
desired_capabilities = {
    "platformName": "Android",
    "platformVersion": "9.0",
    "deviceName": "Android Emulator",
    "app":
    "/Users/DoneIT/PycharmProjects/Project2/venv/app_binari
es/Ascend-devTest.apk"
}
```

```
desired_capabilities1 = {
    "platformName": "Android",
    "platformVersion": "8.0",
    "deviceName": "Android Emulator",
    "app":
    "/Users/DoneIT/PycharmProjects/Project2/venv/app_binari
es/Ascend-devTest.apk"
}
```

```
desired_capabilities2 = {
    "platformName": "Android",
    "platformVersion": "8.1",
    "deviceName": "Android Emulator",
    "app":
    "/Users/DoneIT/PycharmProjects/Project2/venv/app_binari
es/Ascend-devTest.apk"
}
```

```
public driver =
webdriver.Remote('http://127.0.0.1:4723/wd/hub',
desired_capabilities=desired_capabilities)
driver.implicitly_wait(20)
```

A.4 Registration_test_siccess.py

```
def registration_valid_test (driver):
Alert alert = driver.switchTo().alert()
actions = TouchAction(driver)
actions.press(x=334, y=940).move_to(x=334,
y=600).release().perform()
time.sleep(5)
driver.find_element_by_xpath("/hierarchy/android.widget
.FrameLayout/android.widget.LinearLayout/android.widget
```

```

.FrameLayout/android.widget.LinearLayout/android.widget
.FrameLayout/android.view.ViewGroup/android.widget.Scro
llView/android.view.ViewGroup/android.view.ViewGroup[1]
/android.view.ViewGroup/android.widget.LinearLayout/and
roid.widget.FrameLayout/android.widget.EditText").click
()
driver.back()
driver.find_element_by_id("com.doneit.ascend:id/signUpB
uttonACTV").click()

driver.find_element_by_id("fullName").click()
driver.send_keys("Oleh")
driver.find_element_by_id("email").click()
driver.send_keys("test@getnada.com")
driver.find_element_by_id("phoneNumber").click()
driver.send_keys("+380938308864")
driver.find_element_by_id("password").click()
driver.send_keys("pass1234")
driver.find_element_by_id("confPassword").click()
driver.send_keys("pass1234")
driver.find_element_by_id("continueButton").click()
assert
driver.find_element_by_id("firstTimeLoginScreen").is_di
splayed()
driver.quit()

```

A.5 Registration_test_failed.py

```

Alert alert = driver.switchTo().alert()
actions = TouchAction(driver)
actions.press(x=334, y=940).move_to(x=334,
y=600).release().perform()
time.sleep(5)
driver.find_element_by_xpath("/hierarchy/android.widget
.FrameLayout/android.widget.LinearLayout/android.widget
.FrameLayout/android.widget.LinearLayout/android.widget
.FrameLayout/android.view.ViewGroup/android.widget.Scro
llView/android.view.ViewGroup/android.view.ViewGroup[1]
/android.view.ViewGroup/android.widget.LinearLayout/and
roid.widget.FrameLayout/android.widget.EditText").click
()
driver.back()
driver.find_element_by_id("com.doneit.ascend:id/signUpB
uttonACTV").click()

```

```

def registration_invalid_fullName(driver):
driver.find_element_by_id("fullName").click()
driver.send_keys("O")
driver.find_element_by_id("email").click()
driver.send_keys("test@getnada.com")
driver.find_element_by_id("phoneNumber").click()
driver.send_keys("+380938308864")
driver.find_element_by_id("password").click()
driver.send_keys("pass1234")
driver.find_element_by_id("confPassword").click()
driver.send_keys("pass1234")
assert
driver.find_element_by_xpath("//div[contains(text(),
'Please enter your Full Name
correctly')]").is_displayed()

```

```

def registration_invalid_emailFirst(driver):
driver.find_element_by_id("fullName").click()
driver.send_keys("Oleh")
driver.find_element_by_id("email").click()
driver.send_keys("test@getnadacom")
driver.find_element_by_id("phoneNumber").click()
driver.send_keys("+380938308864")
driver.find_element_by_id("password").click()
driver.send_keys("pass1234")
driver.find_element_by_id("confPassword").click()
driver.send_keys("pass1234")
assert
driver.find_element_by_xpath("//div[contains(text(),
'The email is incorrect')]").is_displayed()

```

```

def registration_invalid_emailSecond(driver):
driver.find_element_by_id("fullName").click()
driver.send_keys("Oleh")
driver.find_element_by_id("email").click()
driver.send_keys("testgetnada.com")
driver.find_element_by_id("phoneNumber").click()
driver.send_keys("+380938308864")
driver.find_element_by_id("password").click()
driver.send_keys("pass1234")
driver.find_element_by_id("confPassword").click()
driver.send_keys("pass1234")
assert

```

```
driver.find_element_by_xpath("//div[contains(text(),
'The email is incorrect')]").is_displayed()
```

```
def registration_invalid_phoneNum(driver):
driver.find_element_by_id("fullName").click()
driver.send_keys("Oleh")
driver.find_element_by_id("email").click()
driver.send_keys("test@getnada.com")
driver.find_element_by_id("phoneNumber").click()
driver.send_keys("+38093830886")
driver.find_element_by_id("password").click()
driver.send_keys("pass1234")
driver.find_element_by_id("confPassword").click()
driver.send_keys("pass1234")
assert
driver.find_element_by_xpath("//div[contains(text(),
'The number is invalid')]").is_displayed()
```

```
def registration_invalid_passwordNumber(driver):
driver.find_element_by_id("fullName").click()
driver.send_keys("Oleh")
driver.find_element_by_id("email").click()
driver.send_keys("test@getnada.com")
driver.find_element_by_id("phoneNumber").click()
driver.send_keys("+380938308864")
driver.find_element_by_id("password").click()
driver.send_keys("pass")
driver.find_element_by_id("confPassword").click()
driver.send_keys("pass")
assert
driver.find_element_by_xpath("//div[contains(text(),
'The password should contain at least 1
number')]").is_displayed()
```

```
def registration_invalid_passwordLetter(driver):
driver.find_element_by_id("fullName").click()
driver.send_keys("Oleh")
driver.find_element_by_id("email").click()
driver.send_keys("test@getnada.com")
driver.find_element_by_id("phoneNumber").click()
driver.send_keys("+380938308864")
driver.find_element_by_id("password").click()
driver.send_keys("1234")
driver.find_element_by_id("confPassword").click()
driver.send_keys("1234")
```

```

assert
driver.find_element_by_xpath("//div[contains(text(),
'The password should contain at least 1 latin
letter')]").is_displayed()

def registration_invalid_passwordDontMatch(driver):
driver.find_element_by_id("fullName").click()
driver.send_keys("Oleh")
driver.find_element_by_id("email").click()
driver.send_keys("test@getnada.com")
driver.find_element_by_id("phoneNumber").click()
driver.send_keys("+380938308864")
driver.find_element_by_id("password").click()
driver.send_keys("pass1234")
driver.find_element_by_id("confPassword").click()
driver.send_keys("pass1233")
assert
driver.find_element_by_xpath("//div[contains(text(),Pas
swords don't match')]").is_displayed()
driver.quit()

```

ДОДАТОК Б
Слайди презентації

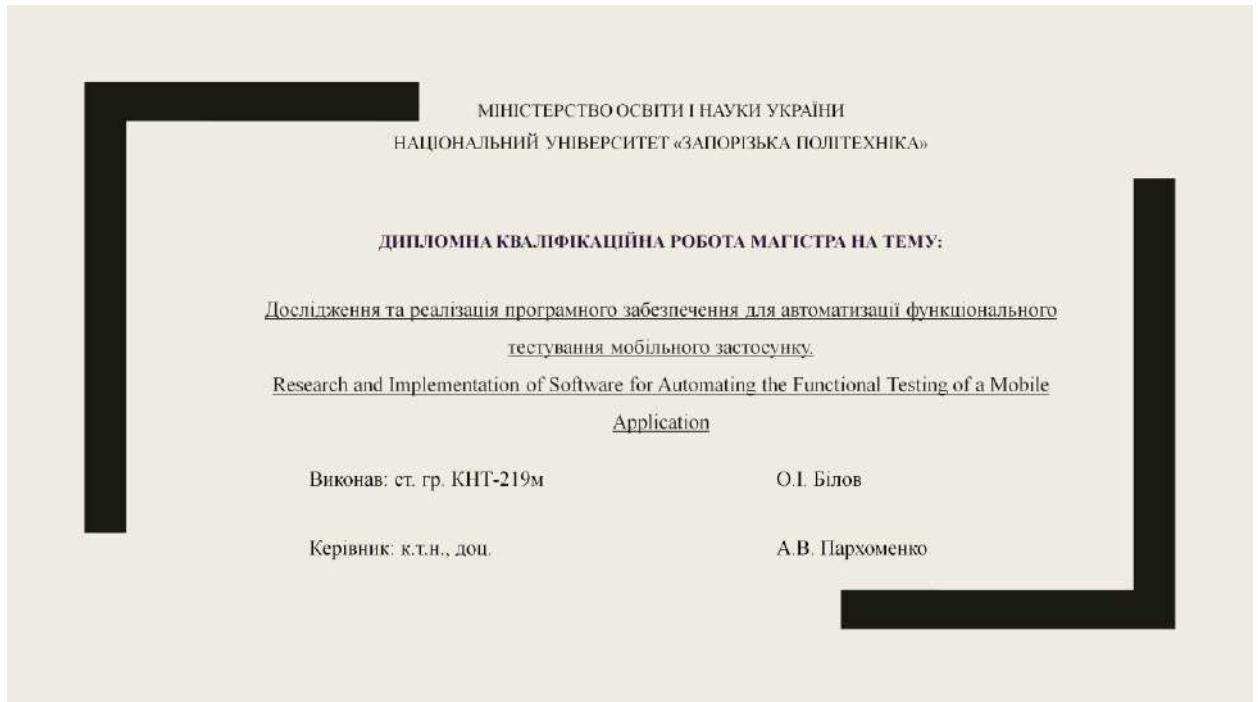


Рисунок Б.1 – Титульний слайд

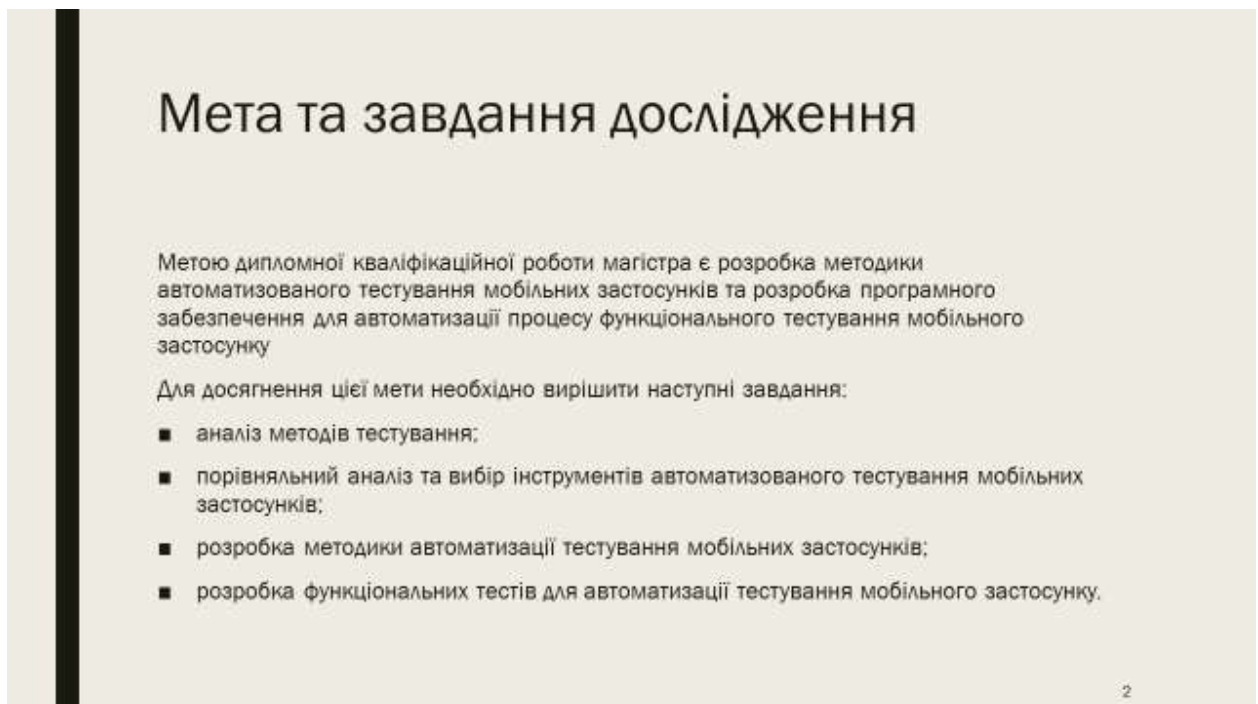


Рисунок Б.2 – Мета та завдання дослідження

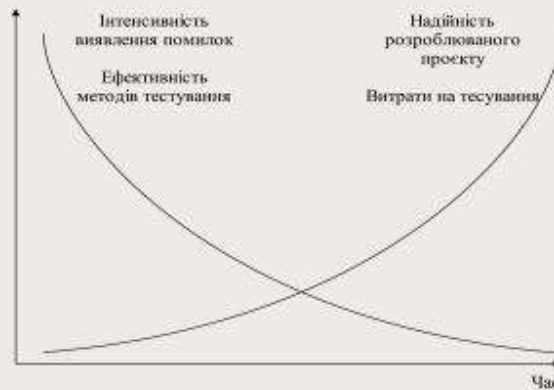
Класифікація методів тестування



3

Рисунок Б.3 – Класифікація методів тестування

Дослідження методів тестування



4

Рисунок Б.4 – Дослідження методів тестування

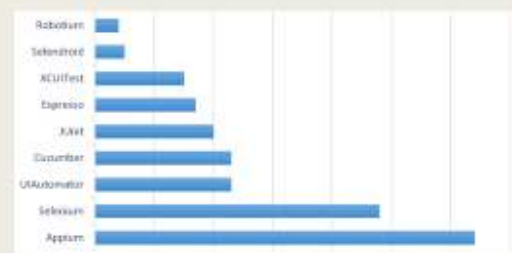
Автоматизація тестів



5

Рисунок Б.5 – Автоматизація тестів

Інструментальні засоби автоматизації тестування



6

Рисунок Б.6 – Інструментальні засоби автоматизації тестування

Вибір інструментів взаємодії з мобільним застосунком

Інструмент	Android	iOS	Мови	Доступ до коду	>1 застосунок	Реальні пристрої
UIAutomator	+	-	Java, Kotlin	-	+	+
Espresso	+	-	Java, Kotlin	+	-	+
XCUITest	-	+	ObjectiveC Swift	-	-	+
Robotium	+	-	Java	-	-	+
Selendroid	+	-	Yes	-	-	+

Інструмент	Android	iOS	Мови	Доступ до коду	>1 застосунок	Реальні пристрої
Calabash	+	-	Ruby JRuby	-	-	+
Appium	+	+	Більше	-	+	+
Earl Giny	-	+	ObjectiveC Swift	+	-	+
WebDriver Agent	-	+	ObjectiveC Swift	-	+	+

7

Рисунок Б.7 – Вибір інструментів взаємодії з мобільним застосунком

Вибір інструменту розробки та виконання тестів

Середовище	Переваги	Недоліки
PyCharm	Підтримка бібліотек Selenium і Appium та безліч інших засобів полегшують роботу тестувальників автоматизаторів. Велике інтернет ком'юніті. Можна редагувати, запускати і налагоджувати Python-код.	Може повільно завантажуватися, а настройки за замовчуванням, деколи доведеться підкоригувати для існуючих проектів.
Sublime	Велика кількість користувачів. Як редактор коду. Sublime Text не є безкоштовним, є пробний період. Sublime Text швидкий, легкий і має хорошу підтримку.	Установка розширень може бути складною. В редакторі немає підтримки налагодження і запуску коду.
PyDev (Eclipse)	Надає всі переваги розробки великого середовища Eclipse.	Для новачків Python у Eclipse може бути тяжким, оскільки IDE велике і вимагає більше досвіду для повноцінного використання.
VS Code	Завдяки Electron, VS Code доступний на всіх платформах з дивно великою функціональністю. Крім застосунок. Продукти Microsoft є платними, того, вихідні коди можна знайти у відкритому доступі.	Замішаний Electron, значить VS Code не нативний застосунок. Продукти Microsoft є платними.

8

Рисунок Б.8 – Вибір інструменту розробки та виконання тестів

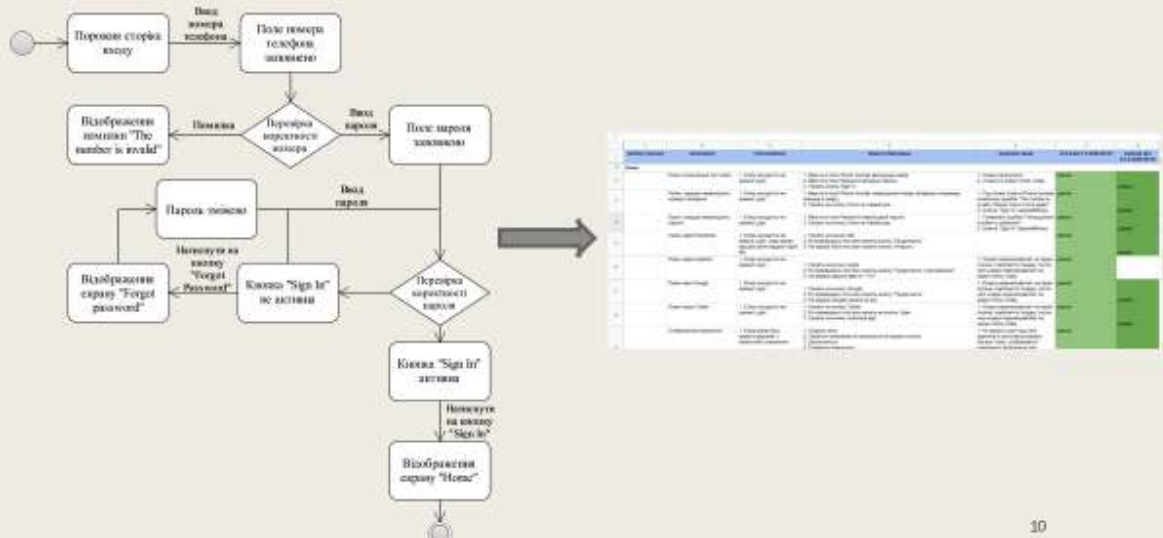
Методика автоматизації тестування мобільних застосунків



9

Рисунок Б.9 – Методика автоматизації тестування мобільних застосунків

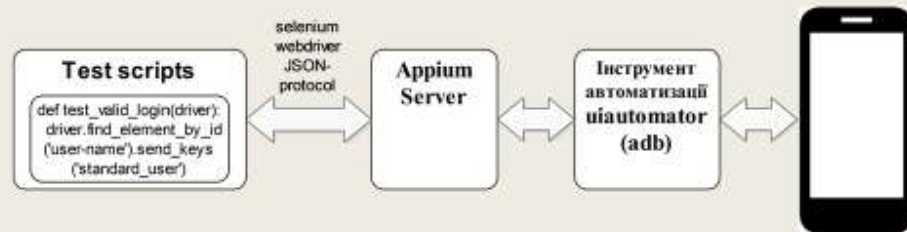
Діаграма переходів та тест-сценарії



10

Рисунок Б.10 – Діаграма переходів та тест-сценарії

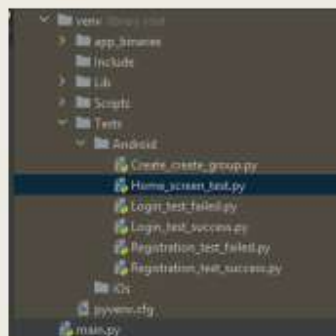
Архітектура взаємодії автоматизованих тестів та мобільного застосунку



11

Рисунок Б.11 – Архітектура взаємодії автоматизованих тестів та мобільного застосунку

Структура проєкту



Test має дві вкладені папки: Android та iOS, в яких знаходяться готові автоматизовані тести.

app_binaries – містить файли застосунків.

Login_test_success.py, Login_test_failed.py – позитивні та негативні сценарії тестування входу у застосунок.

Registration_test_failed.py, Registration_test_success.py – автоматизовані тести для негативних та позитивних сценаріїв реєстрації.

Home_screen_test.py – тести головної сторінки.

Create_group.py – тести основної функції застосунку - створення груп.

main.py – описи усіх методів, констант, елементів, виклику драйверу

12

Рисунок Б.12 – Структура проєкту

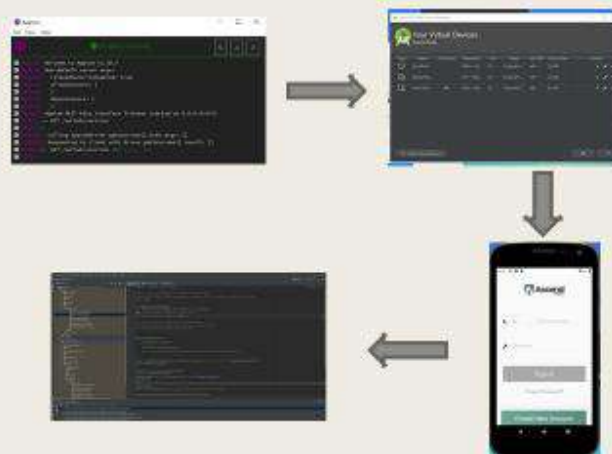
Методи та функції автоматизованих тестів

Метод, функція	Опис
<code>find_element_by_id()</code>	Знайти елемент за його ідентифікатором.
<code>find_element_by_path()</code>	Знайти елемент за його xpath ідентифікатором.
<code>driver.click()</code>	Зробити тап по необхідному елементу екрану.
<code>implicitly_wait()</code>	Очікування.
<code>driver.quit()</code>	Закрити та завершити роботу застосунку.
<code>driver.back()</code>	Повернутися до попереднього стану екрану.
<code>assert()</code>	Перевіряє отриманий результат тесту.
<code>TouchAction()</code>	Метод для виконання дій користувача, наприклад, такого як проводження пальцем по екрану.
<code>findLoginScreenElements()</code>	Метод ініціалізація всіх елементів екрану входу.
<code>testValidLogin()</code>	Метод позитивного тесту входу до застосунк.
<code>testInvalidLoginTel()</code>	Метод негативного тесту входу до застосунк.

13

Рисунок Б.13 – Методи та функції автоматизованих тестів

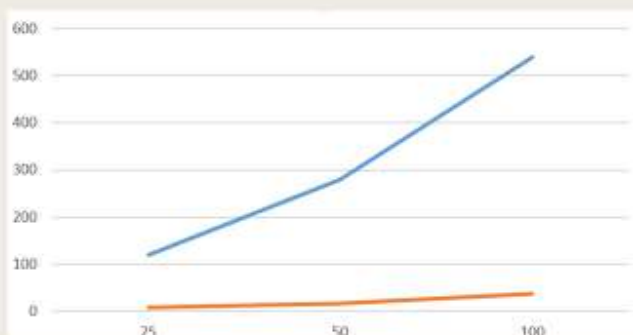
Алгоритм роботи з автоматизованими тестами



34

Рисунок Б.14 – Алгоритм роботи з автоматизованими тестами

Експериментальне дослідження



15

Рисунок Б.15 – Експериментальне дослідження

Висновки

- проведено аналіз методів тестування, та підходів автоматизованого тестування;
- виконано порівняльний аналіз виходячи з якого було обрано інструменти автоматизованого тестування мобільних застосунків;
- розроблено методику автоматизації тестування мобільних застосунків;
- розроблено функціональні тестів для автоматизації тестування мобільного застосунку.

Наукова новизна роботи полягає у тому, що розроблено методику автоматизованого тестування мобільних застосунків, в якій отримав подальший розвиток підхід до функціонального тестування програмного забезпечення за методом «чорної скриньки».

Практична значимість роботи полягає у тому, що розроблено автоматизовані функціональні тести для проведення тестування мобільного застосунку, які дозволили скоротити час тестування на 70-85%.

16

Рисунок Б.16 – Висновки