Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування інституту, назва факультету (відділення))

Кафедра програмного забезпечення
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка методу і програмного засобу відлагодження пристроїв під керування операційної системи Android

Виконав: студент 2 курсу, групи 1ПІ-20м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Савицький Д.С.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Ракитянська Г.Б.
(прізвище та ініціали)
«__» _____ 2021 р.

Опонент: д. т. н., професор Іванчук Я.В.
(прізвище та ініціали)
«__» _____ 2021 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н. проф. Романюк О.Н.
«__» _____2021 р.

Вінниця ВНТУ – 2021 рік

Вінницький національний технічний
університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«13» вересня 2021р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Савицькому Дмитру Сергійовичу

1. Тема роботи – Розробка методу і програмного засобу відлагодження пристроїв під керування операційної системи Android.

Керівник роботи: Ракитянська Ганна Борисівна, к.т.н. доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 24 » вересня 2021 р. № 277.

2. Строк подання студентом роботи
____1 грудня 2021 р._____

3. Вихідні дані до роботи: базові методи відлагодження пристроїв, системи розробки та тестування мобільних додатків, мобільні пристрої, додаткові комплектуючі для зв'язку пристроїв, методи зчитування системної інформації та логів пристроїв; вихідні дані – розроблена програма для відлагодження пристроїв під керуванням ОС Android.

4. Зміст пояснювальної записки: Порівняльний аналіз аналогів, аналіз методів розв'язання поставленої задачі, аналіз методів відлагодження пристроїв, постановка задачі для відлагодження пристроїв, розробка структури та інтерфейсу програми, розробка структури інтерфейсу для програмного засобу відлагодження пристроїв, розробка алгоритмів роботи програми, варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу, розробка модуля зчитування системної інформації, розробка модуля зчитування логів пристрою, тестування програмного забезпечення, розробка інструкції користувача

5. Перелік графічного матеріалу: приклади аналогів, блок-схеми та UML-діаграми додатку, результати тестування.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ракитянська Г.Б. к.т.н. доцент кафедри ПЗ		
5	Ратушняк О.Г., к.е.н., доцент кафедри ЕПіВМ		

7. Дата видачі завдання ___14 вересня 2021 р._____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного стану питання та обґрунтування задачі	15.09. 2021 – 03.10.2021	Вик.
2	Розробка методу відлагодження і моделі програмного додатку	03.10.2021 – 18.10.2021	Вик.
3	Розробка програмного додатку	18.10.2021 – 05.11.2021	Вик.
4	Тестування програмного додатку	05.11.2021 – 19.11.2021	Вик.
5.	Економічна частина	19.11.2021 - 30.11.2021	Вик.

Студент _____ **Савицький Д.С.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ракитянська Г.Б**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 621.374.415

Савицький Д.С. Магістерська кваліфікаційна робота зі спеціальності 121 – інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2021. 125 с.

У магістерській кваліфікаційній роботі розроблено програмний засіб для відлагодження пристроїв під керування операційної системи Android. Вона дозволяє покращити спосіб відлагодження мобільних пристроїв під керування операційної системи Android, шляхом розробки та програмної реалізації алгоритмів відлагодження.

У загальній частині роботи розглянуто особливості методів відлагодження пристроїв, а також обґрунтована доцільність розробки системи. У розрахунково-конструкторській частині виконана розробка медодів та моделі програмної системи. У технологічній частині розроблений та протестований програмний продукт для виконання поставлених задач.

У економічній частині опрацьовано такі питання, як оцінювання комерційного потенціалу розробки, прогнозування витрат на виконання науково-дослідної роботи, розрахунок економічної ефективності науково-технічної розробки, розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Ключові слова: мобільний пристрій, мобільний додаток, операційні системи, Android, відлагодження, розробка ПЗ.

ABSTRACT

Savytskyi D.S. Master's degree in specialty 121 – software engineering, educational program – software engineering. Vinnytsia: VNTU, 2021. 125 p.

In the master’s thesis, a software tool for debugging devices running the Android operating system has been developed. It allows you to improve the way you debug mobile devices running the Android operating system by developing and software debugging algorithms.

In the general part of the work the peculiarities of the methods of device debugging are considered, as well as the expediency of systems development is substantiated. In the calculation and design part, the development of methods and models of the software system was performed. In the technological part the software product for performance of the set tasks is developed and tested.

In the economic part, such issues as assessing the commercial potential of development, forecasting the cost of research, calculating the economic efficiency of scientific and technical development, calculating the effectiveness of investments and the period of their purchase.

Keywords: mobile device, mobile application, operating systems, Android, debugging, software development.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Стрімкий розвиток технологій за останнє століття призвів до суттєвого зменшення розмірів усієї комп'ютерної техніки. Якщо у 60-70х роках XX століття ЕОМ могла займати цілу кімнату, то сьогодні сучасні комп'ютери можуть поміститись у кишені. Крім того, за останні 50 років був здійснений прорив у бездротових комунікаціях. Поєднання телефонів та комп'ютерів призвело до появи так званих смартфонів. На сьогодні смартфон є невід'ємною частиною сучасного суспільства, а ринок мобільних додатків оцінюється в не один мільярд доларів США.

Велика кількість мобільних пристроїв продаються з уже встановленим набором мобільних застосунків, таких як веб браузер, поштовий клієнт, календар (наприклад, Google Calendar), застосунок для придбання та прослуховування музики та інші. Деякі, попередньо встановлені застосунки, можуть бути вилучені з мобільного пристрою користувачем, за допомогою звичайного процесу видалення, звільняючи більше місця для зберігання інших (бажаних) застосунків.

Розробка програмного забезпечення для мобільних пристроїв потребує врахування їх обмежень та можливостей. Мобільні пристрої працюють на акумуляторі та мають менш потужні процесори, ніж персональні комп'ютери, а також мають більше функцій, таких як визначення місцезнаходження та камери. Розробникам також доводиться враховувати широкий спектр розмірів дисплею, різні технічні характеристики та конфігурації обладнання через сильну конкуренцію мобільного програмного забезпечення та зміни в кожній платформі.

Розробка програм для мобільних платформ вимагає використання спеціалізованих інтегрованих середовищ розробки. Мобільні застосунки спочатку тестуються в середовищі розробки, використовуючи емулятори, а потім перевіряються на місцях. Однією з важливих складових тестування мобільних додатків є їх відлагодження за допомогою читання параметрів пристрою та логів його роботи.

Налагодження – це процес виявлення та видалення наявних і потенційних помилок (також званих «багами») у програмному коді, які можуть призвести до неочікуваної поведінки або збою системи. Щоб запобігти некоректній роботі програмного забезпечення або системи, налагодження використовується для пошуку та усунення помилок або дефектів. Налагодження є важливою частиною визначення, чому операційна система чи програма працюють неправильно. Коли різні підсистеми або модулі тісно пов’язані, налагодження стає значно важчим, оскільки будь-яка зміна в одному модулі може призвести до появи нових помилок в іншому.

Однак, усі наявні на даний час засоби відлагодження містять у собі певні недоліки, а саме: більшість з них не є універсальними та вимагають переробки і підключення до кожного окремого додатку; інші, більш універсальні засоби є вимогливими до системи, однак не надають такого широкого спектру параметрів відлагодження та зняття параметрів пристрою.

Саме тому розробка програми для відлагодження пристроїв під керуванням операційної системи Android є актуальною на сьогоднішній день.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою розробки є підвищення надійності процесу відлагодження та розширення його функціональних можливостей за рахунок нового методу збору та опрацювання даних шляхом створення моделі та реалізації програмної системи читання параметрів пристрою та логів його роботи.

У відповідності до поставленої мети в роботі необхідно вирішити такі завдання:

- розробити метод відлагодження мобільних пристроїв;
- розробити модель системи відлагодження мобільних пристроїв;
- розробити інтерфейс програмного продукту, який буде інтуїтивно зрозумілим для користувача;

- розробити програмний продукт, призначений для вирішення проблеми;
- провести тестування програмного продукту для перевірки всіх можливих варіантів використання.

Об’єкт дослідження – процеси та технології відлагодження мобільних пристроїв.

Предмет дослідження – методи та програмні засоби для відлагодження мобільних пристроїв.

Методи дослідження. У процесі досліджень використовувались: теорія зв’язку пристроїв та передачі даних між ними для розробки методу відлагодження; методи системного аналізу для аналізу надійності і стійкості системи; теорія алгоритмів для розробки алгоритмів роботи програми; методи програмування мобільних додатків для розробки програми під Android; методи, засоби та інструменти тестування для контролю якості програмного забезпечення.

Наукова новизна отриманих результатів. Подальшого розвитку дістав метод відлагодження мобільних пристроїв, який, на відміну від існуючих, оптимізує зв’язок мобільного пристрою та робочого комп’ютера розробника та забезпечує підвищення ефективності та надійності процесу пошуку та виправлення проблем та суттєво знижує ризики, пов’язані з недостатньою якістю розроблюваних мобільних додатків.

Подальшого розвитку дістала модель системи, яка, на відміну від існуючих, забезпечує універсальний підхід до пристроїв та мобільних додатків, що дозволяє скоротити час розробки, тестування та відлагодження додатків, знизити вартість процесу та підвищити його ефективність.

Практична цінність отриманих результатів полягає у тому, що отримані в магістерській роботі методи використано для розробки та реалізації алгоритмів відлагодження мобільних пристроїв.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У науковій роботі, опублікованій у співавторстві [1], автору належать такі результати: метод

відлагодження пристроїв, модель системи для відлагодження мобільних пристроїв під керуванням ОС Android.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідались на Міжнародній науково-практичній Інтернет конференції «Електронні інформаційні ресурси: створення, використання, доступ - 2021».

Достовірність теоретичних положень підтверджена результатами тестування розробленого додатку.

Публікації. Результати досліджень опубліковано в 1 науковій праці, а саме, в матеріалах наукової конференції.

Структура та обсяг роботи. Робота містить вступ, п’ять розділів, висновки, перелік використаних джерел, три додатки.

В першому розділі визначено загальний стан питання розробки мобільних додатків, розглянуто існуючі реалізації, та засоби розробки, приведено обґрунтування вибору мов програмування та середовища розробки.

У другому розділі описано засоби і методи відлагодження мобільних пристроїв та розроблено архітектуру програмного засобу.

У третьому розділі виконано проектування інтерфейсу середовища, проведено розробку програми для відлагодження пристроїв під керуванням операційної системи Android, проаналізовано отримані результати.

У четвертому розділі проведено тестування програмного засобу.

У п’ятому розділі проведено економічне обґрунтування доцільності та економічної вигоди даного програмного продукту.

Перелік посилань містить 22 джерела. У додатках міститься технічне завдання на магістерську кваліфікаційну роботу, протокол перевірки роботи, лістинги коду та ілюстративний матеріал до захисту роботи.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ТА ОБҐРУНТУВАННЯ ЗАДАЧІ

1.1 Аналіз сучасного стану питання

Мобільний додаток – це комп’ютерна програма або програмне забезпечення, призначене для роботи на мобільних пристроях, таких як телефон, планшет або годинник. Мобільні програми часто відрізняються від настільних програм, які призначені для запуску на настільних комп’ютерах, і веб-програм, які працюють у мобільних веб-браузерах, а не безпосередньо на мобільному пристрої.

Програми в цілому поділяються на три типи: нативні програми, гібридні та веб-програми. Нативні програми розроблені спеціально для мобільної операційної системи, як правило, iOS або Android. Веб-програми написані на HTML5 або CSS і зазвичай запускаються через браузер. Гібридні додатки створюються з використанням таких веб-технологій, як JavaScript, CSS і HTML 5, і функціонують як веб-програми, замасковані в нативному контейнері [2].

Більшість мобільних пристроїв постачаються разом з кількома програмами, які входять у комплект попередньо встановленого програмного забезпечення, наприклад, поштовий клієнт, веб-браузер, онлайн-карти, календар, а також програма для покупки музики, інших медіа чи програм. Певні попередньо встановлені програми можна видалити за допомогою звичайного процесу видалення, таким чином залишивши більше місця для зберігання потрібних. Якщо програмне забезпечення не дозволяє цього, деякі пристрої можуть бути рутовані, щоб усунути небажані програми.

Дослідники передбачають, що в 2022 році буде завантажено 114 мільярдів додатків (89% з них безкоштовні), що принесе 28,9 мільярдів доларів США, що на 51,6% більше, ніж 18,5 мільярдів доларів США в 2021 році. До другого кварталу 2021 року лише магазини Google Play та Apple заробили 4,9 мільярдів доларів. За оцінками аналітиків, мобільні додатки приносить дохід понад 10 мільярдів євро

на рік у Європейському Союзі, тоді як у 28 країнах ЄС створено понад 529 000 робочих місць через зростання ринку додатків.

Як частина процесу розробки, дизайн мобільних інтерфейсів користувача є важливим у створенні мобільних додатків. Мобільний UI розглядає обмеження пристрою, екран, введення та портативність як рамки для створення зовнішнього вигляду. Майже завжди користувач є центральною частиною взаємодії зі своїм девайсом, а користувацький інтерфейс включає компоненти як апаратного, так і програмного забезпечення. Введення інформації дає змогу користувачам керувати системою, а вихід пристрою дозволяє системі вказувати на наслідки маніпулювання користувачами. Ліміти дизайну мобільного інтерфейсу користувача включають в себе обмежений розмір та форм-фактори, такі як розмір екрану мобільного пристрою [3].

Оскільки програмне забезпечення та електронні системи загалом стали складнішими, різноманітні загальні методи налагодження розширилися за допомогою нових методів виявлення аномалій, оцінки впливу та планування виправлення програмного забезпечення або повного оновлення системи.

Використання слова «баг» як синоніма помилки виникло в програмній інженерії. Застосування цього терміну до обчислювальної техніки та натхнення для використання слова налагодження як синоніма усунення несправностей приписують адміралу Грейс Хоппер, одній з піонерок комп’ютерного програмування, яка також була відома своїм строгим та сухим почуттям гумору. Коли справжня моль потрапила між електричними реле і спричинила проблему в першому комп’ютері ВМС США, адмірал Хоппер та її команда «налагодили» комп’ютер і витягли мотиля з корпусу комп’ютера. Зараз засушений жук знаходиться в експозиції технічного відділу Смітсонівського музею.

Налагодження варіюється за складністю від виправлення простих помилок до виконання тривалих і виснажливих завдань зі збору даних, аналізу та планування оновлення. Навички налагодження програміста можуть бути основним фактором у здатності налагоджувати проблему, але складність налагодження програмного забезпечення значно залежить від складності системи,

а також певною мірою залежить від використовуваної мови програмування. і доступні інструменти, такі як відлагоджувачі. Відлагоджувачі — це програмні засоби, які дозволяють програмісту контролювати виконання програми, зупиняти її, перезапускати, встановлювати точки зупину та змінювати значення в пам'яті.

Безумовно, код, який інженери пишуть як розробники програмного забезпечення, не завжди виконує те, чого вони від нього очікували. Іноді він виконує щось зовсім інше. Коли таке станеться, наступне завдання — з’ясувати, чому, і хоча може виникнути спокуса просто дивитися на програмний код годинами, набагато простіше та ефективніше використовувати інструмент налагодження або відлагоджувач.

На жаль, відлагоджувач – це не той засіб, який може чарівним чином виявити всі проблеми чи помилки в коді. Налагодження означає поетапний запуск коду в засобі відлагодження, як-от Visual Studio, щоб знайти точне місце в програмному коді, де була допущена програмна помилка. Тоді стане зрозуміло, які виправлення потрібно внести у свій код, а інструменти відлагодження часто дозволяють вносити тимчасові зміни, щоб ви могли продовжувати працювати з програмою [4].

Налагодження та тестування є взаємодоповнювальними процесами. Мета тестування полягає в тому, щоб визначити, що відбувається, коли є помилка у вихідному коді програми. Мета налагодження – знайти та виправити помилку.

Процес тестування не допомагає розробнику з’ясувати, в чому полягає помилка кодування – він просто показує, які наслідки помилка кодування має на програму. Після виявлення помилки налагодження допомагає розробнику визначити причину помилки, щоб її можна було виправити.

1.2 Порівняльний аналіз аналогів

Існують як комерційні, так і безкоштовні інструменти для різних мов. Що за певними твердженнями, можуть виявляти сотні різних проблем. Ці інструменти можуть бути надзвичайно корисними при перевірці дуже великих дерев вихідних

кодів, де недоцільно виконувати ознайомлення з кодом. Типовим прикладом виявленої проблеми може бути ситуація неіменованної змінної, яке відбувається до того, як змінній буде присвоєно значення. Як інший приклад, деякі такі інструменти виконують сильну перевірку типів, коли мова цього не вимагає. Таким чином, вони краще знаходять ймовірні помилки в коді, який синтаксично правильний.

При проведенні порівняльного аналізу аналогів, було розглянуто такі додатки “Android Studio”, “Android Debugging Bridge” та “Android Remote Debugger”

Android Remote Debugger – Система для відлагодження що дає широкі можливості для перегляду стану пристрою та його контролю через браузер. Телефон підключається не кабелем як зазвичай а через мережу. Незважаючи на багатий функціонал, програма не є універсальною, оскільки має бути вбудована в сам додаток.

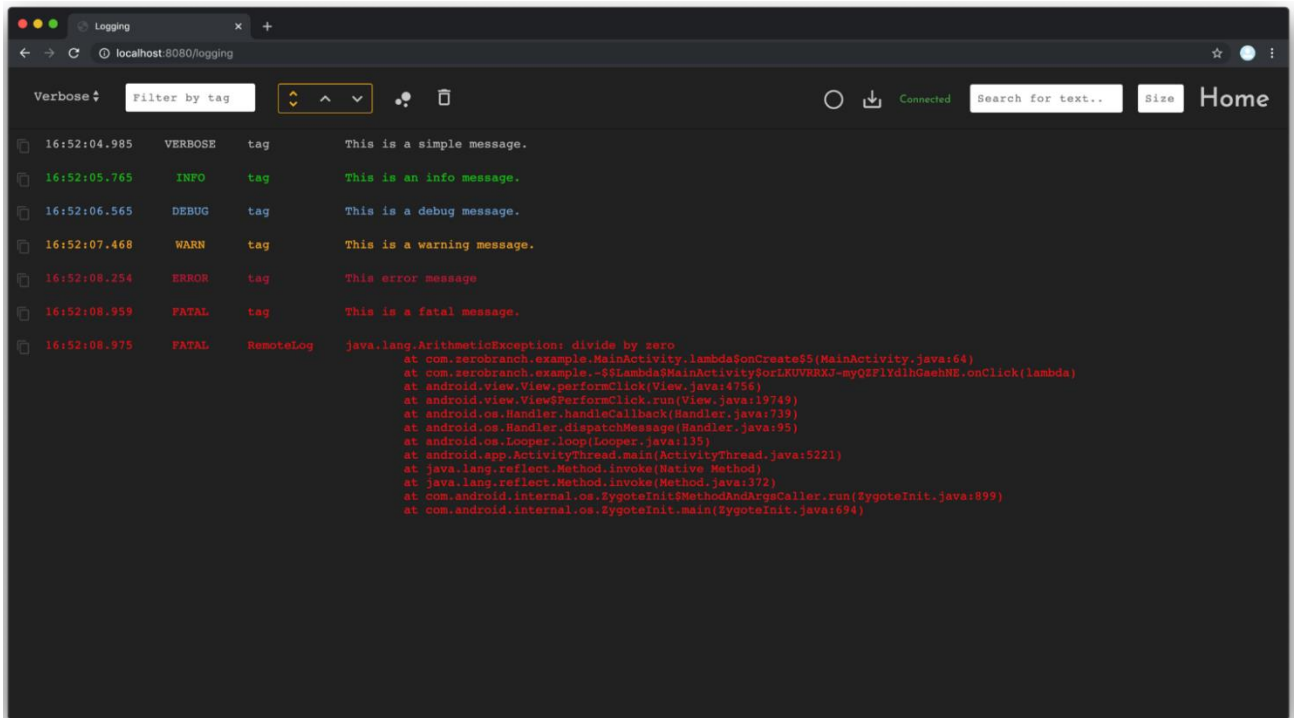

Рисунок 1.1 – Вікно Android Remote Debugger

Pluto — це відлагоджувач для мобільних пристрої з відкритим вихідним кодом для програм написаних для операційної системи Android, який допомагає

перевіряти запити та відповіді HTTP, фіксувати збої та помилки ANR та маніпулювати даними програми на ходу. Pluto дозволяє зберігати таку інформацію, як статус програми (наприклад, конфігурації програми), властивості користувача (наприклад, електронна пошта, профіль) і відбиток пальця пристрою (наприклад, IMEI). Пізніше ці дані можна отримати через інтерфейс налагодження Pluto. Цей метод можна викликати кілька разів, і він продовжуватиме додавати дані. Він має інтерфейс користувача для моніторингу та обміну інформацією, а також API для доступу та використання цієї інформації у програмі.

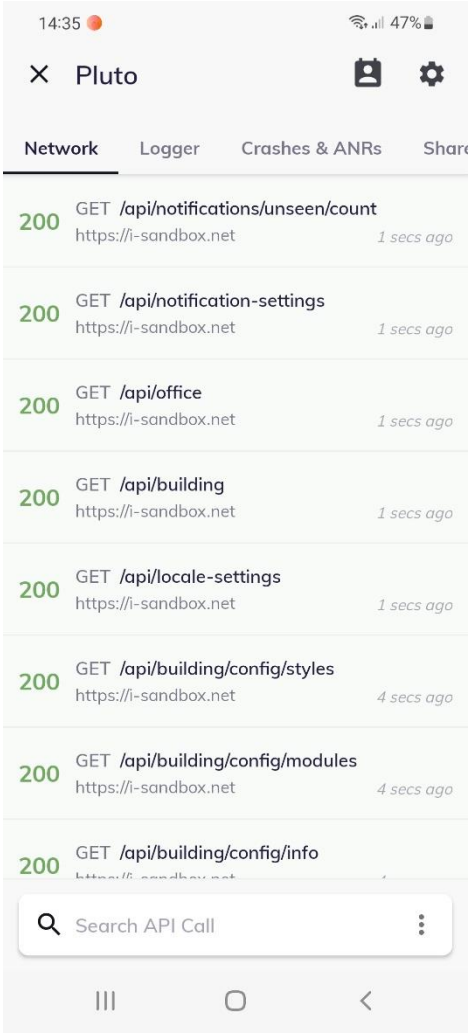

Рисунок 1.2 – Вікно Pluto Debugger

Android Studio — це офіційне інтегроване середовище розробки (IDE) для операційної системи Google Android, побудоване на програмному забезпеченні

IntelliJ IDEA від JetBrains і розроблене спеціально для розробки Android. Він доступний для завантаження в операційних системах на базі Windows, macOS і Linux або як служба на основі підписки в 2020 році. Це заміна Eclipse Android Development Tools (E-ADT) як основної IDE для розробки додатків Android. Середовище включає в себе платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності.

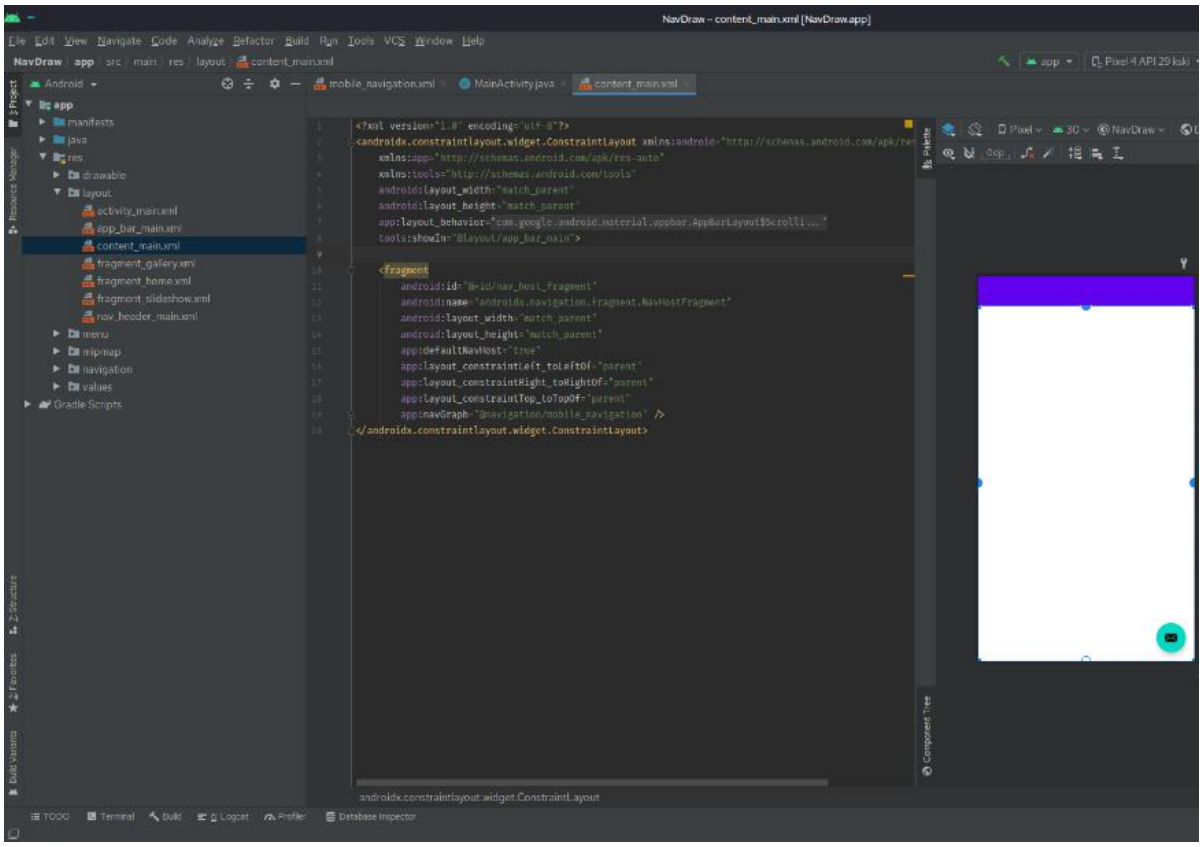

Рисунок 1.3 – Вікно Android Studio

Android Debug Bridge (adb)-це універсальний інструмент командного рядка, що дозволяє спілкуватися з пристроєм. Команда adb полегшує різноманітні дії з пристроєм, такі як встановлення та відлагодження програм, а також надає доступ до оболонки Unix, яку можна використовувати для виконання різноманітних команд на пристрої. Це клієнт-серверна програма, що включає три компоненти:

- Клієнт, який надсилає команди.
- Даемон(adbd), який запускає команди на пристрої.
- Сервер, який керує зв'язком між клієнтом і даемоном.

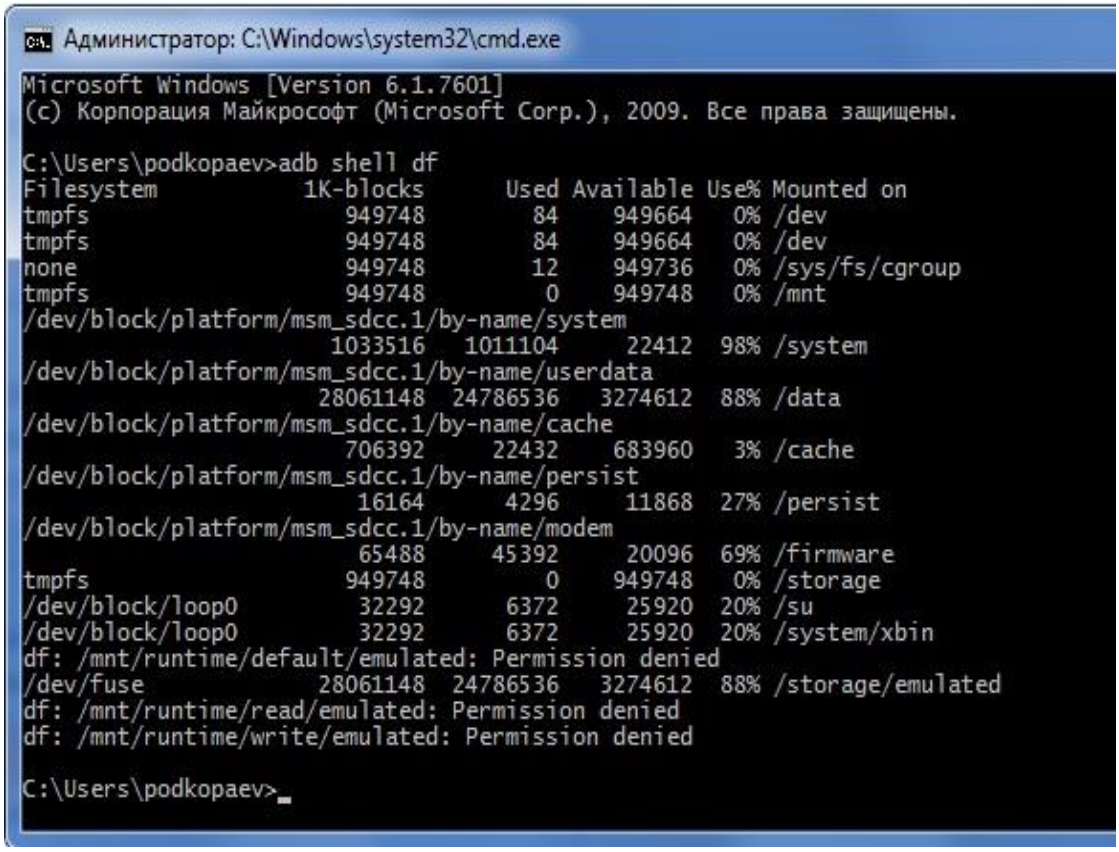

Рисунок 1.4 – Командний рядок adb

Результати порівняльного аналізу власного додатку та його аналогів наведено в таблиці 1.1.

Таблиця 1.1 Порівняння програмних додатків

Параметр	Додаток			
	Android Remote Debugger	Android Studio	Android Debugging Bridge	SuperDebug
Графічний інтерфейс	1	1	0	1
Універсальність	0	1	1	1
Кросплатформовість	1	1	1	1
Низька ресурсозатратність	0	0	1	1
Загальний коефіцієнт	2	3	3	4

У результаті аналізу таблиці доведено актуальність розробки власного програмного додатку для відлагодження пристроїв під керування операційної системи Android, оскільки сумарний коефіцієнт переваг вищий ніж у наявних аналогів.

1.3 Обґрунтування вибору мови програмування та середовища розробки

При виборі мови програмування для розробки програмного додатку було розглянуто кілька варіантів: C++, C#, Java та Python.

C++ – об’єктно орієнтована мова програмування високого рівня вперше випущена в 1985 році, з 1979 розроблялась Б’ярном Страуструпом в AT&T Bell Laboratories на основі мови C. Також відома під початковою назвою «C з класами». Мова C++ складається з двох основних компонентів: пряме відображення апаратних функцій, наданих головним чином підмножиною C, і абстракції з нульовими накладними витратами на основі цих відображень. Страуструп описує C++ як «легку мову програмування абстракції розроблену для створення та використання ефективних та елегантних абстракцій»; і «пропонування як апаратного доступу, так і абстракції є основою C++. Ефективне виконання – це те, що відрізняє його від інших мов» [5].

C# – об'єктно-орієнтована мова програмування для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде в період з 1998 по 2001 роки. Синтаксис мови має багато спільних рис разом із Java, на яку в свою чергу сильно повпливав C++. Перевагами C# є висока швидкість виконання програмного коду, інтеграція в систему .NET та ОС Microsoft Windows. C# на відміну від Java не підтримує технологію Just-in-time compilation та не є платформо-незалежною [6].

Java – це об’єктно-орієнтована мова програмування і покликана мати якомога менше залежностей від платформи розробки. З 2009 над розвитком мови працює корпорація Oracle, яка того ж самого року викупила компанію Sun

Microsystems. В офіційній програмній реалізації програми написані на Java компілюються у байт-код, що при запуску інтерпретується віртуальною машиною для кожної конкретної платформи. Компільований код Java може працювати на кожній платформі, що підтримує Java без потреби перекомпіляції (compile once, run anywhere). Станом на 2021 рік, Java є однією з найпопулярніших мов програмування у світі, останнім часом, в основному, для додатків з клієнт-серверною архітектурою [7].

Python – інтерпретована мова програмування високого рівня, що має динамічну типізацію та підтримує об’єктно орієнтовану парадигму програмування. Мова була розроблена в 1990 році в Нідерландах Гвідо ван Россумом. Python має підтримку модулів та пакетів модулів, що в свою чергу сприяє модульності та повторному використанню коду. Багата стандартна бібліотека є однією з найбільш привабливих особливостей мови Python. Крім того, в цій мові програмування існує підтримка декількох основних парадигм програмування, серед них: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [8].

Таблиця 1.2 Порівняння мов програмування

Параметр	Додаток			
	C++	C#	Java	Python
Автоматичне керування пам’яттю	0	1	1	1
Ручне управління пам’яттю	1	0	0	0
Статична типізація	1	1	1	0
Множинне наслідування	1	0	0	1
Режим командної стрічки	0	0	0	1
Загальний коефіцієнт	3	2	2	3

У результаті проведеного аналізу наявних мов програмування на основі даних з таблиці 1.2 було вирішено обрати Python як основну мову програмування

для розробки програмного додатку для відлагодження пристроїв під керування операційної системи Android.

Інтегроване середовище розробки (IDE) — це програмне забезпечення, яке надає програмістам комплексні засоби для розробки програмного забезпечення. IDE зазвичай складається щонайменше з редактора вихідного коду, інструментів автоматизації збірки та відлагоджувача. Інтегровані середовища розробки розроблені для максимального підвищення продуктивності програміста, забезпечуючи тісно пов’язані компоненти з подібними користувальницькими інтерфейсами. IDE представляють єдину програму, в якій здійснюється вся розробка.

При виборі середовища розробки було розглянуто декілька програмних комплексів, такі як Microsoft Visual Studio, Visual Studio Code, Dev C++ та Eclipse.

Dev C++ — вільне інтегроване середовище розробки для мов програмування C/C++. У дистрибутив входить компілятор MinGW. Сам Dev-C++ написаний на Delphi. Розповсюджується згідно з ліцензією GNU GPL. Свого часу був доступний Linux-порт, проте на теперішній час актуалізована тільки Windows-версія (Рис. 1.5).

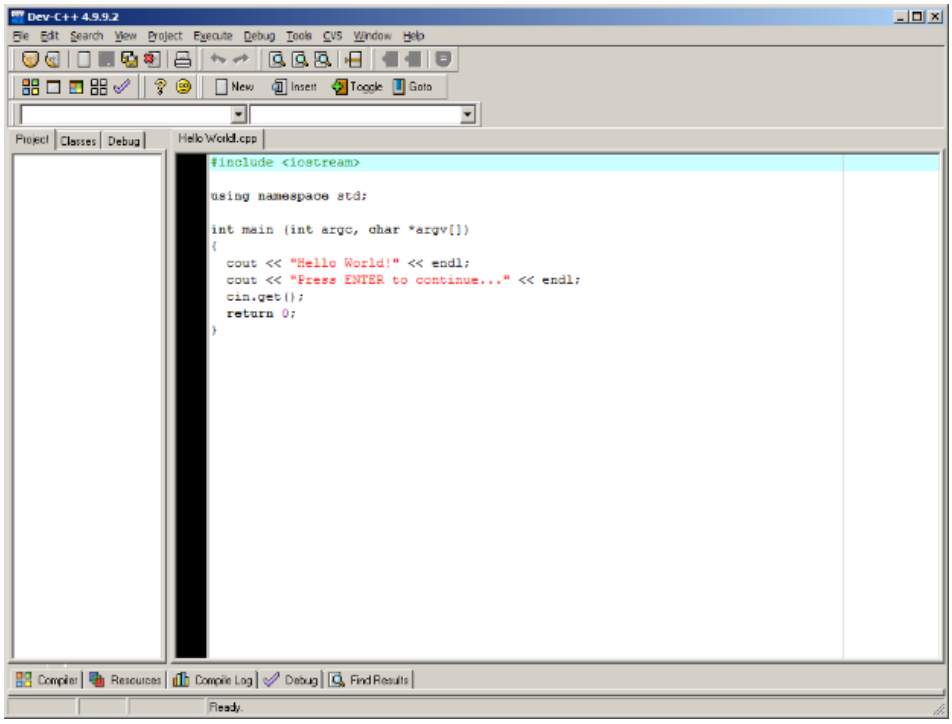

Рисунок 1.5 – Головне вікно Dev C++

Microsoft Visual Studio – серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки та веб-служби. Visual Studio підтримує декілька мов програмування в тому числі C++ та C# і дозволяє ефективно поєднувати їх в одному проекті [9]. Крім того це середовище розробки дає можливість підключення сторонніх плагінів (Рис 1.6).

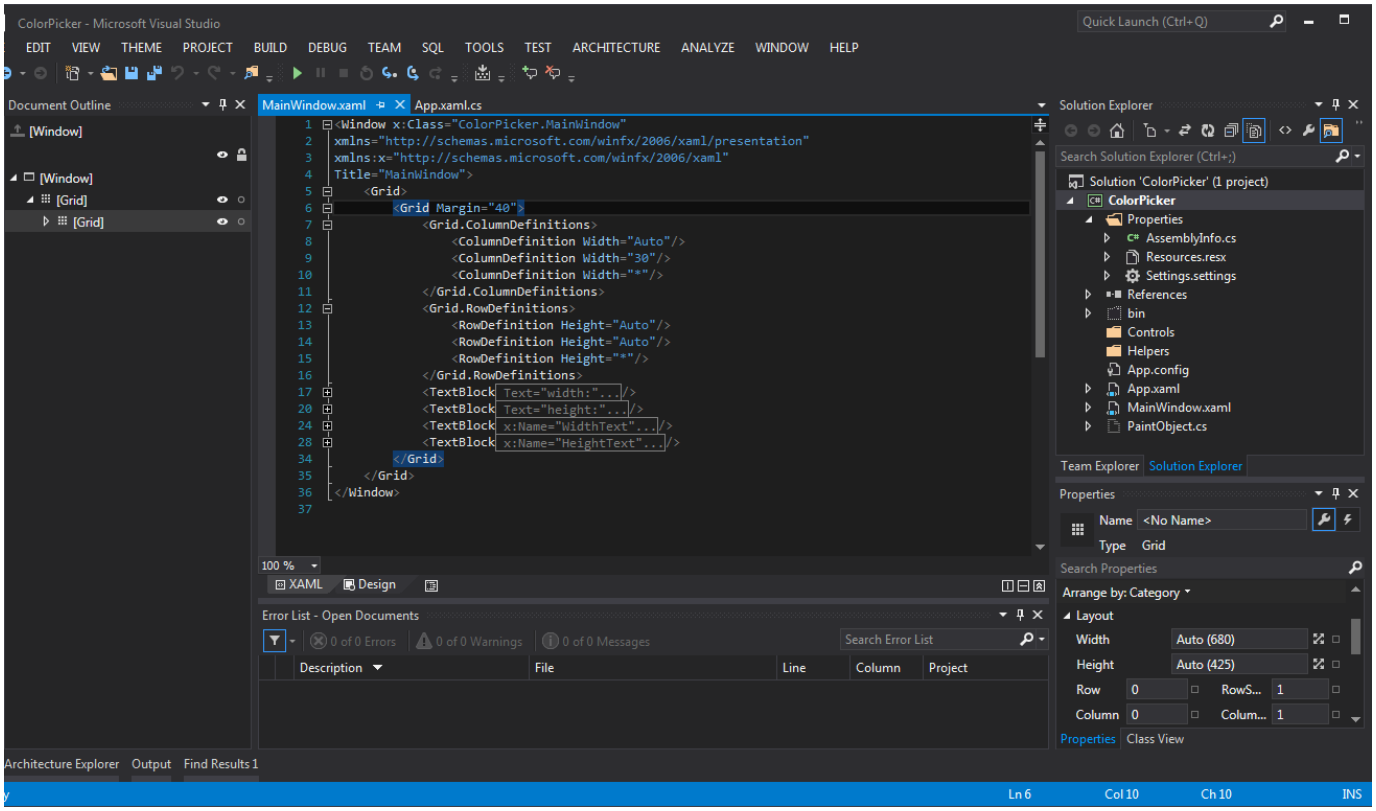

Рисунок 1.6 – Головне вікно Microsoft Visual Studio

Visual Studio включає редактор коду, який підтримує IntelliSense (компонент завершення коду), а також рефакторингу коду. Інтегрований відлагоджувач працює і як відлагоджувач на рівні джерела, і як відлагоджувач на рівні машини. Інші вбудовані інструменти включають профайлер коду, конструктор для створення додатків із графічним інтерфейсом користувача, веб-дизайнер, конструктор класів і конструктор схем баз даних. Він приймає плагіни, які

розширюють функціональність майже на всіх рівнях, включаючи додавання підтримки систем контролю джерел (наприклад, Subversion і Git) і додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для мов, що стосуються домену, або наборів інструментів для інших аспектів розробки програмного забезпечення. життєвий цикл (наприклад, клієнт Azure DevOps: Team Explorer). (Рис. 1.7).

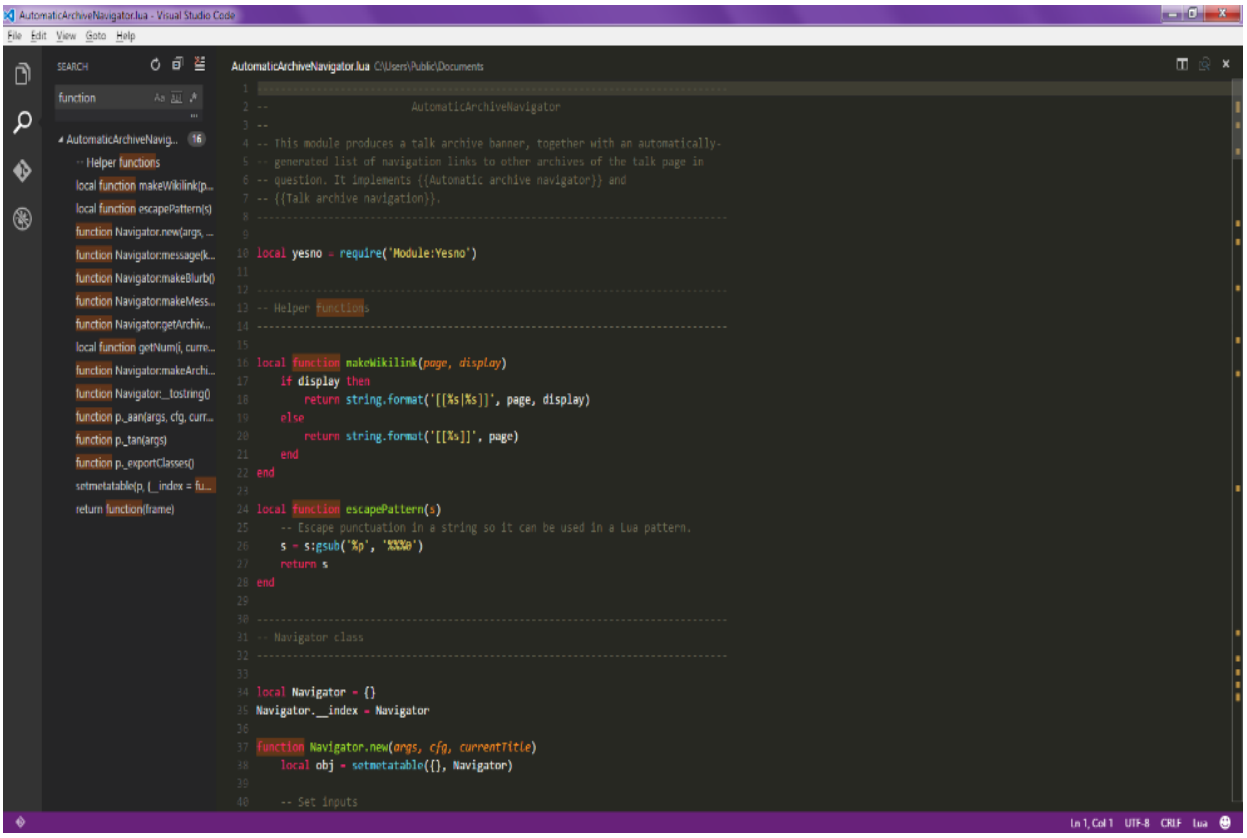

Рисунок 1.7 – Головне вікно Visual Studio Code

Eclipse – модульне інтегроване середовище розробки програмного забезпечення. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для програмістів на мові Java, системи контролю версій, конструктори GUI тощо. Програмний комплекс може бути використаний для розробки застосунків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи C, C++, PHP, Python, тощо (Рис. 1.8).

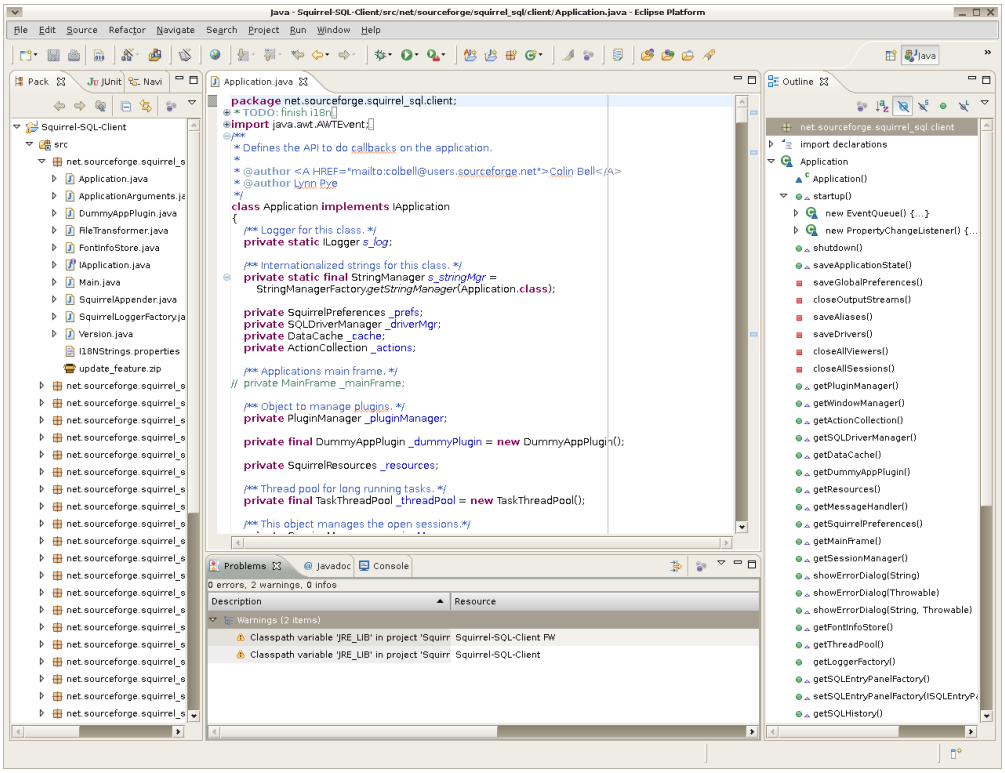

Рисунок 1.8 - Головне вікно Eclipse

Порівняльний аналіз за різними параметрами різних середовищ розробки представлено в таблиці 1.3

Таблиця 1.3 – Порівняння середовищ розробки

Параметр	Додаток			
	VS Code	Visual Studio	Dev C++	Eclipse
Статичний аналіз коду	1	0	0	1
Підтримка сторонніх плагінів	1	1	0	0
Розширені можливості відлагодження	1	0	1	0
Мультиплатформеність	0	1	0	1
Загальний коефіцієнт	3	2	1	2

За результатами аналізу даних з таблиці 1.3 було прийнято рішення обрати середовище розробки Microsoft Visual Studio Code серед усіх аналогів для розробки програмного додатку для відлагодження пристроїв під керування операційної системи Android.

1.4 Постановка задачі для відлагодження мобільних пристроїв

В результаті детального аналізу сучасного стану питання відлагодження мобільних пристроїв і методів його вирішення було визначено наступні завдання, які є необхідним виконати для створення програмного продукту:

- розробити метод відлагодження мобільних пристроїв;
- розробити модель відлагодження мобільних пристроїв;
- розробити програмний продукт, призначений для вирішення проблеми;
- провести тестування програмного продукту для перевірки всіх можливих варіантів використання.

Програма повинна працювати під управлінням таких операційних систем як Windows 10/11 та різним дистрибутивами Linux (наприклад Ubuntu 19.04 LTS і вище).

Мінімальна конфігурація:

- Тип процесора: 64-розрядний процесор з тактовою частотою 1 ГГц;
- Об’єм оперативної пам’яті: 512 МБ для 64-розрядної системи;
- Розмір жорсткого диску: 100 МБ для 64-розрядної системи;
- Графічний пристрій DirectX 11.

Рекомендована конфігурація:

- Тип процесора: 64-розрядний процесор з тактовою частотою 2 ГГц;
- Об’єм оперативної пам’яті: 1 ГБ для 64-розрядної системи;
- Розмір жорсткого диску: 100 МБ для 64-розрядної системи;
- Графічний пристрій DirectX 11.

1.5 Висновки

У результаті аналізу аналогів розроблюваного додатку було зроблено висновок про доцільність розробки власного додатку для відлагодження пристроїв під керування операційної системи Android, адже загальний коефіцієнт

власного додатку – 3, а у найближчого аналога – 2. Було розроблено основні завдання, які є необхідними для розробки програмного продукту.

У результаті аналізу об’єктно-орієнтованих мов програмування було обрано мову програмування Python для розробки програмного продукту, оскільки її загальний коефіцієнт складає 3, а решти аналогів – 2.

У результаті аналізу середовищ розробки було обрано середовище розробки Microsoft Visual Studio Code. адже загальний коефіцієнт Visual Studio Code – 3, а у найближчих аналогів – 2,5.

2 РОЗРОБКА МЕТОДУ ВІДЛАГОДЖЕННЯ І МОДЕЛІ ПРОГРАМНОГО ДОДАТКУ

2.1 Метод відлагодження мобільних пристроїв

Відлагодження – це процес виявлення та видалення наявних і потенційних помилок (також званих «багами») у програмному коді, які можуть призвести до неочікуваної поведінки або збою системи. Щоб запобігти некоректній роботі програмного забезпечення або системи, відлагодження використовується для пошуку та усунення помилок або дефектів. Коли різні підсистеми або модулі тісно пов’язані, відлагодження стає значно важчим, оскільки будь-яка зміна в одному модулі може призвести до появи нових помилок в іншому. Іноді на відлагодження програми потрібно більше часу, ніж на написання безпосередньо її коду.

Відлагодження само по собі є критично важливим етапом при розробці та тестуванні програмних продуктів. Використовується для пошуку прихованих проблем в системі та дослідження стабільності системи. Відповідно, якщо проблеми не будуть знайдені, або не буде можливості дослідити систему для того щоб їх виправити, то ризики будуть високі: від фінансових втрат, і в рідкісних випадках до судових позовів та закриття компанії.

Якісний процес відлагодження і тестування в цілому зменшує ризики пов'язані з неякісним ПЗ, та значно скорочує "ціну помилки" допущеної на етапі проектування чи розробки.

Як правило, процес відлагодження починається відразу після написання коду і продовжується на послідовних етапах, коли код об’єднується з іншими одиницями програмування для формування програмного продукту. У великій програмі, яка містить тисячі й тисячі рядків коду, процес відлагодження можна спростити за допомогою таких стратегій, як модульні тести, огляди коду та парне програмування.

Відлагодження є важливою частиною визначення, чому операційна система, програма чи програма працюють неправильно. Навіть якщо розробники використовують той самий стандарт кодування, велика ймовірність того, що нова програма все одно матиме помилки. Завжди першими виявляють і виправляють помилки в програмних компонентах, які найбільше використовуються.

Щоб виявити помилки, може бути корисно переглянути журнали коду та використовувати окремий інструмент відлагодження або режим відлагодження інтегрованого середовища розробки (IDE). На цьому етапі може бути корисно, якщо розробник знайомий зі стандартними повідомленнями про помилки. Однак, якщо розробники не коментують належним чином під час написання коду, навіть найчистіший код може стати проблемою для когось налагодити.

На жаль, відлагоджувач – це не той засіб, який може чарівним чином виявити всі проблеми чи помилки в коді. Налагодження означає поетапний запуск коду в засобі відлагодження, як-от Visual Studio, щоб знайти точне місце в програмному коді, де була допущена програмна помилка. Тоді стане зрозуміло, які виправлення потрібно внести у свій код, а інструменти відлагодження часто дозволяють вносити тимчасові зміни, щоб ви могли продовжувати працювати з програмою.

Налагодження та тестування є взаємодоповнювальними процесами. Мета тестування полягає в тому, щоб визначити, що відбувається, коли є помилка у вихідному коді програми. Мета налагодження – знайти та виправити помилку.

Процес тестування не допомагає розробнику з’ясувати, в чому полягає помилка кодування – він просто показує, які наслідки помилка кодування має на програму. Після виявлення помилки налагодження допомагає розробнику визначити причину помилки, щоб її можна було виправити.

Для того, щоб відагодити програму, користувач повинен почати з проблеми, виділити вихідний код проблеми, а потім виправити її. Користувач програми повинен знати, як вирішити проблему, оскільки очікується знання про аналіз проблеми. Коли помилку виправлено, програмне забезпечення готове до використання. Інструменти відлагодження (так звані відлагоджувачі)

використовуються для виявлення помилок кодування на різних етапах розробки. Вони використовуються для відтворення умов, в яких сталася помилка, а потім для вивчення стану програми на той момент і визначення причини. Програмісти можуть відстежувати виконання програми крок за кроком, оцінюючи значення змінних, і зупиняти виконання там, де це потрібно, щоб отримати значення змінних або скинути змінні програми. Деякі пакети мов програмування забезпечують відлагоджувач для перевірки коду на наявність помилок під час його написання під час виконання.

Нижче наведено процес відлагодження:

1. Відтворити проблему.

2. Описати помилку. Спробувати отримати якомога більше інформації від користувача, щоб отримати точну причину несправності.

3. Зробити знімок чи запис екрану програми, коли з'явиться помилка. Отримати всі значення змінних і станів програми на той момент.

4. Проаналізувати знімок на основі стану та дії. На основі цього спробувати знайти причину помилки.

5. Виправити існуючу помилку, та перевірити систему, чи не виникає жодної нової помилки.

Орієнтовна схема відлагодження показана на рисунку 2.1

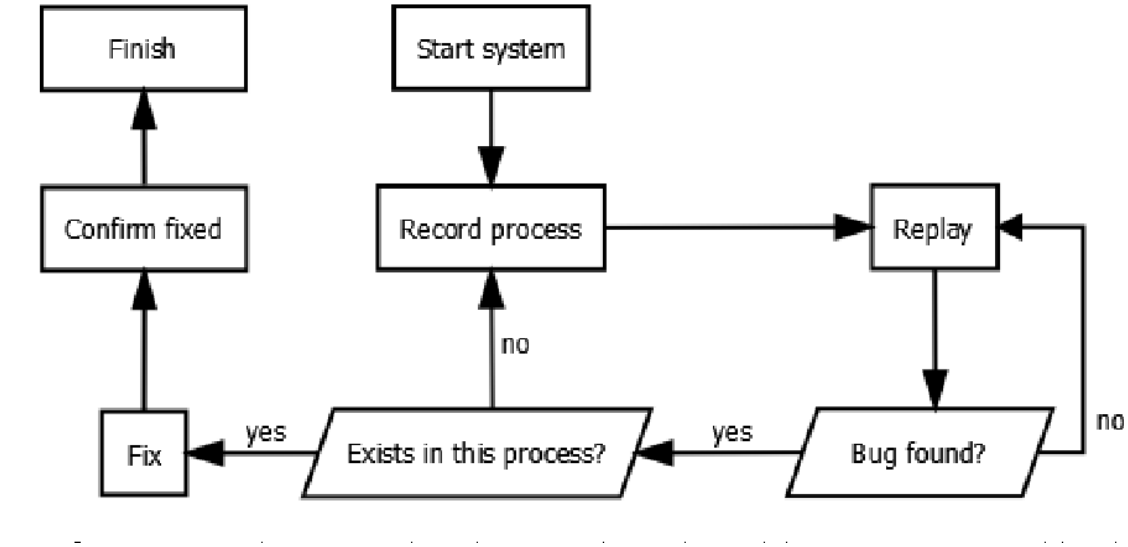

Рисунок 2.1 – Схема відлагодження

2.2 Модель системи відлагодження мобільних пристроїв

Враховуючи існуючий мето відлагодження пристроїв, буде розроблена покращена модель додатку. Мобільні застосунки спочатку тестуються в середовищі розробки, використовуючи емулятори, а потім перевіряються на місцях. Розробникам також доводиться враховувати широкий спектр розмірів дисплею, різні технічні характеристики та конфігурації обладнання через сильну конкуренцію мобільного програмного забезпечення та зміни в кожній платформі. Під час розробки та тестування виникає велика кількість технічних та програмних помилок, пов’язаних як з самим додатком, версією операційної системи, розміром екрану, версією прошивки, та навіть специфіки конкретного мобільного пристрою конкретного виробника. Список типових помилок в мобільних додатках наведений в таблиці 2.1.

Таблиця 2.1 – Список типових помилок при розробці мобільних додатків

Помилка	Частота виникнення	Ймовірність виявлення і виправлення	Витрати на тестування і доробку
Помилки в графічному інтерфейсі додатку	Дуже часто	0.95 – 0.99	Низькі
Серверні помилки	Дуже часто	0.90 – 0.95	Середні
Помилки в передачі даних між клієнтом та серверною частиною	Часто	0.70 – 0.80	Середні
Помилки в бізнес-логікі та функціях додатку	Часто	0.80 – 0.99	Середні
Системні помилки та аварійне завершення програми	Нечасто	0.99 – 1	Високі
Помилки безпеки пристрою та додатку	Рідко	0.3 – 0.7	Дуже високі
Витоки пам’яті	Рідко	0.3 – 0.5	Високі
Висока енерговитрата спричинена функціоналом додатку	Дуже рідко	0.4 – 0.6	Високі

Однією з важливих складових тестування мобільних додатків є їх відлагодження за допомогою читання параметрів пристрою та логів його роботи. Файл логів складається з часу повідомлення та рівня логування кожного повідомлення. Існує сім рівнів логування (Табл. 2.2)

Таблиця 2.2 – Рівні логування в ОС Android

Рівень	Опис
V: Verbose	Системні повідомлення низького рівня
D: Debug	Службова інформація для налагодження
I: Info	Інформаційне повідомлення
W: Warning	Попередження про потенційний збій
E: Error	Повідомлення про помилку
F: Fatal	Повідомлення про аварійне завершення роботи додатку
S: Silent	Відсутність повідомлень

Приклад фрагменту файлу логів показано на рисунку 2.2.

```
I SamsungAlarmManager: setLocked to kernel - T:3 / 20211122T153430, set=1550859083, now=1550857687
D WindowManager: finishDrawingWindow: Window{19b228e u0 NotificationShade} mDrawState=DRAW_PENDING
D ActivityTaskManager: setLockScreenShown(true,true) is called from 0
V UsbDeviceManager: onKeyguardStateChanged: isShowing:true secure:true user:0
D StorageManagerService: handleMessage -> H_KEYGUARD_CHANGED isShowing=true
D UsbDeviceManager: handleMessage -> MSG_UPDATE_SCREEN_LOCK: mScreenLocked=true
D UsbDeviceManager: getPersistProp: return=persist.sys.usb.config
D UsbDeviceManager: setEnabledFunctions: usbFunctions=mtp,conn_gadget,adb forceRestart=false
D UsbDeviceManager: trySetEnabledFunctions: usbFunctions=mtp,conn_gadget,adb forceRestart=false
D SecContentProvider: called from android.uid.system:1000
D RestrictionPolicy: isUsbDebuggingEnabled : true
D UsbDeviceManager: Update mCurrentFunctions=mtp,conn_gadget,adb mSecCurrentFunctions=mtp,conn_gadget,adb
D WifiDisplayAdapter: isFitToActiveDisplayLocked : false
D SamsungAlarmManager: setInexact (T:2/F:8/AC:false) 20211122T160429 now=1550857712 - CU:1000/CP:1324/L:1
I SamsungAlarmManager: setLocked to kernel - T:2 / 20211122T153434, set=1550862637, now=1550857712
I SamsungAlarmManager: setLocked to kernel - T:3 / 20211122T153430, set=1550859083, now=1550857712
D SamsungAlarmManager: setInexact (T:2/F:8/AC:false) 20211122T153529 now=1550857712 - CU:1000/CP:1324/L:9
I SamsungAlarmManager: setLocked to kernel - T:2 / 20211122T153434, set=1550862637, now=1550857712
I SamsungAlarmManager: setLocked to kernel - T:3 / 20211122T153430, set=1550859083, now=1550857712
I Telecom:SamsungTelecomServiceImpl: isInCall - callingPackage : com.android.systemui
I Telecom:SamsungTelecomServiceImpl: isInCall - callingPackage : com.android.systemui
D AdaptiveBrightnessLongtermModelBuilder: Received android.intent.action.SCREEN_OFF
I MdnieScenarioControlService: action  :  android.intent.action.SCREEN_OFF
I Telecom:SamsungTelecomServiceImpl: isInCall - callingPackage : com.android.systemui
I Telecom:SamsungTelecomServiceImpl: isInCall - callingPackage : com.android.systemui
D WifiDisplayAdapter: isFitToActiveDisplayLocked : false
I AdaptiveDisplaySolutionService: action  :  android.intent.action.SCREEN_OFF
I EyeComfortSolutionService: action  :  android.intent.action.SCREEN_OFF
D WifiDisplayAdapter: isFitToActiveDisplayLocked : false
D WifiDisplayAdapter: isFitToActiveDisplayLocked : false
D WifiDisplayAdapter: isFitToActiveDisplayLocked : false
D SamsungAlarmManager: Cancel Alarm calling from uid:1000 pid :1324 / L:73cbcd4
I SamsungAlarmManager: setLocked to kernel - T:2 / 20211122T153434, set=1550862636, now=1550857735
D WifiDisplayAdapter: isFitToActiveDisplayLocked : false
D WifiDisplayAdapter: isFitToActiveDisplayLocked : false
```

Рисунок 2.2 – Приклад фрагменту файлу логів Android

Розробка програмного забезпечення для мобільних пристроїв потребує врахування їх обмежень та можливостей. Мобільні пристрої працюють на акумуляторі та мають менш потужні процесори, ніж персональні комп'ютери, а також мають більше функцій, таких як визначення місцезнаходження та камери. При розробці додатків, їх тестуванні та важливо відслідковувати стан батареї для того щоб відслідкувати підвищене енергоспоживання, або перегрів внаслідок програмних помилок (Рис. 2.3)

```
Current Battery Service state:
  AC powered: false
  USB powered: true
  Wireless powered: false
  Max charging current: 0
  Max charging voltage: 0
  Charge counter: 2963664
  status: 2
  health: 2
  present: true
  level: 79
  scale: 100
  voltage: 4174
  temperature: 279
  technology: Li-ion
  batteryMiscEvent: 65536
  batteryCurrentEvent: 32768
  mSecPlugTypeSummary: 2
  LED Charging: true
  LED Low Battery: true
  current now: 291
  charge counter: 2963664
  Adaptive Fast Charging Settings: true
  Super Fast Charging Settings: true
USE_FAKE_BATTERY: false
FEATURE_WIRELESS_FAST_CHARGER_CONTROL: true
  mWasUsedWirelessFastChargerPreviously: true
  mWirelessFastChargingSettingsEnable: true
LLB CAL: 20210619
LLB MAN:
LLB CURRENT: YEAR2021M12D5
LLB DIFF: 23
SEC_FEATURE_BATTERY_FULL_CAPACITY: true
  mFullCapacityEnable: false
FEATURE_HICCUP_CONTROL: true
FEATURE_SUPPORTED_DAILY_BOARD: false
SEC_FEATURE_BATTERY_LIFE_EXTENDER: false
SEC_FEATURE_USE_WIRELESS_POWER_SHARING: true
health: vendor.samsung.hardware.health@2.0::ISehHealth@Proxy
```

Рисунок 2.3 – Приклад звіту про стан батареї

Завдяки розроблюваній програмній системі, є можливість враховувати декілька різних факторів, що виникають під час розробки та тестування мобільних додатків, в програмі поєднується декілька функцій та методів, щоб підвищити надійність та ефективність процесу відлагодження. Такий підхід буде більш привабливим для потенційних користувачів, адже він об’єднує в собі декілька різних підходів, для чого раніше вимагалось використання декількох програмних систем.

2.3 Розробка архітектури системи

Архітектура програмного забезпечення системи зображує організацію або структуру системи та надає пояснення того, як вона веде себе. Система являє собою сукупність компонентів, які виконують певну функцію або набір функцій. Іншими словами, архітектура програмного забезпечення забезпечує міцну основу, на якій можна будувати програмне забезпечення.

Серія архітектурних рішень і компромісів впливають на якість, продуктивність, ремонтопридатність і загальний успіх системи. Неврахування поширених проблем і довгострокових наслідків може поставити вашу систему під загрозу [10].

Існує безліч моделей і принципів архітектури високого рівня, які зазвичай використовуються в сучасних системах. Їх часто називають архітектурними стилями. Архітектура програмної системи рідко обмежується одним архітектурним стилем. Натомість комбінація стилів часто складає повну систему.

Архітектура програмного забезпечення відкриває структуру системи, приховуючи деталі реалізації. Архітектура також фокусується на тому, як елементи та компоненти системи взаємодіють один з одним. Розробка програмного забезпечення глибше занурюється в деталі реалізації системи. Проблеми проектування включають вибір структур даних і алгоритмів або деталей реалізації окремих компонентів.

Проблеми архітектури та дизайну часто перетинаються. Замість того, щоб використовувати жорсткі та швидкі правила для розрізнення архітектури та дизайну, має сенс поєднати їх. У деяких випадках рішення носять явно більш архітектурний характер. В інших випадках рішення зосереджені на дизайні та тому, як він допомагає реалізувати цю архітектуру.

Важливою деталлю є те, що архітектура є дизайном, але не всякий дизайн є архітектурним. На практиці архітектор – це той, хто проводить межу між архітектурою програмного забезпечення (архітектурним проектуванням) та детальним проектуванням (неархітектурним проектуванням). Немає правил чи вказівок, які б підходили для всіх випадків, хоча були спроби формалізувати розрізнення.

Шаблон проектування програмного забезпечення — це загальне рішення для багаторазового використання, що часто виникає в певному контексті в розробці програмного забезпечення. Це не готовий дизайн, який можна перетворити безпосередньо у вихідний або машинний код. Швидше, це опис або шаблон для вирішення проблеми, який можна використовувати в багатьох різних ситуаціях. Шаблони проектування — це формалізовані найкращі методи, які програміст може використовувати для вирішення поширених проблем під час проектування програми або системи. Шаблони проектування можна розглядати як структурований підхід до комп'ютерного програмування, що знаходиться на проміжному рівні між рівнями парадигми програмування та конкретним алгоритмом [11]. При підготовці до розробки програмної системи для відлагодження мобільних пристроїв під керуванням ОС Android, було розглянуто декілька патернів розробки, а саме: фабричний метод(Factory Method), будівельник(Builder) та декоратор(Decorator).

Factory Method — це породжувальний шаблон проектування, який забезпечує інтерфейс для створення об’єктів у суперкласі, але дозволяє підкласам змінювати тип об’єктів, які будуть створені.

Фабричний метод полягає у створенні об'єктів, як шаблонний метод — у реалізації алгоритму. Суперклас визначає всю стандартну та загальну поведінку

(використовуючи чисті віртуальні «заповнювачі» для кроків створення), а потім делегує деталі створення підкласам, які надає клієнт.

Factory Method робить дизайн більш адаптованим і лише трохи складнішим. Інші шаблони проектування вимагають нових класів, тоді як Factory Method вимагає лише нової операції.

Орієнтовна діаграма класу, що описує фабричний метод представлена на рисунку 2.4

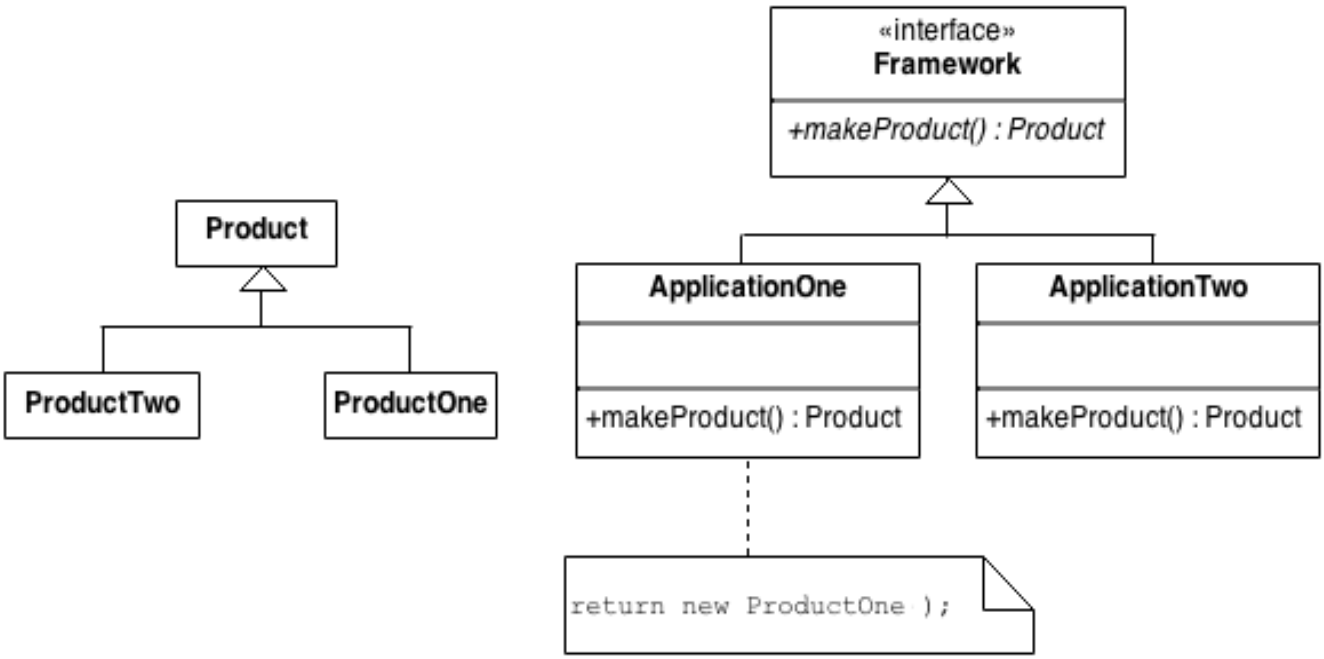

Рисунок 2.4 – Орієнтовна діаграма класу фабричного методу

Будівельник (Builder) — це породжувальний шаблон проектування, який дозволяє створювати складні об’єкти крок за кроком. Шаблон дозволяє створювати різні типи та уявлення об’єкта, використовуючи той самий код конструкції.

Шаблон Builder можна застосовувати, коли побудова різних уявлень продукту включає подібні кроки, які відрізняються лише деталями.

Інтерфейс базового конструктора визначає всі можливі етапи побудови, і бетонні будівельники реалізують ці кроки для створення конкретних уявлень продукту. Тим часом клас режисера керує порядком будівництва.

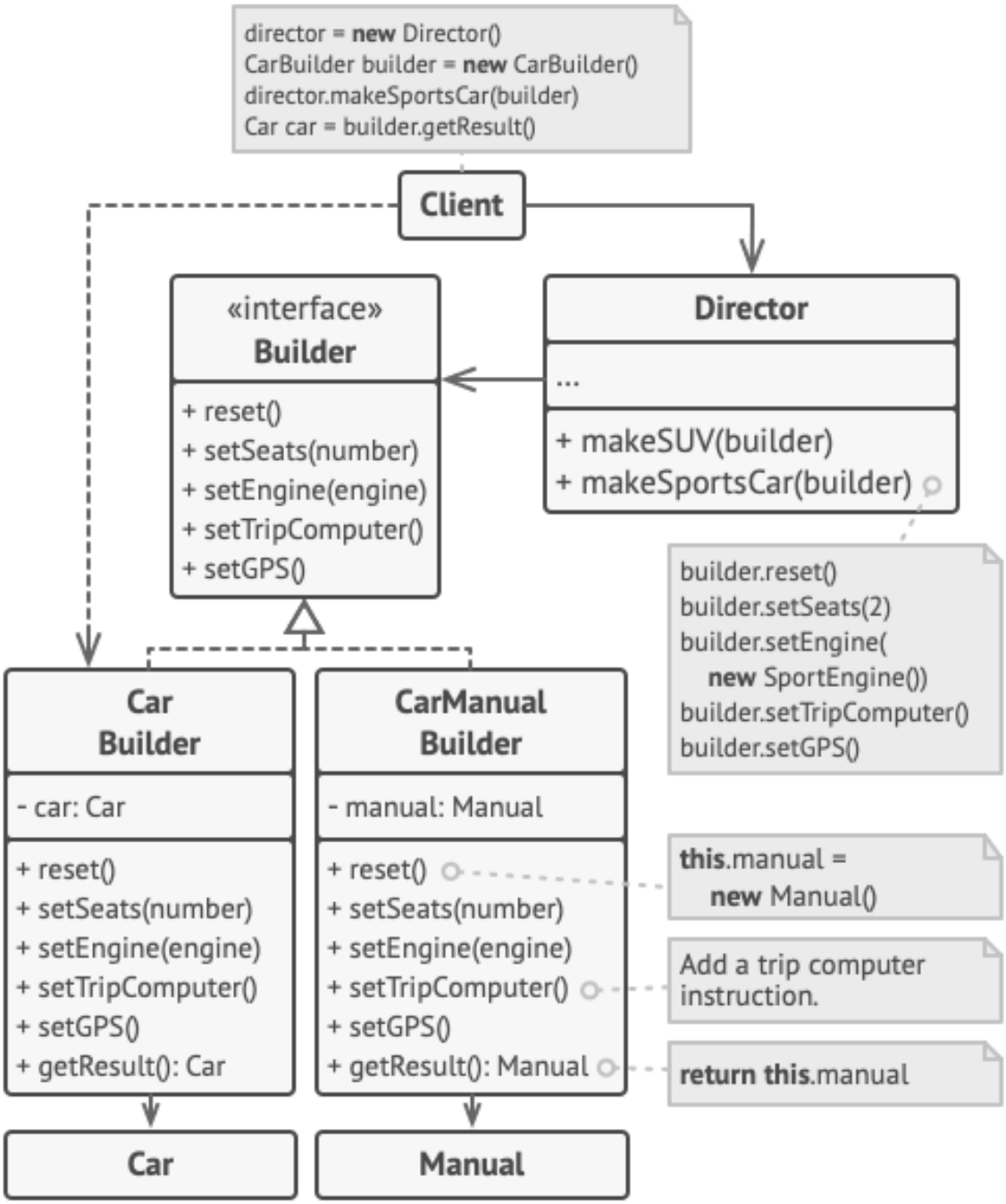

Рисунок 2.5 – Орієнтовна діаграма класу методу будівельника

Декоратор — це структурний шаблон проектування, який дає змогу приєднувати до об’єктів нову поведінку, розміщуючи ці об’єкти всередині спеціальних об’єктів-обгорток, які містять поведінку. Шаблон декоратора можна використовувати для розширення (декорування) функціональності певного об’єкта статично, або в деяких випадках під час виконання, незалежно від інших екземплярів того ж класу, за умови, що під час проектування виконано певну основу.

Цей шаблон розроблено таким чином, що кілька декораторів можуть бути накладені один на одного, щоразу додаючи нові функції до перевизначених методів (Рис 2.6).

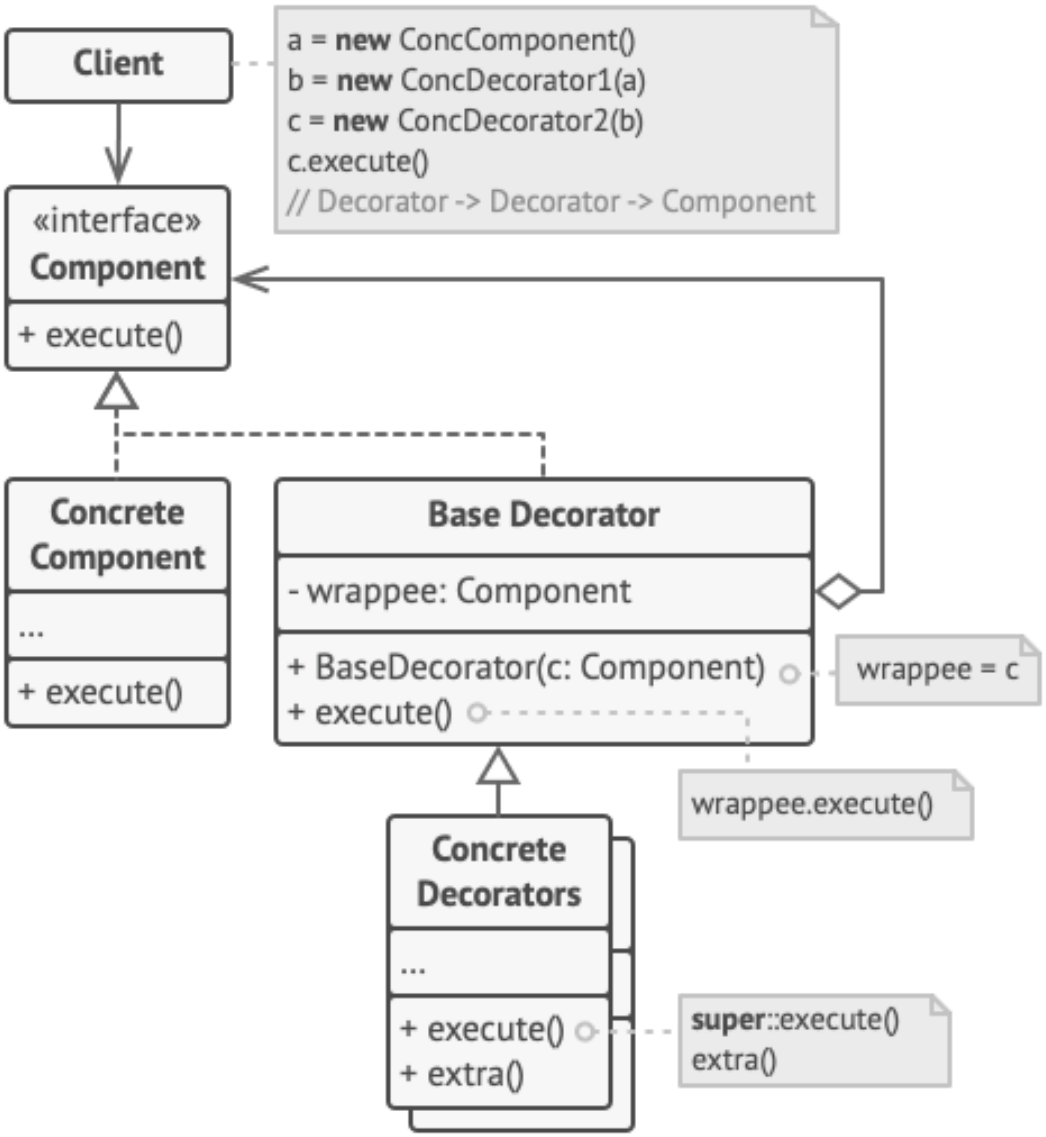

Рисунок 2.5 – Орієнтовна діаграма класу шаблону декоратора

Проаналізувавши декілька шаблонів проектування програмного забезпечення, було визначено, що для розроблюваної програмної системи для відлагодження мобільних пристроїв під керуванням операційної системи Android найкраще підходить фабричний метод. Цей шаблон було обрано, через те що він є відносно простим у імпементації, а його можливість розширювати функціонал дає змогу надалі розвивати систему з мінімальними затратами.

Для відображення взаємозв’язків між підпроцесами програмного продукту побудовано діаграму компонентів. Вона дозволяє визначити архітектуру системи, встановивши залежності між програмними модулями та компонентами. Пунктирні лінії на діаграмі представляють відношення взаємозалежностей компонентів програмного додатку. Діаграма компонентів представлена на рисунку 2.6.

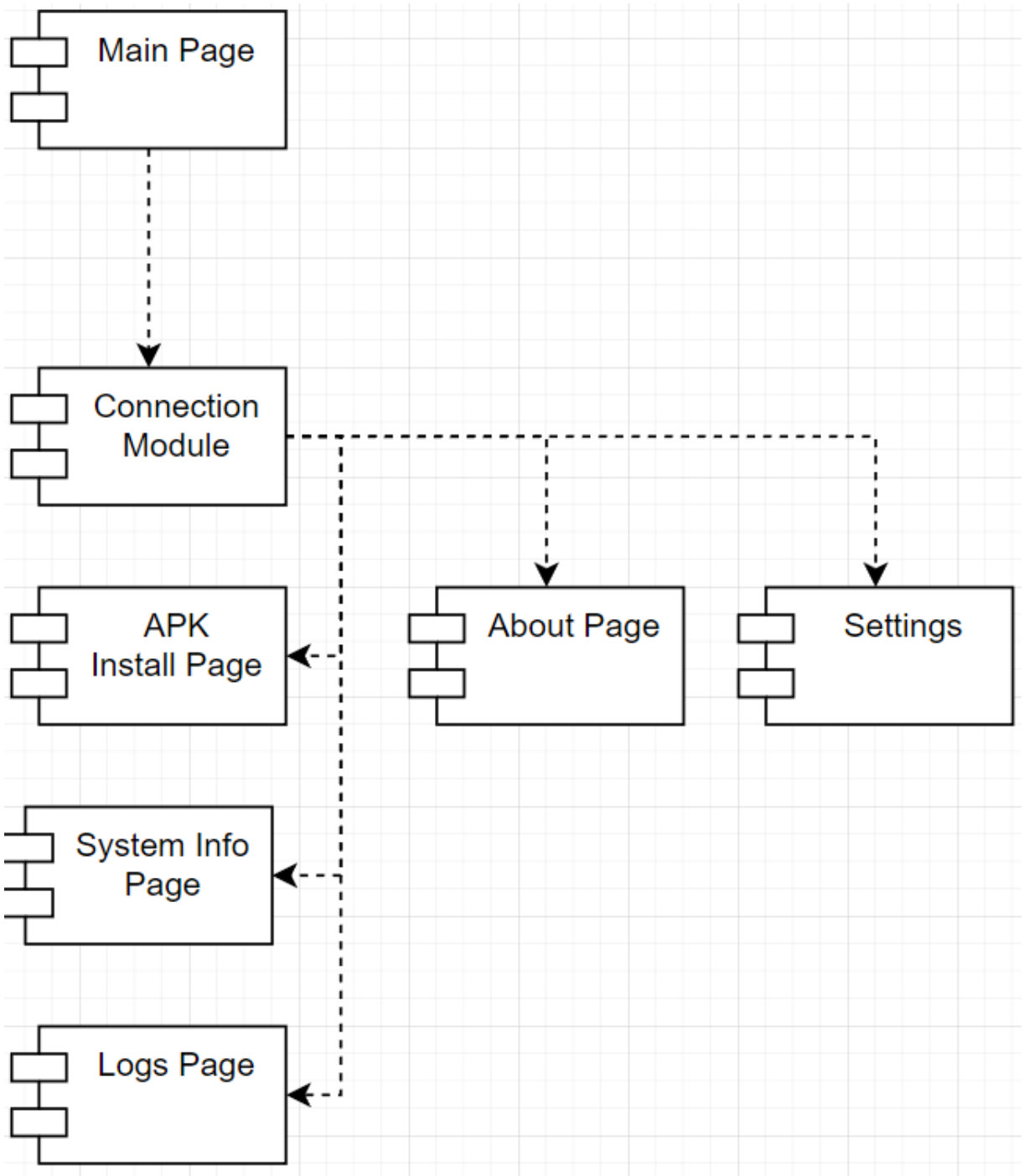

Рисунок 2.6 – Діаграма компонентів програми

В уніфікованій мові моделювання (UML) діаграма класів – це тип статичної структурної діаграми, яка надає опис системи, указуючи на класи програмної системи, їхні атрибути, операції та зв’язки між різними об'єктами.

Діаграми класів є основними будівельними блоками в об'єктно-орієнтованій моделі програмного забезпечення. Вони використовуються для створення загального моделювання структури програми. Діаграми класів можуть також використовуватися для моделювання даних. Класи на діаграмі класів представляють як основні елементи, взаємодії в додатку, так і класи, які потрібно запрограмувати. Діаграма класу програми показана на рисунку 2.7.

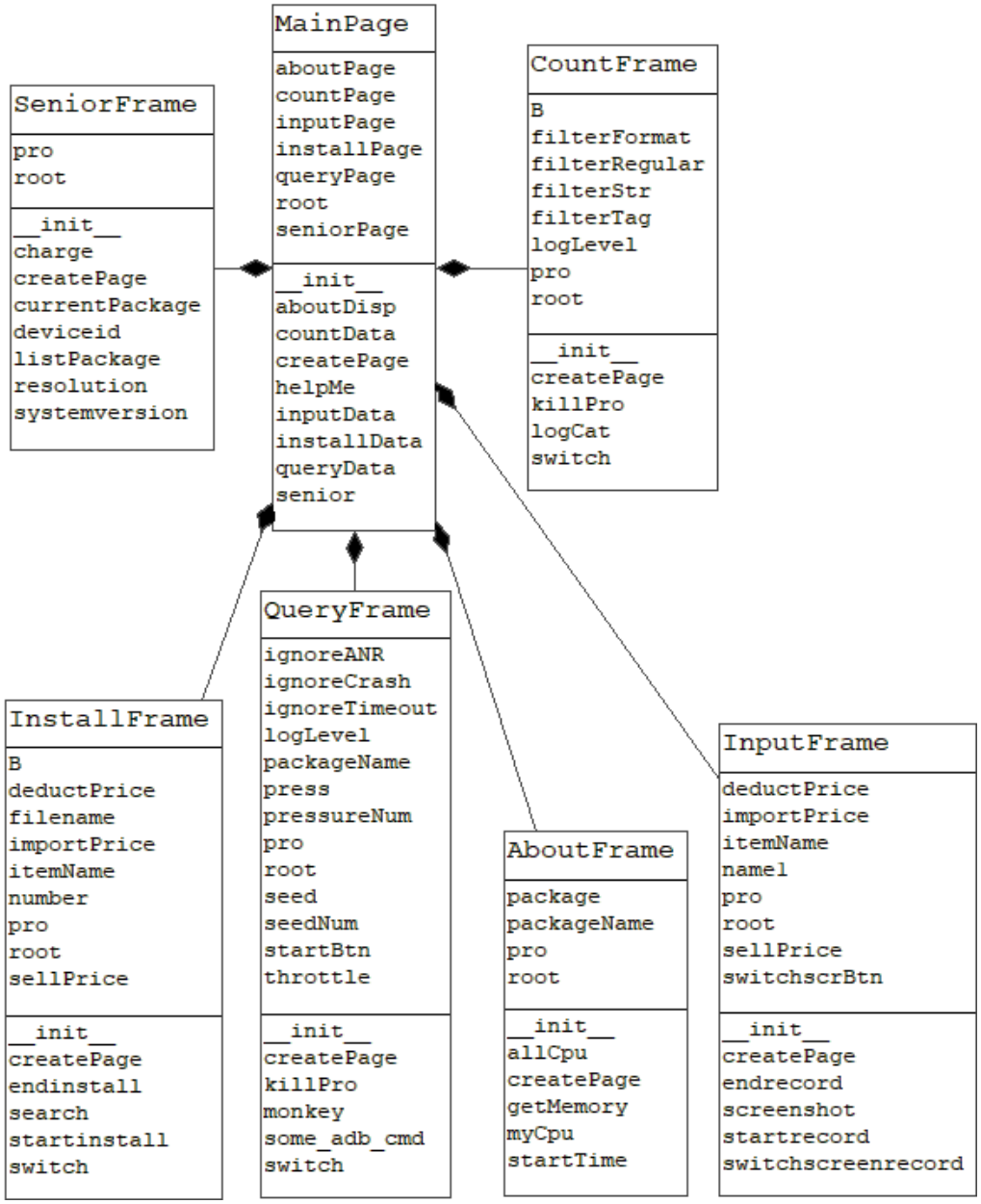

Рисунок 2.7 – Діаграма класів програми

2.4 Висновки

В даному розділі було описано методи відлагодження різних мобільних пристроїв та додатків, призначених для них. На основі описаних методів побудовано вдосконалену модель відлагодження цих пристроїв. Наведено діаграми процесу, таблиця поширеності помилок при розробці мобільних додатків та формати системних даних які повертає пристрій.

Наведено та описано схему роботи програми, у вигляді діаграм компонентів та класів, а також розглянуто архітектуру системи, обрано патерн (шаблон) проектування для розробки програмної системи.

3 РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

3.1 Обґрунтування вибору інтерфейсу

Інтерфейс користувача це сукупність засобів та правил взаємодії користувача з пристроями чи інформаційними системами. Прикладом інтерфейсу може слугувати контрольна панель верстату на виробничому підприємстві, чи так званої інтерфейс системи «оператор-верстат» [12].

Графічний візуальний інтерфейс користувача – тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, що засновані на використанні тексту, текстовому наборі команд та текстовій навігації.

Графічний інтерфейс як правило поділяють на 3 типи: Single Document Interface (одновіконний інтерфейс), Multiple Document Interface (багатовіконний інтерфейс) та Tabbed Document Interface (інтерфейс з вкладками).

Single Document Interface – тип інтерфейсу в якому не виділяються батьківські чи фонові вікна, що містять у собі елементи керування активним вікном. Програма виконується лише у одному вікні [13]. Прикладом додатку з таким типом інтерфейсу є програми з пакету Microsoft Office (рис 2.1).

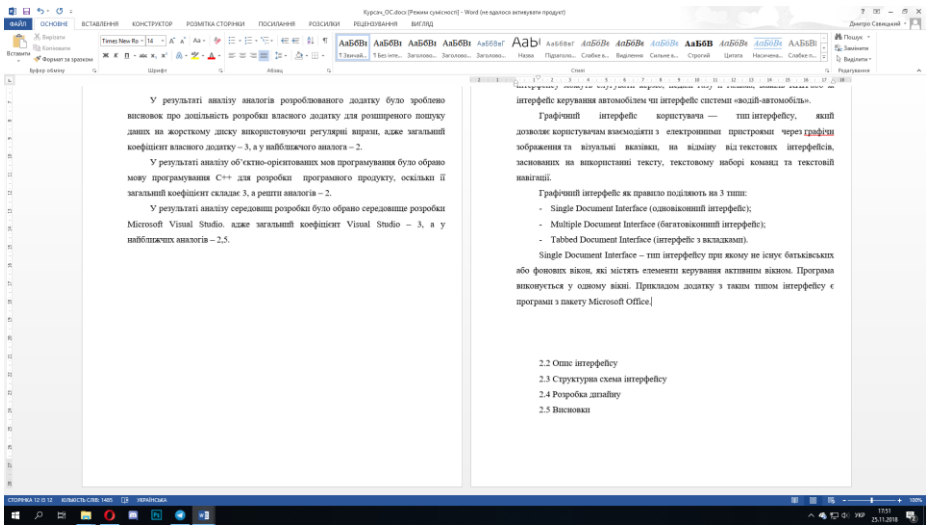

Рисунок 3.1 – Вигляд програми Microsoft Word

Multiple Document Interface містить всередині основного вікна ще декілька додаткових, а весь функціонал програмного засобу розподілений поміж ними [14]. Прикладом такої програми може слугувати засіб Adobe Photoshop, який підтримує як багатовіконний так і інтерфейс з вкладками (Рис. 2.2).

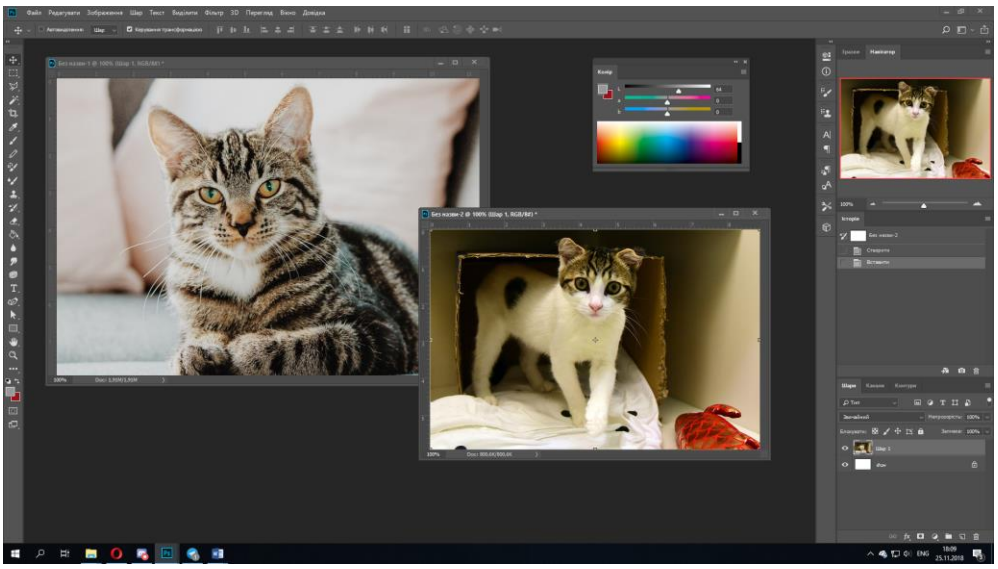

Рисунок 3.2 – Вигляд програми Adobe Photoshop CC 2018

Tabbed Document Interface – спосіб організації графічного інтерфейсу користувача, при якому кожен активний документ відображається на окремій вкладці головного вікна. Такий інтерфейс як правило використовують при розробці веб-браузерів, як наприклад Google Chrome (Рис. 2.3).

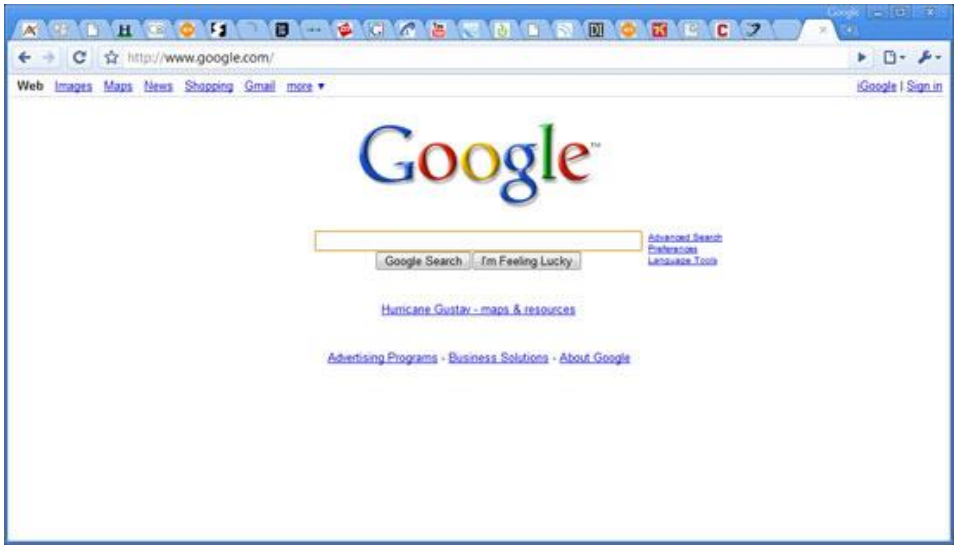

Рисунок 3.3 – Інтерфейс Google Chrome

Враховуючи результати аналізу декількох типів графічного користувацького графічного інтерфейсу та виходячи з очікуваного функціоналу розроблюваного додатку було вирішено використовувати табуйоване компонування інтерфейсу (Tabbed Document Interface)

3.2 Опис інтерфейсу

Графічний інтерфейс користувача складається з головного вікна, вікна опису програми та вікна налаштувань процесу відлагодження пристрою. Головне вікно складається із заголовку програми, панелі інструментів, робочої області, шкали прогресу та рядка статусу.

Робоча область включає в себе поля вводу для запиту та фільтрів, кнопки керування поле прогресу поточної операції та поле виводу результатів пошуку.

Рядок статусу містить в собі інформацію про результати процесу відлагодження а також дає підказки користувачеві.

Меню містить такі пункти:

- Файл(«Почати відлагодження», «Вихід з програми»);
- Налаштування(«Налаштування»);
- Про додаток(«Про додаток»).

Кнопка «Почати відлагодження» розпочинає процедуру відображення логів мобільного пристрою.

Кнопка «Вихід з програми» завершує всі процеси системи процес та закриває вікно додатку.

Кнопка «Налаштування» відкриває вікно з налаштуваннями процесу відлагодження.

Кнопка «Про додаток» відкриє нове вікно, в якому буде відображена інформація про додаток.

На рисунку 2.4 представлена графічна схема головного вікна додатку.

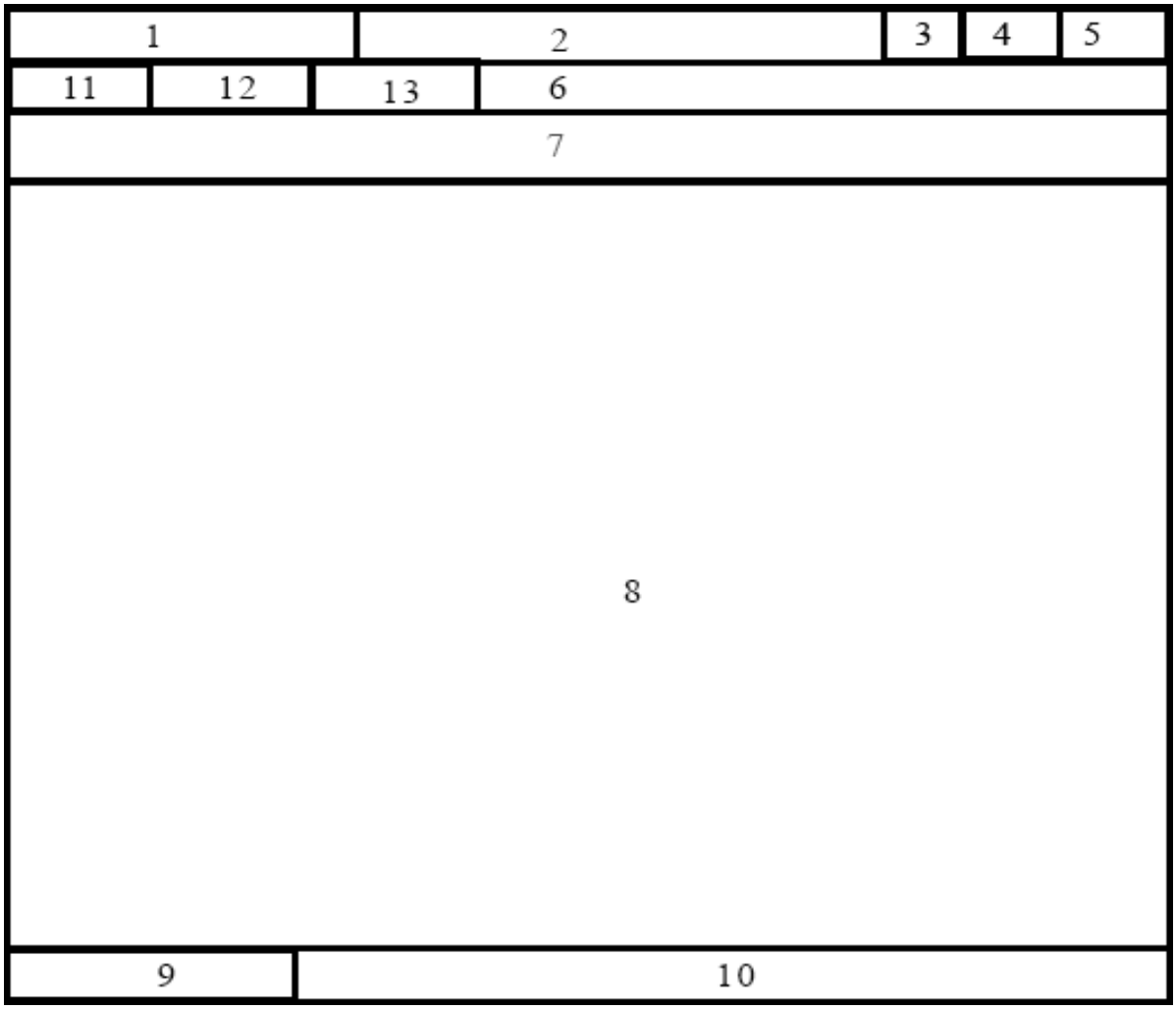

Рисунок 3.4 – Графічна схема головного вікна додатку

1. Іконка програми;
2. Назва програми;
3. Кнопка мінімізації програми;
4. Кнопка максимізації програми;
5. Кнопка закриття програми;
6. Вкладки меню;
7. Панель керування;
8. Робоча частина програми;
9. Інформація про результати відлагодження;
10. Рядок статусу.
11. Пункт меню «Файл»;
12. Пункт меню «Налаштування»;

13. Пункт меню «Про додаток».

На рисунку 2.5 зображена модульна схема головного вікна додатку.

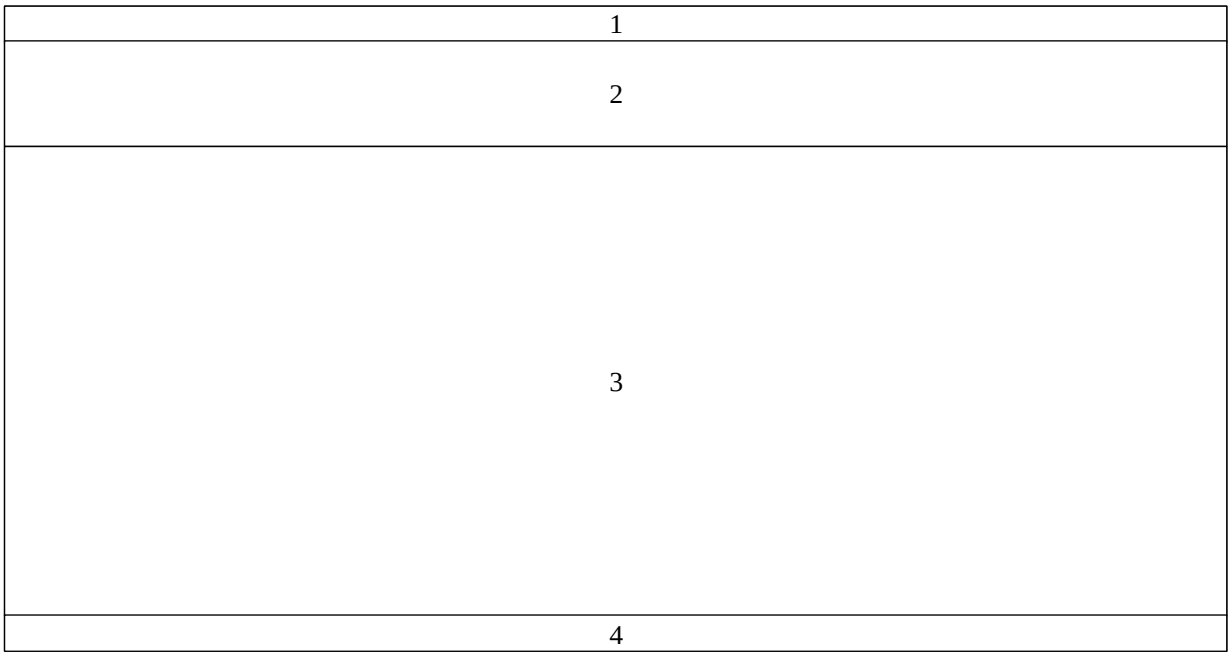

Рисунок 3.5 – Модульна схема головного вікна додатку

1. Заголовок вікна;
2. Панель керування;
3. Робоча область;
4. Рядок статусу.

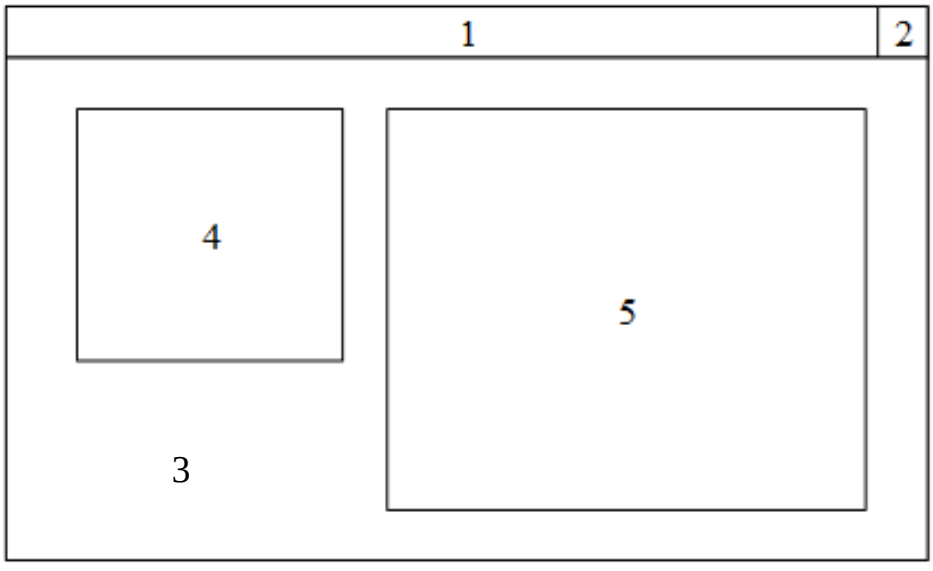

Рисунок 3.6 – Графічна схема вікна «Про додаток»

Вікно «Про додаток» складається з іконки програми та детальної інформації про додаток.

Графічна схема вікна «Про додаток» наведена на рисунку 2.6.

1. Назва вікна;
2. Кнопка закриття вікна;
3. Робоча область вікна;
4. Іконка програми;
5. Інформація про програму.

Вікно налаштувань містить елементи керування певними атрибутами пошуку, наприклад ввімкнення чи вимкнення функції використання регулярних виразів в процесі пошуку файлів.

Графічна схема вікна «Налаштування» наведена на рисунку 2.7.

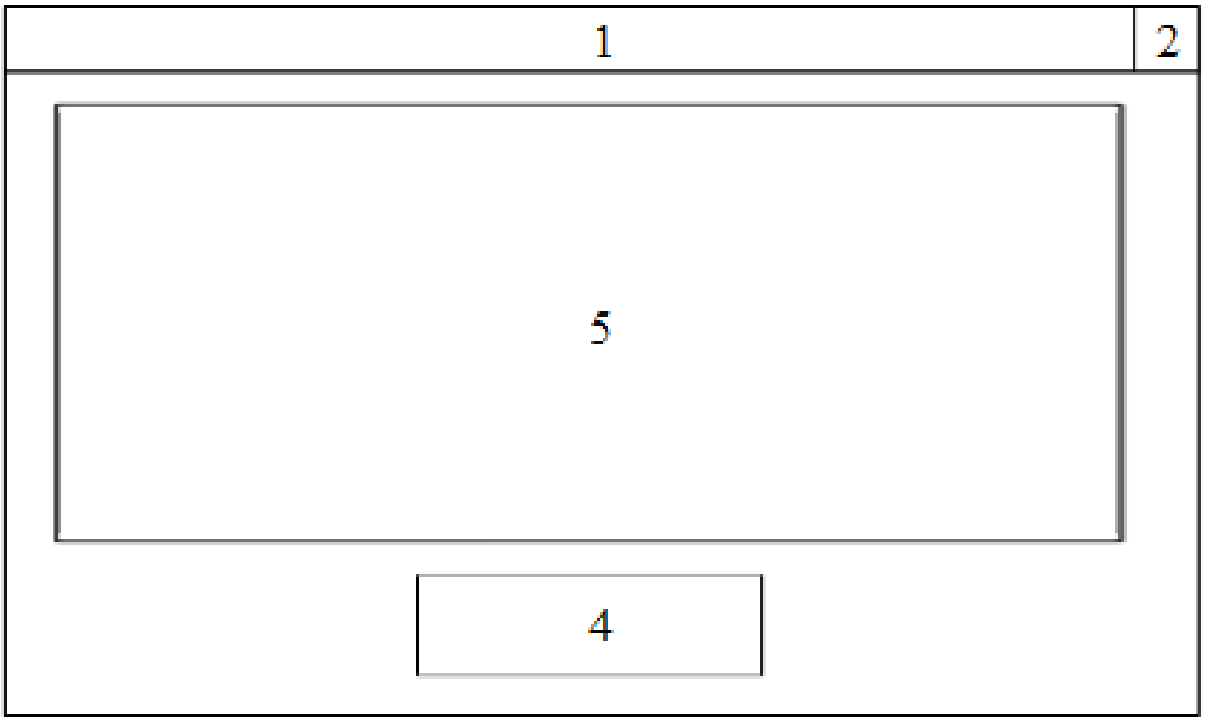

Рисунок 3.7 – Графічна схема вікна «Налаштування»

1. Назва вікна;
2. Кнопка закриття вікна;
3. Робоча область вікна;
4. Кнопка «Зберегти»;

3.3 Розробка інтерфейсу програми

Додаток складається з 3 вікон:

1. Головне вікно. Містить основні елементи керування процесом відлагодження пристрою.
2. Про додаток. Містить детальну про додаток.
3. Налаштування. Вікно містить елементи керування певними атрибутами програмної системи.

Структурну схему інтерфейсу наведено на рисунку 2.8.

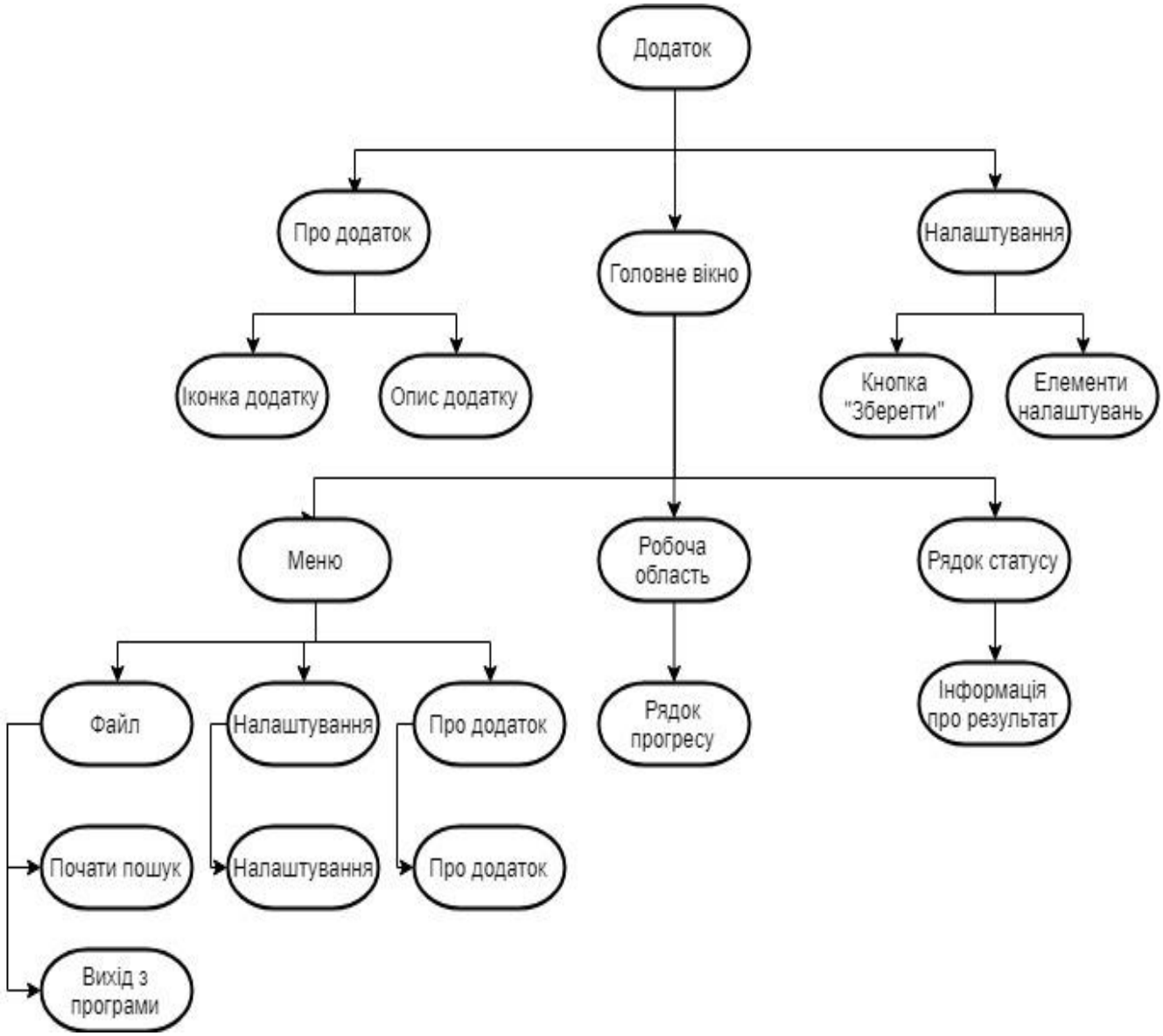

Рисунок 3.8 – Структурна схема інтерфейсу

Windows Aero – це художній стиль а також комплекс технічних рішень побудови інтерфейсу, вживаний в операційних системах Microsoft Windows починаючи з Windows Vista. Назва є акронімом з англійської: «Authentic, Energetic, Reflective and Open».

В Windows 8 вперше було представлено дизайн Metro який прийшов на зміну Windows Aero. Основною його особливістю є плиткова структура та відсутність великої кількості градієнтів як у Aero [15].

Нині більшість додатків використовує саме стиль Metro (рис 2.9), оскільки саме він використаний в останніх версіях ОС Windows, хоча деякі з тих чи інших причин продовжують використовувати уже морально застарілий дизайн Windows Aero.

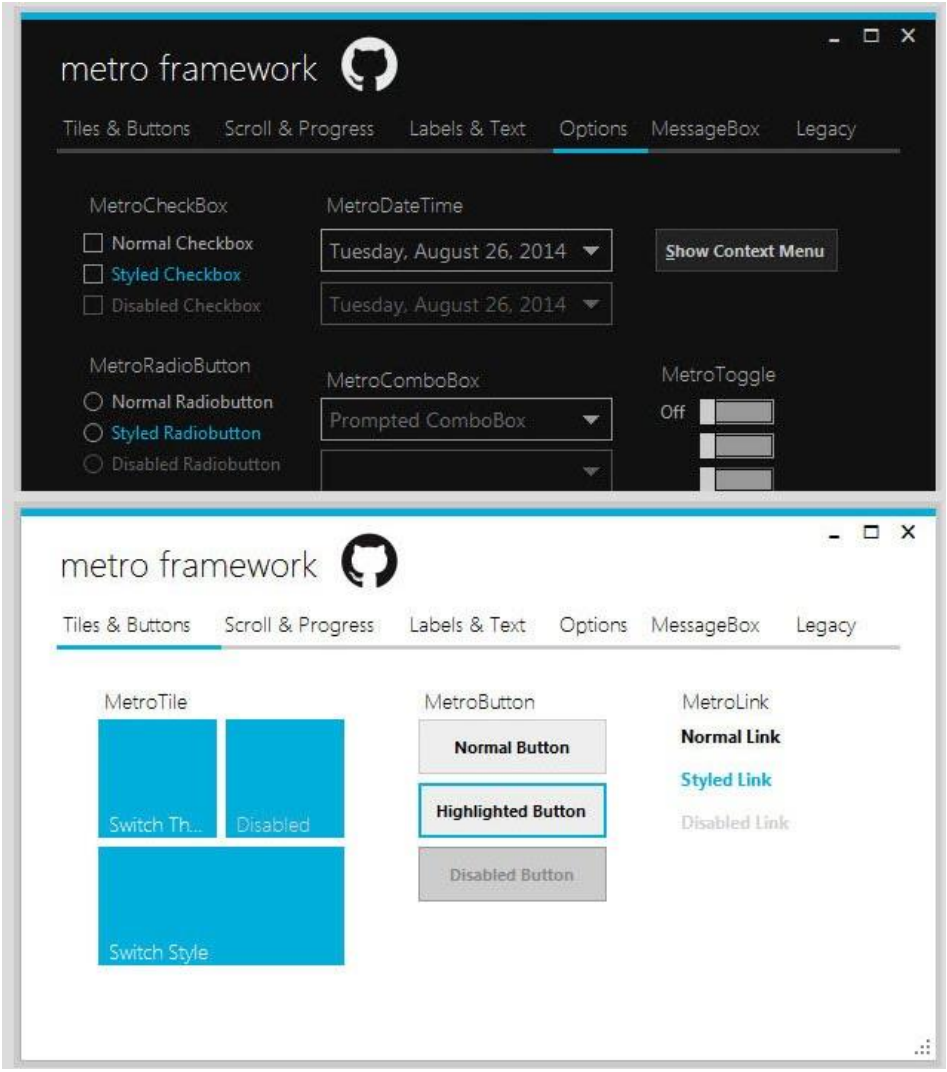

Рисунок 3.9 – Приклад додатку в стилі Metro у світлій та темній темах

Для розробки додатку відлагодження пристроїв під керування операційної системи Android було обрано стиль інтерфейсу Metro, адже останні актуальні версії ОС Windows виконані саме у ньому.

Для технічної реалізації графічного інтерфейсу користувача було обрано бібліотеку Tkinter.

Tkinter – крос-пратформова бібліотека для створення графічних інтерфейсів, що поставляється в комплекті разом з інтерпретатором мови програмування Python, і є найбільш поширеною системою для розробки графічних інтерфейсів для цієї мови з підтримкою більшості сучасних операційних систем (Microsoft Windows, Linux та MacOS)

Tkinter вирізняється серед інших інструментів для створення графічних інтерфейсів своєю простотою у освоєнні і послідує основним фундаменам мови програмування Python – лаконічність та читабельність програмного коду. Інтерфейс написаний з допомогою Tkinter складається з віджетів – будівельних блоків, з яких складається весь інтерфейс [16]. Враховуючи простоту та легкість бібліотеки Tkinter, вона є оптимальним вибором при розробці крос-платформових додатків мовою програмування Python. Макет графічного інтерфейсу додатку наведено на рисунку 2.10.

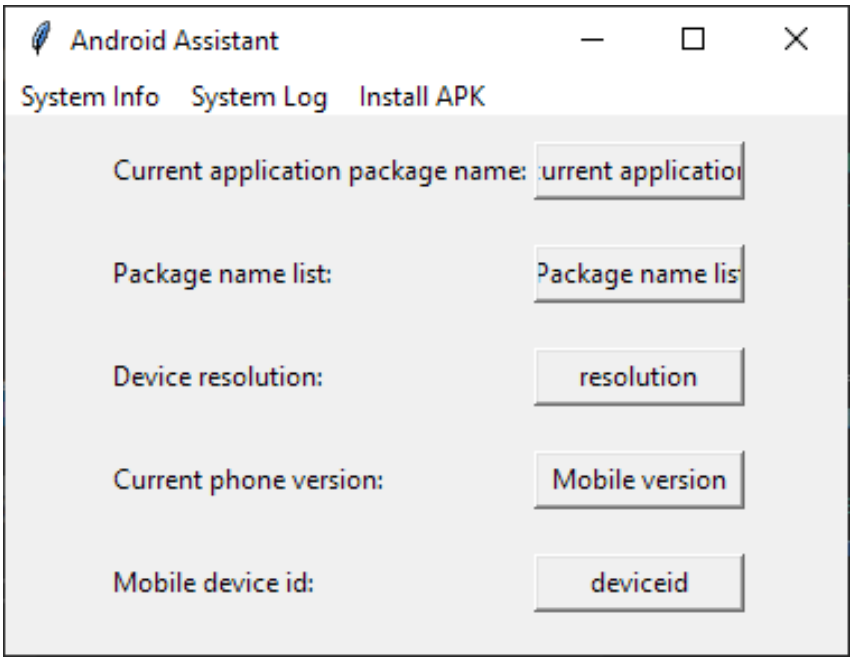

Рисунок 3.10 - Макет графічного інтерфейсу додатку

3.4 Блок-схема і структурна схема додатку

Алгоритм – набір певних інструкцій, що описують порядок виконування дій для досягнення результату в рішенні поставленої проблеми за скінченну кількість кроків. Алгоритми є порядком правил виконування процесу, який досягає поставленої цілі за визначений скінченний час. Для візуалізації алгоритмів як правило використовують блок-схеми.

При розробці програмних додатків алгоритми є переліком деталізованих інструкцій, що реалізують процес обрахунку, який відбувається через визначену послідовність логічних станів, що закінчується кінцевим станом системи. Перехід від одного до наступного стану не є обов'язково визначеним (детермінованим) — певні алгоритми можуть містити в собі елементи випадковості.

Поняття алгоритму належить до засад математичної науки. Процеси обчислення алгоритмічного характеру, як наприклад арифметичні дії над цілими числами, знаходження найбільшого спільного дільника або найменшого спільного діленого (НСД) двох чисел, тощо, були відомі людству ще з давніх часів. Проте, чітке визначене поняття алгоритму було сформоване лише на початку XX століття [17].

Кожен алгоритм передбачає існування вхідних (початкових) даних та в результаті роботи призводить до певного очікуваного результату. Робота кожного взятого алгоритму відбувається шляхом виконання певної послідовності деяких визначених елементарних дій. Ці дії називаються кроками, а процес їх виконання – алгоритмічним процесом. Таким чином відзначають властивість дискретності алгоритму.

Важливою властивістю алгоритмів є масовість, тобто можливість застосування до різних вхідних даних. Це означає, що кожен алгоритм повинен розв'язувати перелік задач одного типу.

Необхідною умовою, що задовольняє алгоритм, є визначеність, або детермінованість. Тобто що виконання алгоритму відбувається у один спосіб та призводить до однакового результату для одних і тих самих вхідних даних.

Загальний алгоритм роботи додатку:

- Ініціалізація головного вікна та його елементів;
- Здійснення пошуку файлів за заданим запитом;
- Виведення результату.

Загальна блок-схема додатку зображена на рисунку 3.1.

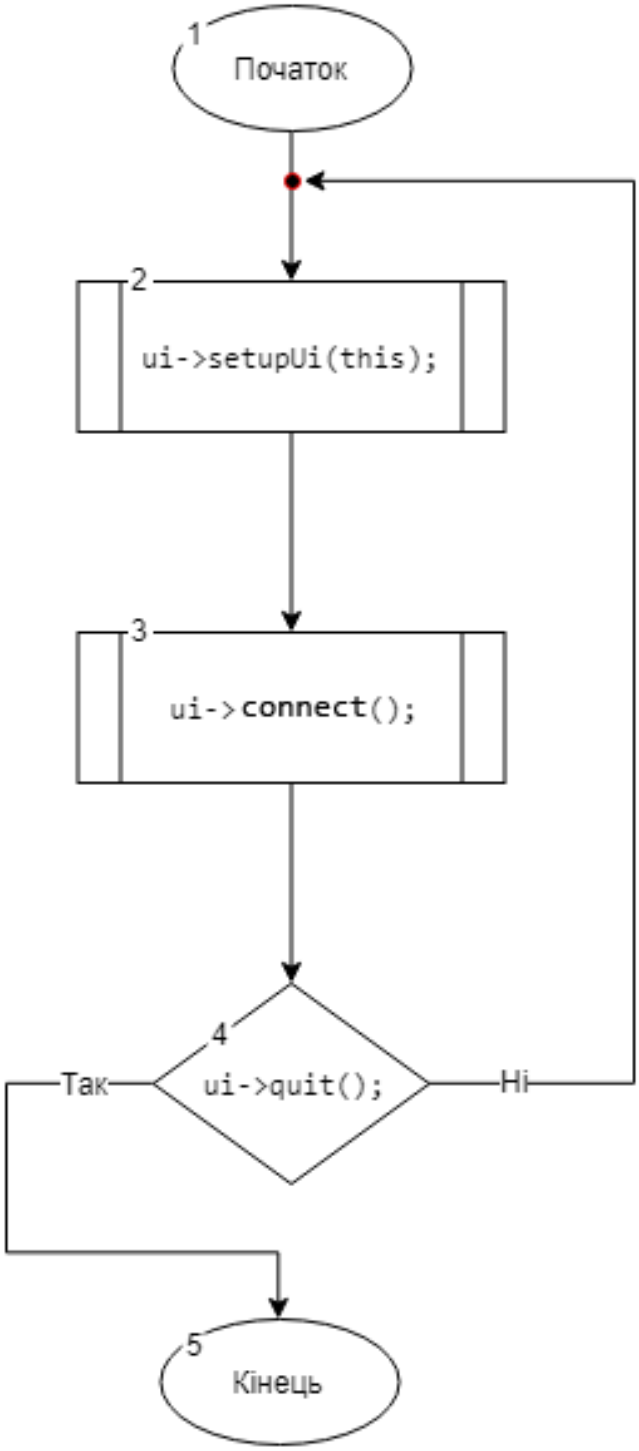

Рисунок 3.11 – Загальна блок-схема додатку

На рисунку 3.2 представлена блок-схема модуля програми, що відповідає за процес відлагодження та читання логів.

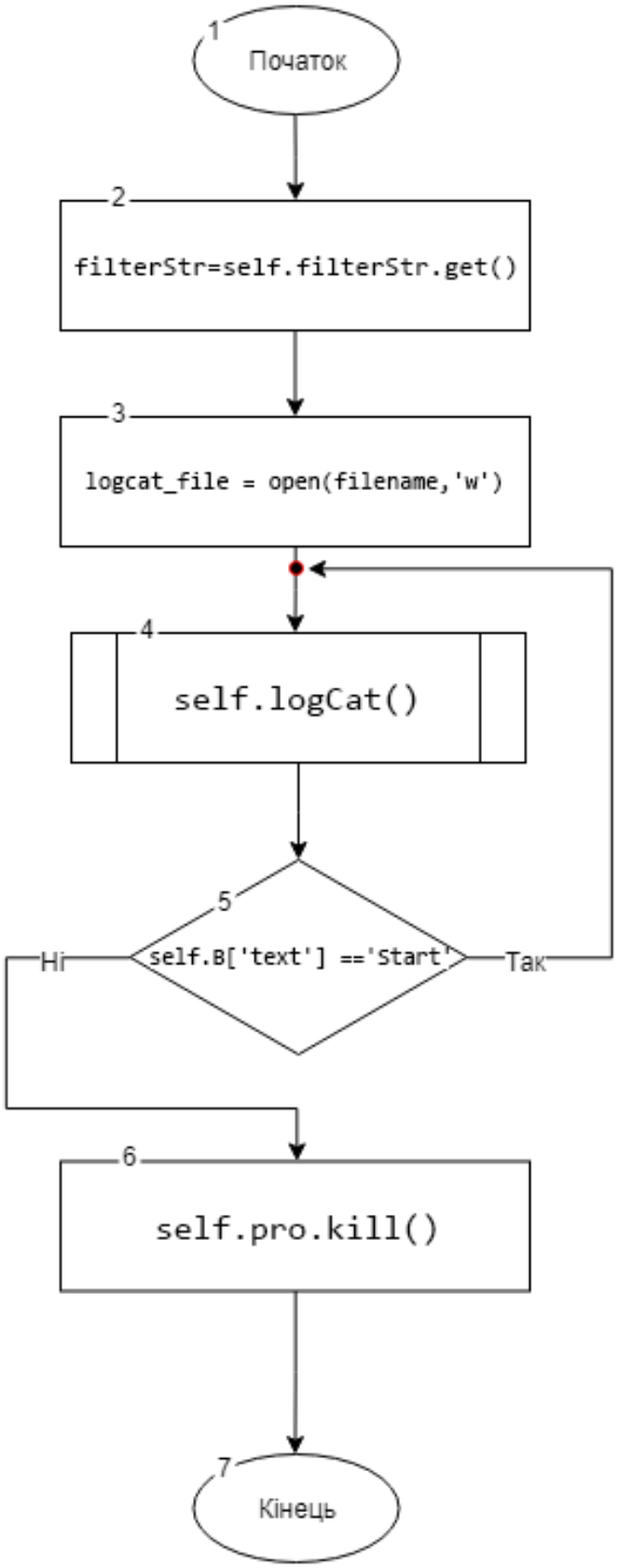

Рисунок 3.12 – Загальна блок-схема модуля відлагодження

3.5 Розробка алгоритму роботи програмного додатку і базових модулів

Додаток розроблено з використанням технології tkinter, крос-платформового інструментарію розробки програмного забезпечення мовою програмування Python. Дана бібліотека дозволяє запускати написане за його допомогою ПЗ на більшості сучасних операційних систем, просто компілюючи текст програми для кожної операційної системи без зміни вихідного коду. Пакет містить всі основні класи, які можуть бути потрібні для розробки прикладного чи сервісного програмного забезпечення, починаючи з елементів графічного інтерфейсу й закінчуючи класами для роботи з мережею, базами даних, OpenGL, SVG і XML. Бібліотека дозволяє керувати нитками, працювати з мережею та забезпечує крос-платформовий доступ до файлової системи.

Кожне вікно програми має свої події (наприклад натискання кнопки в межах вікна, чи натискання певної клавіші на клавіатурі). Через подію програма викликає певні методи програми. Наприклад, подія початку процесу запису логі викликає метод logcat(), що оцінює орієнтовну кількість файлів в теках для правильної роботи індикатору прогресу.

При запуску додатку створюється головне вікно та ініціалізуються основні методи програмного додатку. Далі додаток обробляє введений користувачем запит і здійснює пошук, виводячи результат(перелік тек та файлів) у відповідне поле головного віка.

В уніфікованій мові моделювання (UML) діаграма класів – це тип статичної структурної діаграми, що описує систему, вказуючи на класи системи, їх атрибути, операції та зв’язки між об'єктами.

Діаграми класів є основними будівельними блоками об'єктно-орієнтованого моделювання програмного забезпечення. Вони використовуються для загального моделювання структури програми. Діаграми класів можуть також використовуватися для моделювання даних. Класи на діаграмі класів представляють як основні елементи, взаємодії в додатку, так і класи, які потрібно запрограмувати [18]. Діаграма класів додатку показана на рисунку 3.13.

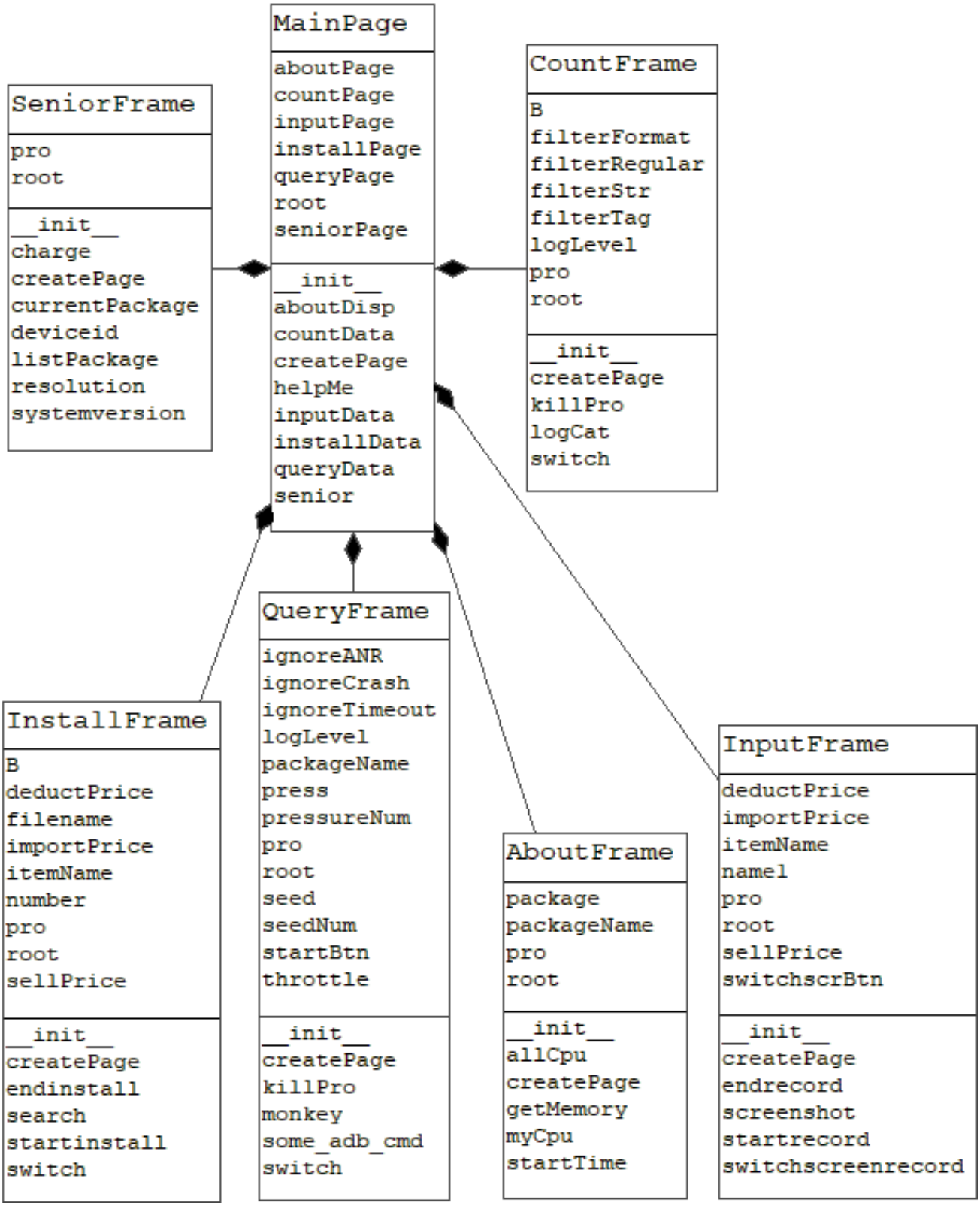

Рисунок 3.13 – Діаграма класів додатку

Робота програми починається з вікна ініціалізації, де користувач системи має під’єднати мобільний пристрій за допомогою кабелю USB. Додаток перевіряє стан пристрою, та після успішного зв’язку користувач бачить екран системних

даних пристрою, або повідомлення про помилку, якщо з’єднання не відбулось з певних причин (наприклад, телефон не під’єднано кабелем) Нижче наведено програмний код модуля ініціалізації пристрою.

```
    def createPage(self):
            self.page = Frame(self.root)
            self.page.pack()
            Button(self.page,    text='Connect    to    phone',width=12,
command=self.loginCheck).grid(row=1, stick=W, pady=10,column=1)
            Button(self.page,                        text='Quit',width=12,
command=self.page.quit).grid(row=2, column=1, stick=E)
        def loginCheck(self):
            out = subprocess.getstatusoutput('adb devices -l')
            out1 = subprocess.getstatusoutput('adb devices')
            if out[0]==0:
                if'device product' in out[1]:
                    self.page.destroy()
                    MainPage(self.root)
                else:
                    self.sayTry()
            else:
                if out1[0]==0:
                    if'device' in out1[1]:
                        self.page.destroy()
                        MainPage(self.root)
                else:
                    self.sayNoadb()
        def connectPhone(self):
            self.page.destroy()
        def sayTry(self):
            tk.messagebox.showinfo("Message",  "Phone  connection  failed,
please try to reconnect")
        def sayFail(self):
```

Рисунок 3.14 – Код модуля ініцілізації

Після успішного з'єднання, користувач бачить екран системних даних пристрою, таких як: версія операційної системи, розмір екрану, серійний номер, дізнатись стан заряду батареї, тощо. Програмний код сторінки модуля представлено нижче.

```
def __init__(self, master=None):
        Frame.__init__(self, master)
        self.root = master # Define internal variable root
        self.createPage()

    def createPage(self):
        Label(self, text='Android version:').grid(row=0, sticky=W, pady=10)
        Label(self, text=self.systemversion()).grid(row=0, column = 1,
sticky=W, pady=10)
        Button(self, text='Current package', width=12,
command=self.currentPackage).grid(row=4 ,sticky=W+E, pady=10, column = 0,
columnspan=2)
        Button(self, text='Package name list', width=12,
command=self.listPackage).grid(row=3, sticky=W+E, pady=10, column = 0,
columnspan=2)
        Label(self, text='Device resolution:').grid(row=2, stick=W,
pady=10)
        Label(self, text=self.resolution()).grid(row=2, column = 1,
sticky=W, pady=10)
        Label(self, text='Device id:').grid(row=1, stick=W, pady=10)
        Label(self, text=self.deviceid()).grid(row=1, column = 1, sticky=W,
pady=10)
Frame.__init__(self, master)
self.root = master # Define internal variable root
self.createPage()
Button(self, text='Package name list', width=12,
command=self.listPackage).grid(row=3, sticky=W+E, pady=10, column = 0,
columnspan=2)
```

Рисунок 3.15 – Код модуля системних даних

Одним з інструментів відлагодження є перевірка стану батареї мобільного пристрою. Програма має здатність читати наступні дані: відсоток заряду, стан підключення, тип зарядки (стандартна, швидка чи бездротова зарядка), температура батареї, вольтаж, загальний стан батареї та інші системні дані. Код модуля перевірки стану заряду наведено нижче.

```
    def resolution(self):
            out = subprocess.getstatusoutput('adb shell wm size')
            return format(out[1])

        def systemversion(self):
            out    =    subprocess.getstatusoutput('adb    shell    getprop
ro.build.version.release')
            return format(out[1])

        def deviceid(self):
            out = subprocess.getstatusoutput('adb get-serialno')
            return format(out[1])

        def charge(self):
            print('battery information')
            out = subprocess.getstatusoutput('adb shell dumpsys battery')
            top = tk.Toplevel()
            top.title('battery information')

            top.geometry('%dx%d'% (600, 600))
            t = Text(top, width=600, height=600)
            t.insert('1.0', "{}".format(out[1]))
            t.pack()
            top.mainloop()
    nowtime = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
            filename ='package'+nowtime + ".txt"
            logcat_file = open(filename,'w')
```

Рисунок 3.16 – Код модуля логування

Приклад формату виведеного стану батареї мобільного пристрою представлено на рисунку 3.17.

Рисунок 3.17 – Приклад формату стану батареї

Ще одним з розроблених модулів є можливість віддаленого встановлення на пристрій пакетів .apk. Файл пакету Android (Android Package File, APK) — це файл, який використовується для розповсюдження програм в операційній системі

Google Android. Файл APK містить весь код програми (наприклад, файли .dex), ресурси, активи, сертифікати та файл маніфесту додатку. Нижче представлено код модуля встановлення додатків.

```
    def __init__(self, master=None):
            Frame.__init__(self, master)
            self.root = master # Define internal variable root
            self.itemName = StringVar()
            self.importPrice = StringVar()
            self.sellPrice = StringVar()
            self.deductPrice = StringVar()
            self.number = StringVar()
            self.createPage()

        def createPage(self):
            Label(self).grid(row=0, stick=W, pady=10)
            Button(self, text='Find installation file', width=25,
height=5, command=self.search).grid(row=1, stick=W, pady=10)
        def search(self):
            default_dir = r"C:\Users\lenovo\Desktop" # Set the default
open directory
            self.filename =
tkinter.filedialog.askopenfilename(title=u"Select
File",initialdir=(os.path.expanduser(default_dir)),filetypes=[("apk
format", "apk")])
            Label(self,text='The path
is:').grid(row=2,stick=W,pady=10)
            Label(self, text=self.filename).grid(row=3, pady=10,
stick=W)
            self.B = Button(self, text='Start installation', width=12,
command=self.switch)
            self.B.grid(row=6, stick=E, pady=10)
```

Рисунок 3.18 – Код модуля інсталяції пакетів

3.6 Висновки

У ході аналізу типів інтерфейсу для розробки графічного інтерфейсу додатку було прийнято рішення використовувати табуйований тип інтерфейсу. Стилем графічного інтерфейсу було обрано комплекс рішень Metro.

Розроблено структурні та графічні схеми інтерфейсу, описано функції та призначення кожного елемента інтерфейсу.

Було розроблено алгоритми роботи компонентів програмного продукту.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Методики тестування

Тестування програмного забезпечення – процес технологічного дослідження, призначений для визначення інформації про якість програмного продукту. Тестування є невід’ємною складовою при розробці будь-якого виду програмного забезпечення, та часто витрачає до половини загальних ресурсів виділених на процес розробки програмних продуктів [19].

При тестуванні будь-якого застосунку складається тестовий набір або так званий тест-план. Тестовий набір складається з окремих тестів і розробляється таким чином, щоб забезпечити якомога більше покриття множини ймовірних впливів на об'єкт тестування. При тестуванні програмного додатку використовують велику кількість різних методик, в тому числі такі як «біла скринька» та «чорна скринька».

Тестування за допомогою методики «білої скриньки» полягає в знанні особою, що проводить тестування продукту всіх внутрішніх процесів програмного продукту та його повну внутрішню структуру. Це тестування засновано на детальному аналізі керуючих структур даних та алгоритмів програмного продукту [20]. Програма вважається повністю перевіреною, якщо проведено тестування всіх маршрутів її графа управління.

Для тестування формуються такі тестові варіанти, в яких:

- гарантується перевірка максимальної кількості незалежних маршрутів програми;
- знаходяться гілки True та False для всіх логічних рішень;
- виконуються всі можливі цикли (у межах їхніх кордонів та діапазонів);
- аналізується правильність внутрішніх структур даних.

Зазвичай тестування поділяється на три типи: функціональне тестування, нефункціональне тестування або тестування продуктивності та тестування пов’язане зі змінами (Рис. 4.1).

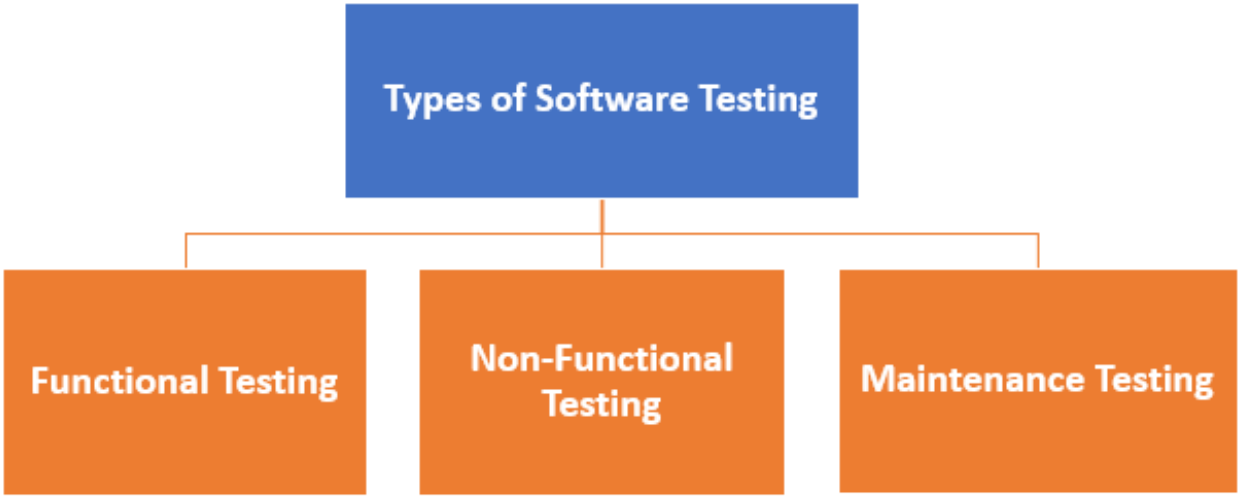

Рисунок 4.1 – Типи тестування

При тестуванні програмного додатку за різними методиками, процес тестування також поділяють на методи «білої скриньки» та «чорної скринька».

При використанні методики «чорної скриньки» не використовуються будь-які знання про внутрішню структуру. При тестуванні ігнорується внутрішня логічна структура функцій, має значення лише відповідність вхідних та вихідних даних [21]. Тест методики демонструють:

- як виконуються функції програми;
- як приймаються вихідні дані;
- як отримуються результати;

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

- некоректних чи відсутніх функцій;
- помилок інтерфейсу;
- помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;
- помилок характеристик;
- помилок ініціалізації та завершення.

У результаті порівняльного аналізу методик тестування було обрано методику «чорної скриньки» для тестування програмного додатку.

Для тестування розробленого програмного продукту було розроблено тест-план, що містить у собі тест-кейси, які мають покрити практично увесь існуючий в програмі функціонал. Таблицю тест-кейсів програми «SuperDebug» наведено у таблиці 4.1

4.2 Інструкція тестування програмного додатку

Методом «чорної скриньки» було проведено тестування усього наявного функціоналу програмного додатку. Тест-кейси та результати їх виконання наведено у таблиці 4.1.

Таблиця 4.1 – Варіанти тестування програмного додатку

№	Кроки	Вхідні дані	Вихідні дані	Результат
1	- Під’єднати пристрій за допомогою кабеля - Натиснути “Connect”	Мобільний пристрій	Головне вікно відлагодження	Відповідає
2	- Не під’єднувати пристрій за допомогою кабеля - Натиснути “Connect”	Мобільний пристрій	Повідомлення про помилку	Відповідає
3	- Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перевірити системні дані пристрою (версія ОС, серійний номер)	Системні дані пристрою	Коректні дані	Відповідає
4	Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перевірити стан батареї пристрою	Мобільний пристрій	Вікно стану батареї	Відповідає

Продовження таблиці 4.1

№	Кроки	Вхідні дані	Вихідні дані	Результат
5	Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перевірити список встановлених пакетів додатків	Мобільний пристрій	Список встановлених пакетів додатків	Відповідає
6	Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перевірити поточний відритий пакет додатку	- Мобільний пристрій - Додаток	Назва поточного відкритого пакету	Відповідає
7	- Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перейти до вкладки “System Logs”	Натискання вкладки	Вкладка логування	Відповідає
8	- Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перейти до вкладки “System Logs” - Виставити фільтри - Натиснути “Start” - Перевірити файл логів пристрою	Мобільний пристрій	Файл логів пристрою	Відповідає
9	- Натиснути кнопку «Про програму» або клавішу F1	Натискання клавіші	Вікно відомостей про програму	Відповідає

Продовження таблиці 4.1

№	Кроки	Вхідні дані	Вихідні дані	Результат
10	- Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перейти до вкладки “System Logs” - Залишити поле фільтрів порожнім - Натиснути “Start” - Перевірити файл логів пристрою	Мобільний пристрій	Файл логів без врахування фільтрів	Відповідає
11	- Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перейти до вкладки “Install APK” - Вибрати шлях до файлу пакету - Перевірити наявність встановленого додатку	Мобільний пристрій Файл додатку	Встановлений додаток	Відповідає
12	- Під’єднати пристрій за допомогою кабеля - Натиснути “Connect” - Перейти до вкладки “Install APK” - Вибрати шлях до некоректного фалу пакету - Перевірити наповідомлення про помилку	Мобільний пристрій Файл додатку	Повідомлення про помилку	Відповідає

Протестуємо згідно з написаними тест-кейсами розроблений програмний продукт під назвою «SuperDebug». Головне вікно програми містить дві кнопки: “Coonect to phone”, що призначена для з’єднання з пристроєм, та “Quit”, яка призначена для виходу з програми. Вікно з’єднання додадку зображена на рисунку 4.1.

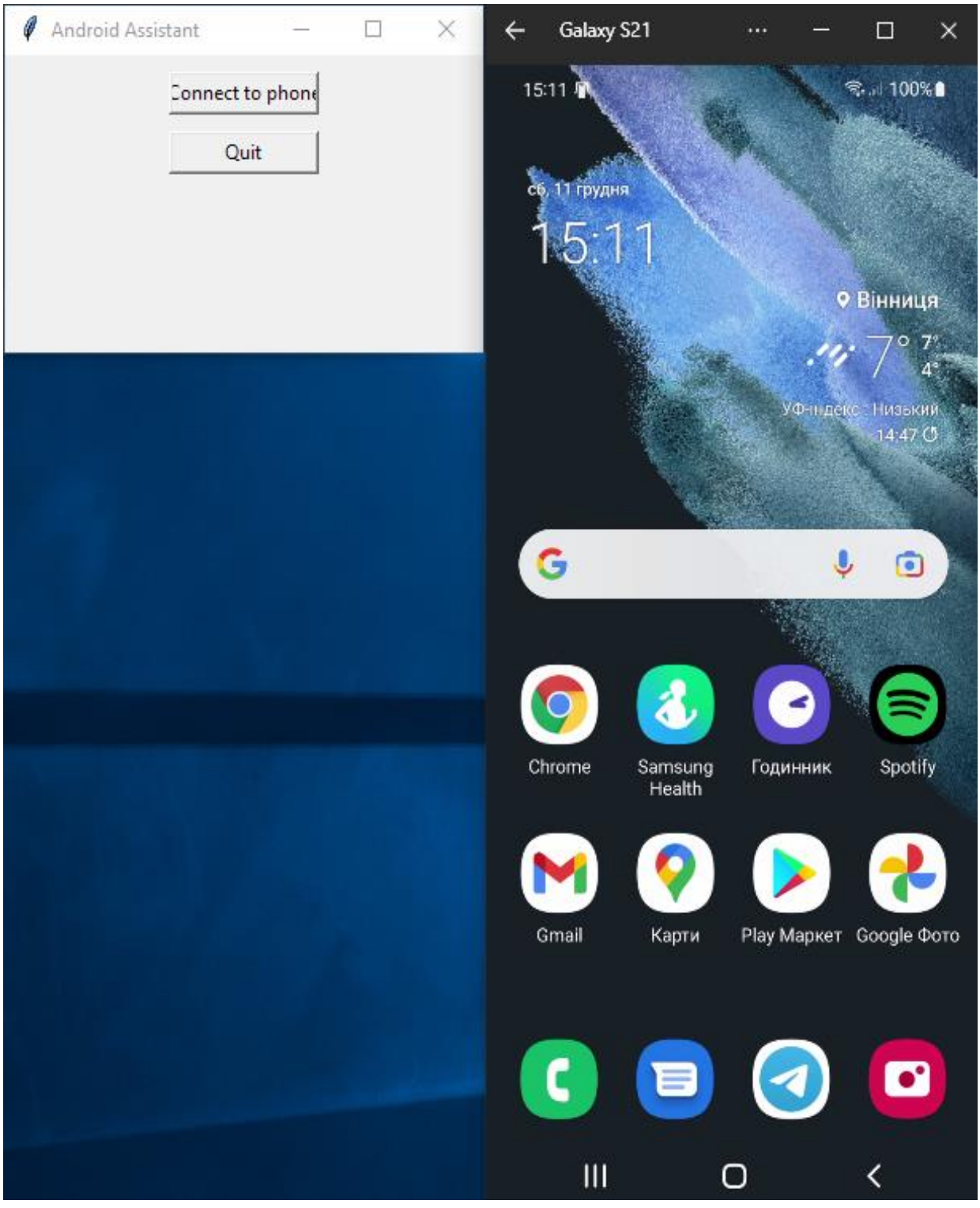

Рисунок 4.2 – Вікно з’єднання додадку

У випадку, якщо мобільний пристрій не підключено, або кабель пошкоджений, процес під’єднання буде неможливим. В такому разі користувач отримує модальне вікно з повідомленням про помилку. Вікно з повідомленням про помилку показано на рисунку 4.3.

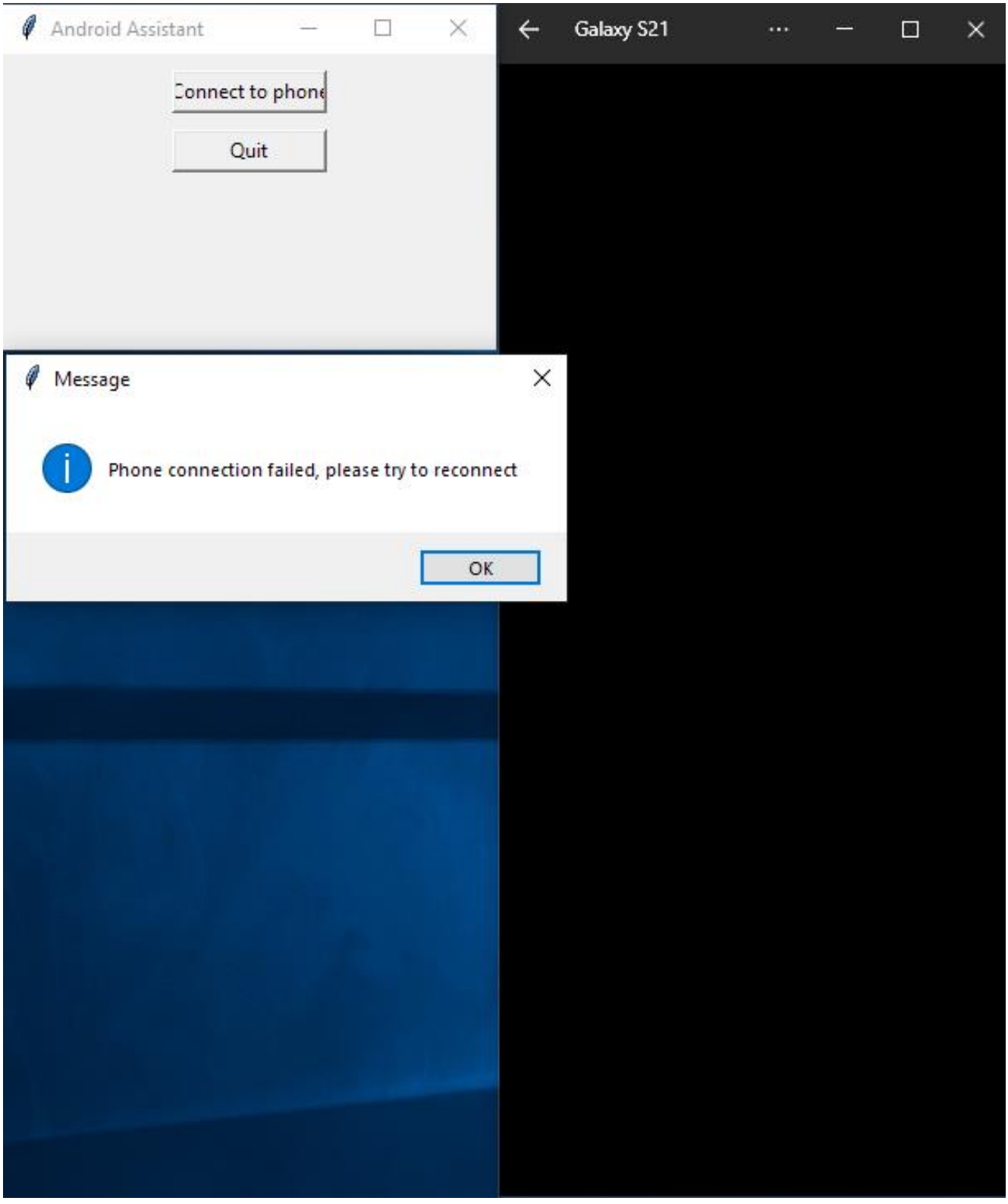

Рисунок 4.3 – Повідомлення про помилку після невдалого з’єднання

Після успішного підключення, користувач побачить головне вікно, що складається з системної інформації пристрою: версія операційної системи, серійний номер мобільного пристрою, роздільна здатність дисплею, кнопки для отримання інформації про встановлені додатки та стан батареї. Вкладка системної інформації показана на рисунку 4.4.

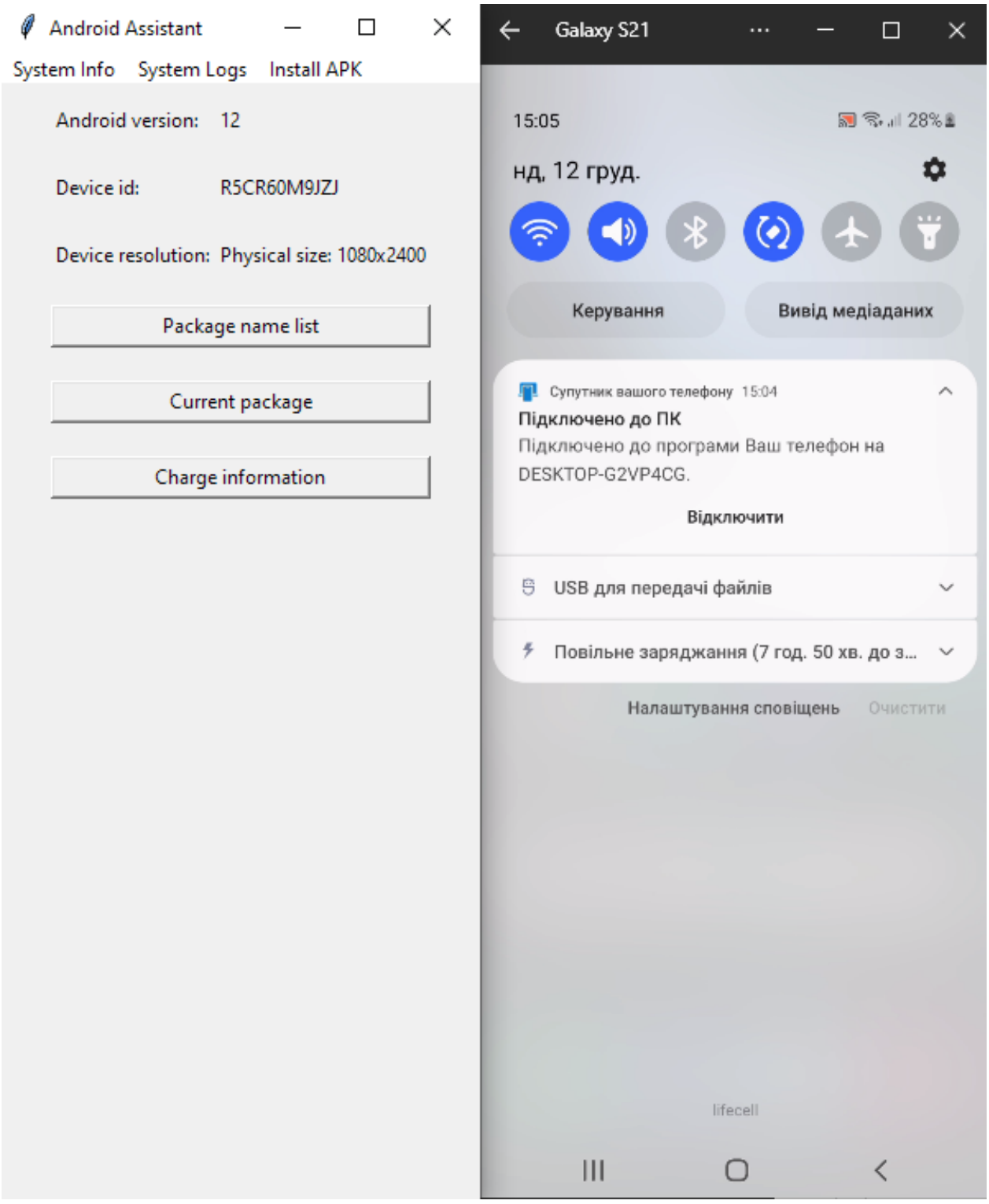

Рисунок 4.4 – Вкладка системної інформації

У додатка, серед інших можливостей, є здлатність переглядати список як усіх встановлених пакетів (дадатків), так і отримати інформацію про який конкретно додаток відкритий на мобільному пристрої. На рисунку 4.5 показаний приклад інформації про відкритий додаток галереї зображень.

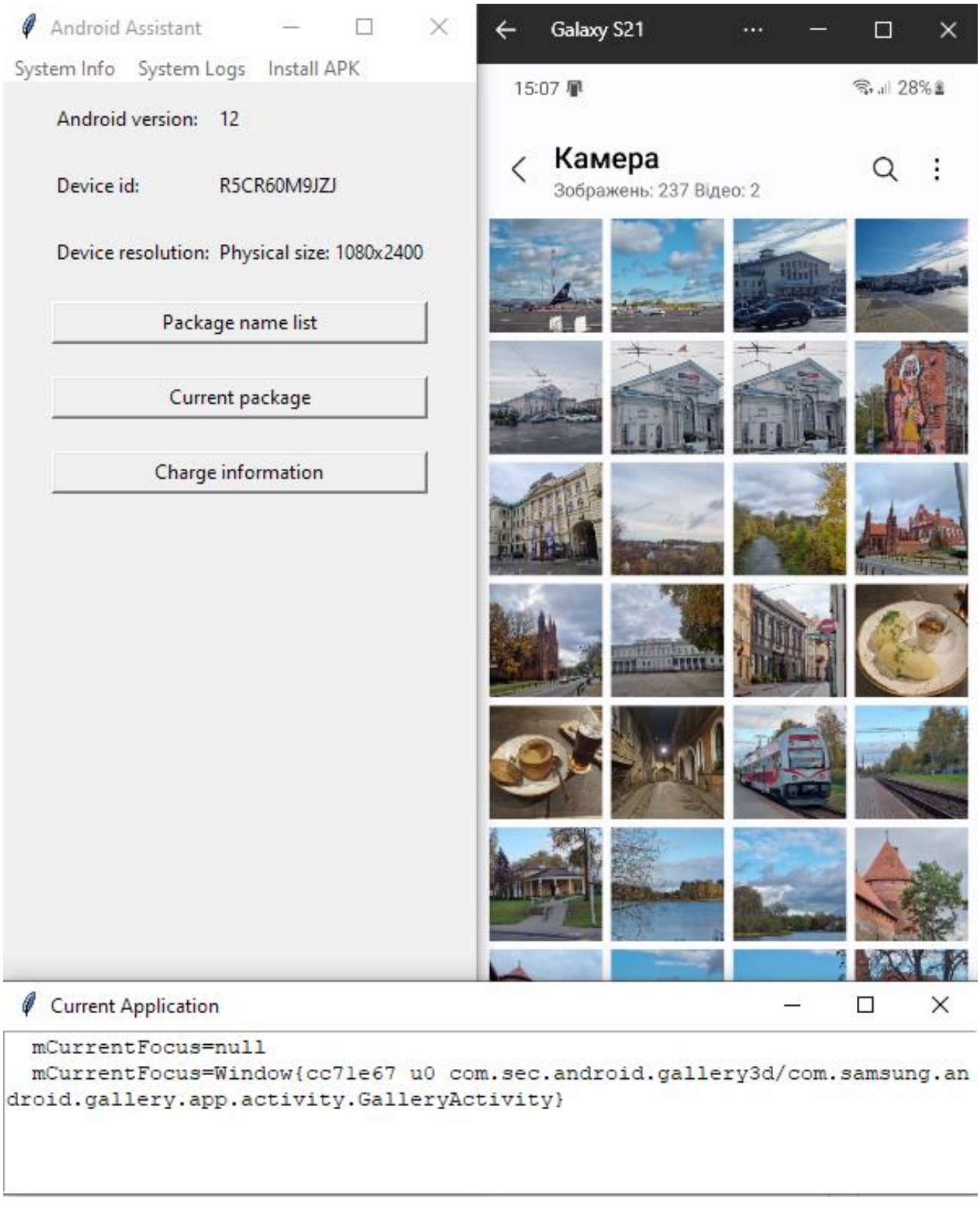

Рисунок 4.5 – Вікно інформації про поточний відкритий додаток

Також в системі було розроблено можливість перегляду усіх встановлених пакетів додатків, як системних та сервісних, так і тих, які були встановлені безпосередньо користувачем. Приклад списку встановлених програм показано на рисунку 4.6.

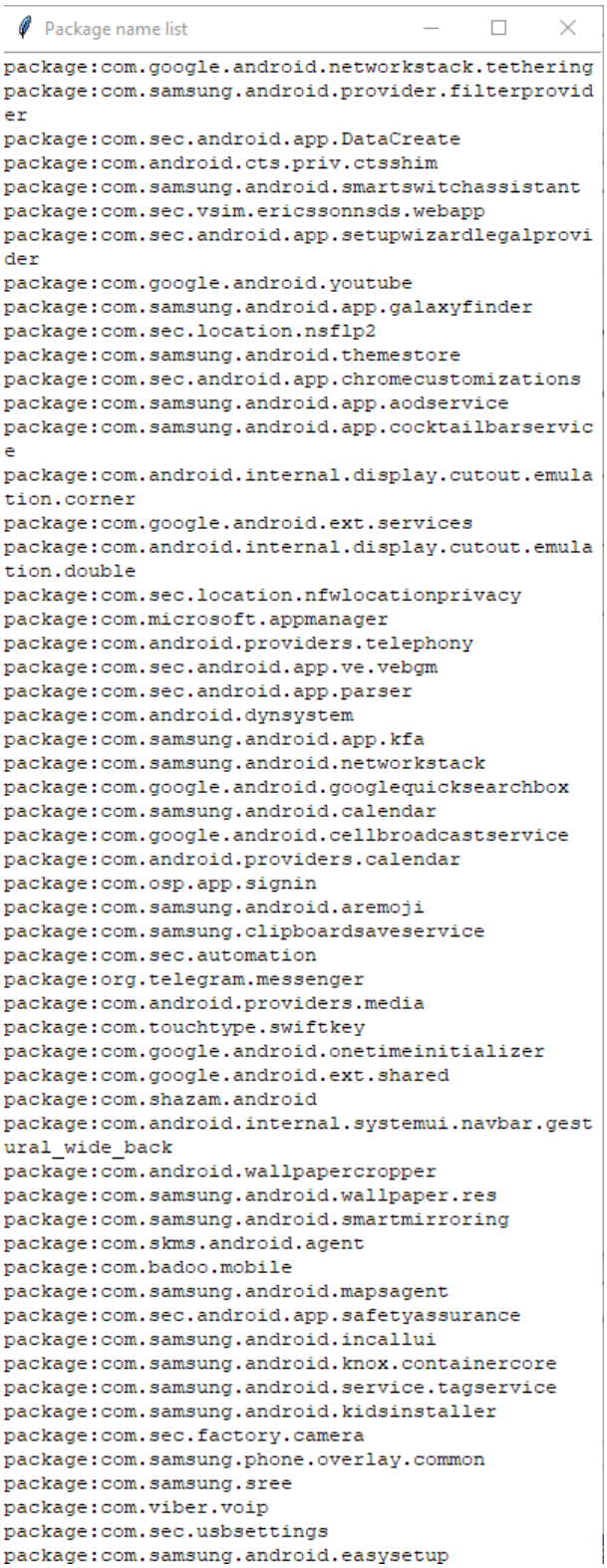

Рисунок 4.6 – Приклад списку встановлених додатків

Одним з інструментів відлагодження є перевірка стану батареї мобільного пристрою. Програма має здатність читати наступні дані: відсоток заряду, стан підключення, тип зарядки (стандартна, швидка чи бездротова зарядка), температура батареї, вольтаж, загальний стан батареї. Приклад виведеного стану батареї показано на рисунку 4.7.

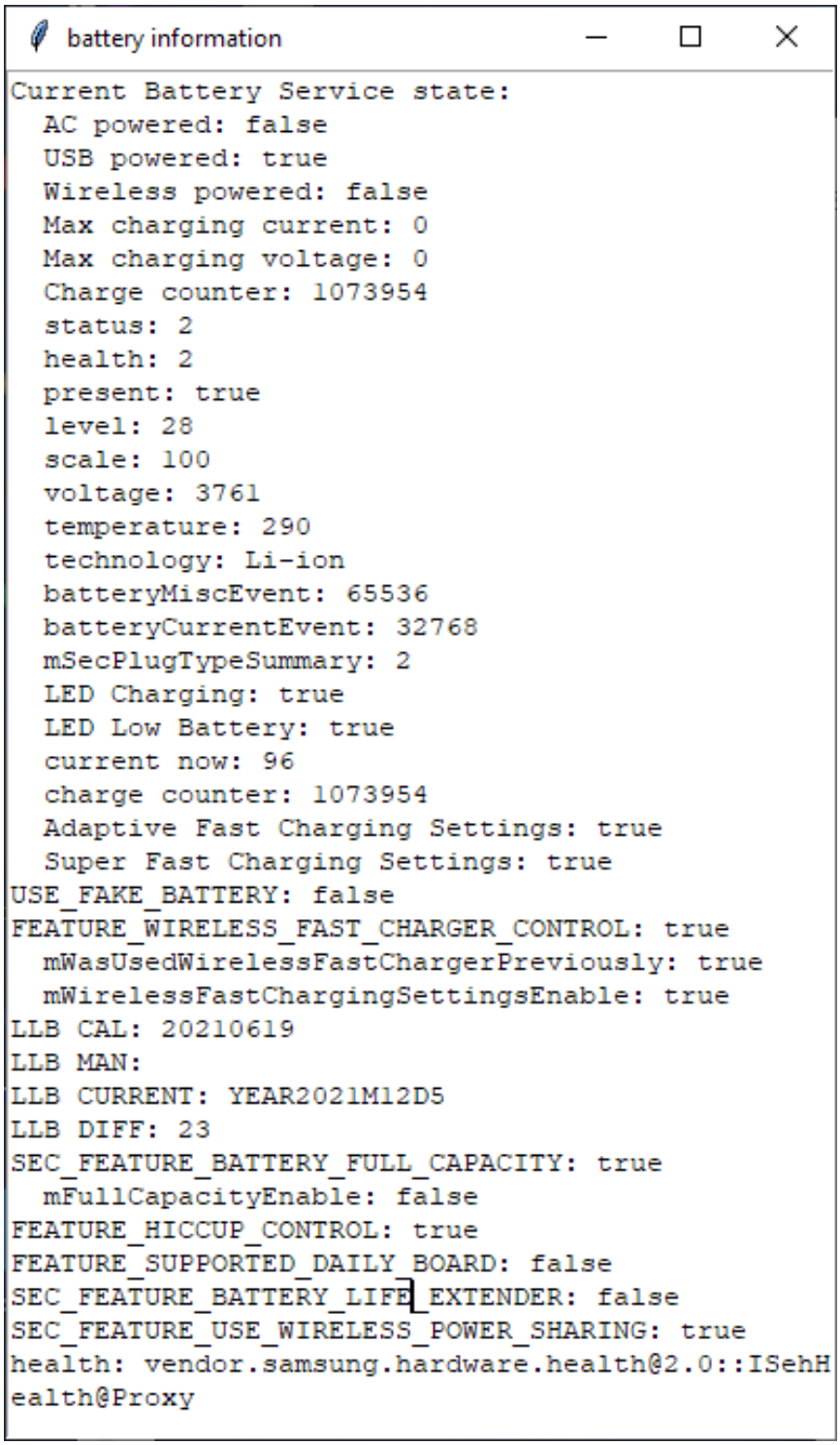

Рисунок 4.7 – Приклад виведеного стану батареї пристрою

Однією з важливих складових тестування мобільних додатків є їх відлагодження за допомогою читання параметрів пристрою та логів його роботи. Для цього був розроблений модуль логування, що міститься у окремій вкладці програми. Вкладка містить поле для фільтру за текстом та рівнем логування. Зчитані логи після цього записуються у новий файл. Вкладка модуля логування показана на рисунку 4.8.

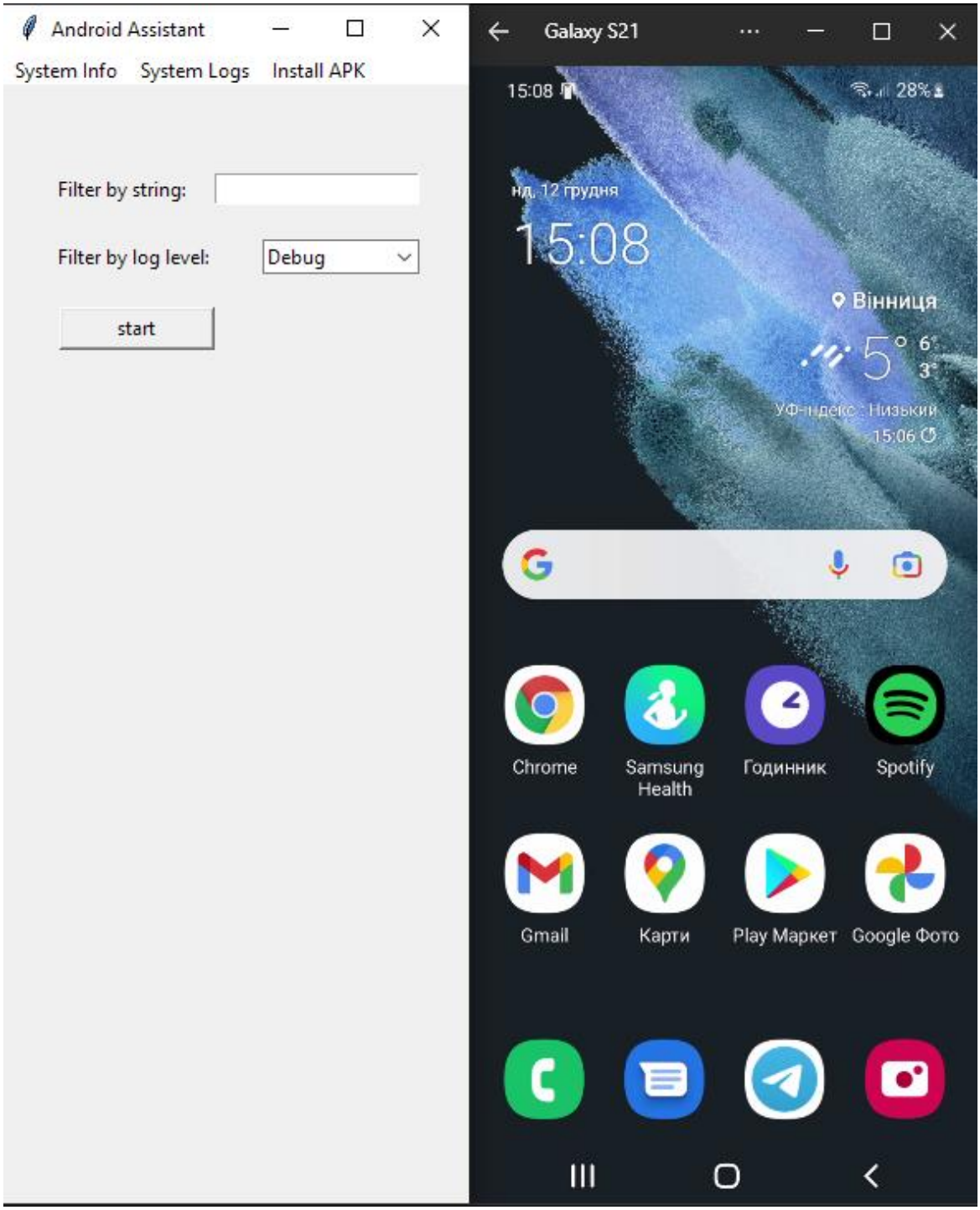

Рисунок 4.8 – Вкладка логування

Ще одним з розроблених модулів є можливість віддаленого встановлення на пристрій пакетів .apk. Файл пакету Android (Android Package File, APK) — це файл, який використовується для розповсюдження програм в операційній системі Google Android. Процес вибору файлу для встановлення показано на рисунку 4.9

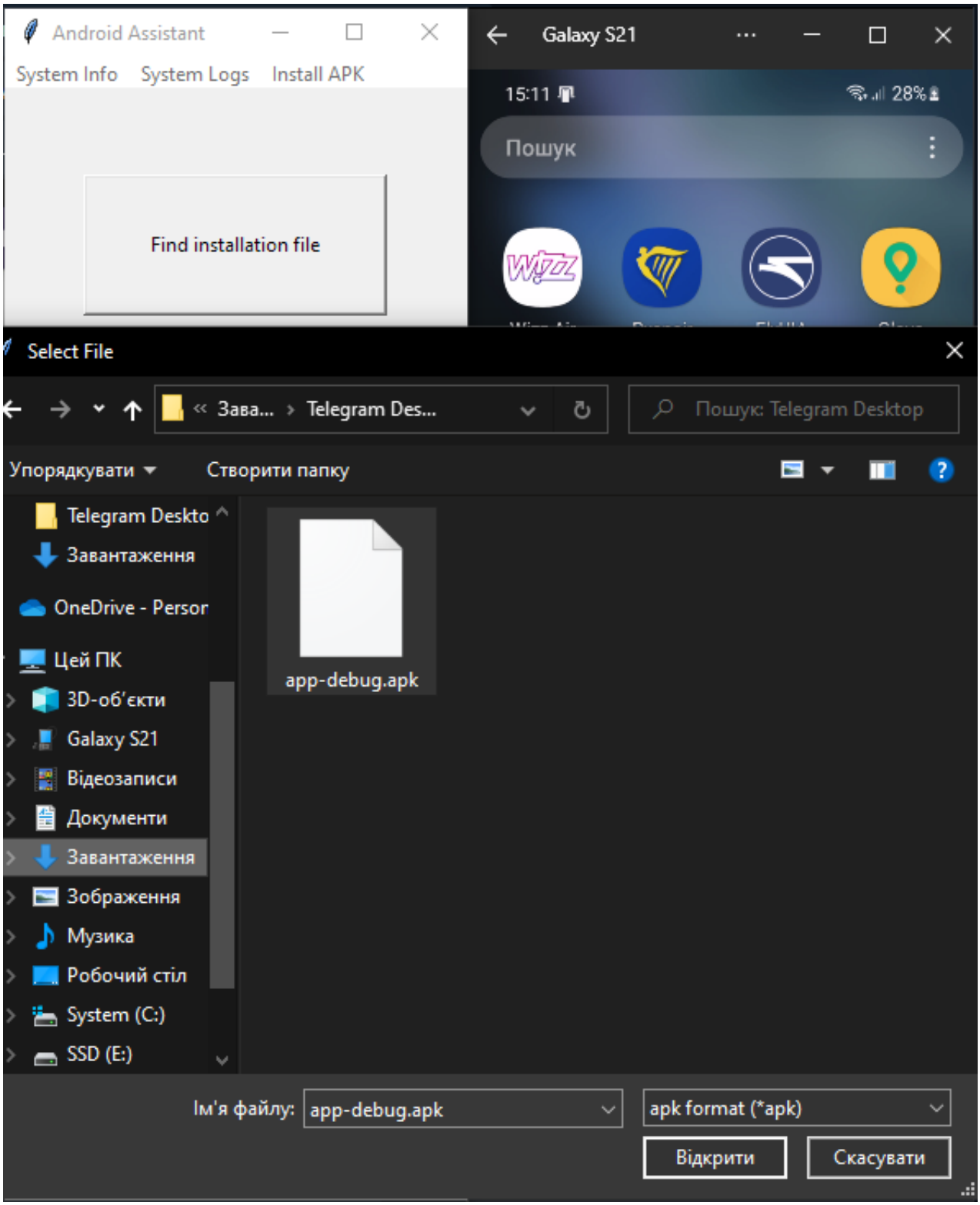

Рисунок 4.9 – Процес вибору файлу інсталяції

Після вибору файлу інсталяцції на комп'ютері, він буде переданий на мобільний пристрій. І після того система Android успішно перевірить додаток на безпечність, почнець процес його встановлення на пристрій. Цей процес супроводжується повідомленнями для користувача. Приклад встановелння додатку показано на рисунку 4.10

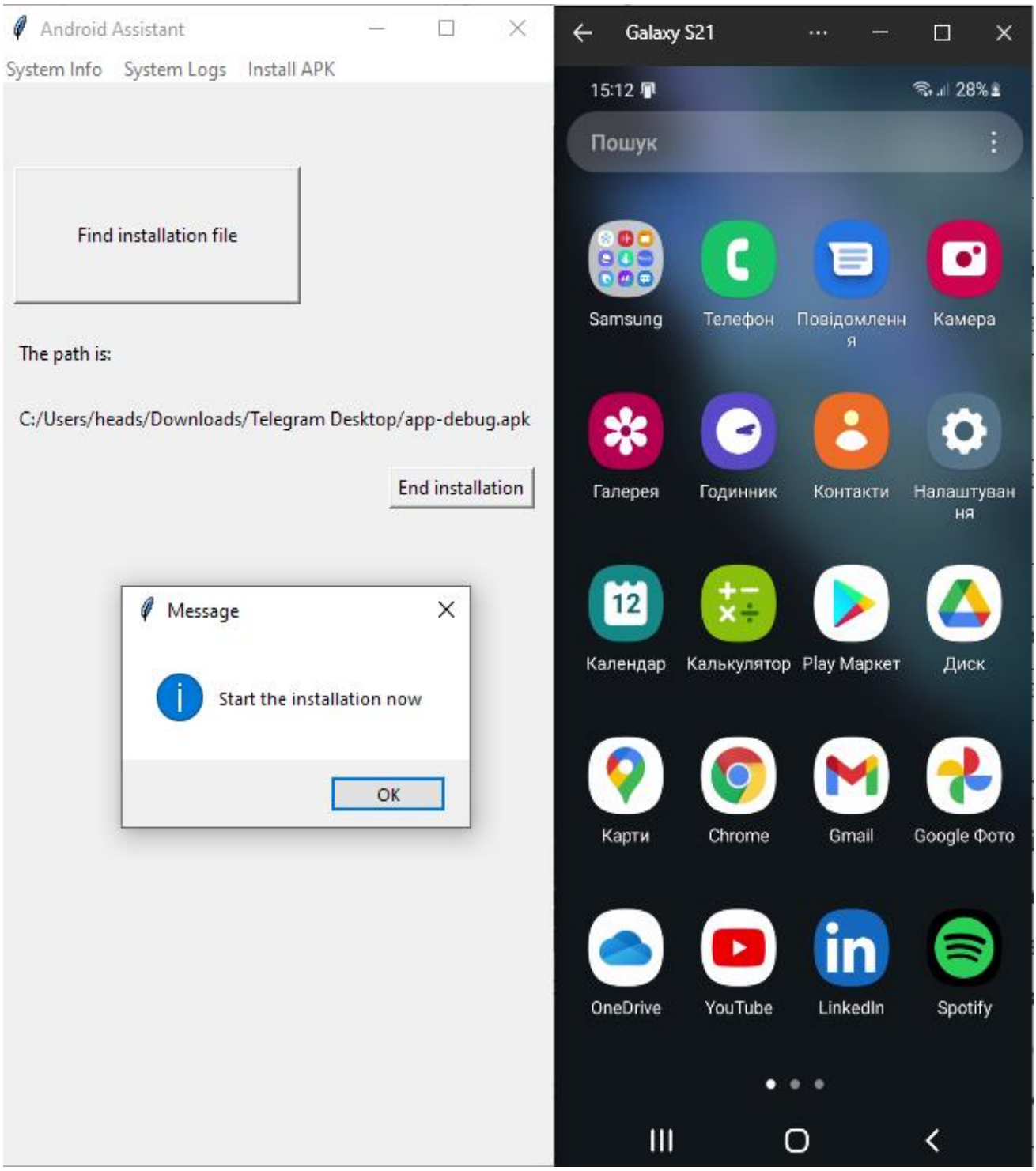

Рисунок 4.10 – Процес інсталяції додатку

Після цього завантажений додаток можна буде відкрити на телефоні. Приклад цього показано на рисунку 4.11

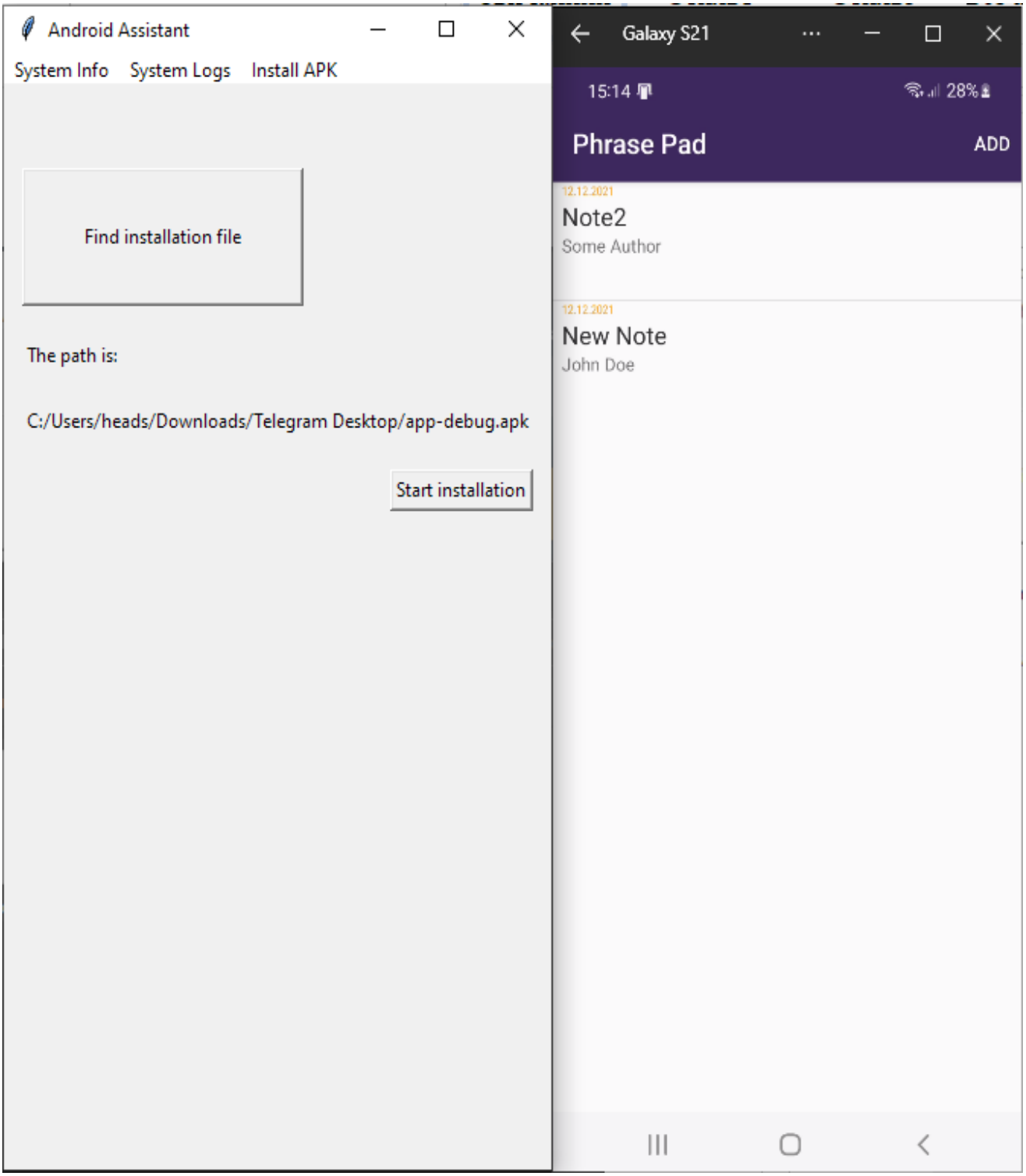

Рисунок 4.11 – Встановлений додаток

Отже, в ході проведеного тестування розробленого програмного продукту дефектів та помилок в роботі виявлено не було. Додаток працює коректно та

відповідає абсолютно всім поставленим функціональним та нефункціональним вимогам.

4.3 Розробка інструкції користувача

Вимоги до складу і параметрів технічних засобів: система повинна працювати на IBM-сумісних персональних комп'ютерах. У таблицях 4.1 та 4.2 наведені мінімальна та рекомендована конфігурації персонального комп'ютера для запуску та належної роботи з програмою.

Таблиця 4.2 – Мінімальна конфігурація:

Тип процесора	32-розрядний (x86) процесор з тактовою частотою 1 ГГц
Об'єм оперативної пам'яті	512 МБ для 32-розрядної системи
Розмір жорсткого диску	100 МБ для 32-розрядної системи
Графічний пристрій	графічний пристрій DirectX9

Таблиця 4.3 – Рекомендована конфігурація:

Тип процесора	64-розрядний (x64) процесор з тактовою частотою 2 ГГц
Об'єм оперативної пам'яті	1 ГБ для 64-разрядної системи
Розмір жорсткого диску	100 МБ для 64-розрядної системи
Графічний пристрій	графічний пристрій DirectX9

Програма повинна працювати під управлінням сімейства сучасних операційних систем Windows (7/8/8.1,10), Linux (Ubuntu 19.04 LTS) та MacOS (Catalina/Big Sur). Так як програма не вимагає інсталяції, то для роботи з нею достатньо запустити файл main.pyw. Після цього користувач може запускати та працювати з програмою.

Розроблена програма виконана у мінімалістичному стилі. Інтерфейс програми містить такі елементи керування: вкладки системної інформації, логів та встановлення пакеті, текстових полів системної інформації, кнопок для виклику додаткової інформації, фільтри для логів, елементи керування іншими діями пов’язаними з відлагодженням додатків, тощо.

Для запуску процесу відлагодження за допомогою логів користувачу потрібно у відповідній вкладці ввести текст у поля фільтрів та натиснути кнопку, а згодом знову натиснути ту саму кнопку для завершення процесу. По його завершенню буде сформовано файл логів присрою.

Для встановлення додатків на пристрій необхідно перейти на відповідну вкладку, обрати файл потрібного формату, та запустити процес інсталяції. І після того система Android успішно перевірить додаток на безпечність, почнець процес його встановлення на пристрій. Цей процес супроводжується повідомленнями для користувача.

4.4 Висновки

У результаті аналізу різних методів для тестування програмного додатку було обрано методику «чорної скриньки».

Відповідно до обраного методу тестування було розроблено інструкцію тестування програмного додатку, таблицю тест-кейсів, відповідно до якої проведено тестування із доведенням відповідності роботи програмного додатку до функціональних та нефункціональних вимог.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу розробки методу та програмного засобу для відлагодження пристроїв під керування операційної системи Android.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету кафедри програмного забезпечення: к.т.н., доцент. Ракитянська Г.Б., д.т.н., професор Ліщинська Л.Б, к.т н., доцент Коваленко О. О. Для проведення технологічного аудиту було використано таблицю 4.1 [22] в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
	Технічна здійсненність концепції:				
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
	Ринкові переваги (недоліки):				
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження табл. 5.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію	Процедура отримання дозвільних документів для виробництва та реалізації	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 4.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Ракитянська Г.Б.	Ліщинська Л.Б.	Коваленко О. О.
	Бали, виставлені експертами:		
1	3	4	3
2	4	3	3
3	4	4	4
4	3	3	3
5	4	3	3
6	4	3	2
7	4	4	3
8	4	3	3
9	3	4	2
10	3	3	3
11	4	4	4
12	3	4	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 42$	$СБ_3=37$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{43+42+37}{3} = 41$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 41 бал, що згідно таблиці 4.2 вважається, що рівень комерційного потенціалу проведених досліджень є високий.

В роботі розробляється універсальний програмний засіб для відлагодження

мобільних пристроїв, що буде поширюватись за допомогою електронних носіїв та мережі Інтернет. Цільовою аудиторією системи є інженери з розробки та тестування ПЗ. Система буде використовуватись на фінальних етапах створення програмного забезпечення різного призначення для мобільних телефонів.

Порівняємо нову розробку з аналогом, який є на ринку. В якості аналога для розробки було обрано Android Remote Debugger – система для відлагодження що дає широкі можливості для перегляду стану пристрою та його контролю через браузер. Телефон підключається не кабелем як зазвичай а через мережу. Незважаючи на багатий функціонал, програма не є універсальною, оскільки має бути вбудована в сам додаток. У розробці дана проблема вирішується за рахунок зв'язку не з додатком, а пристроєм, що дозволяє відлагоджувати декілька додатків одночасно, їх взаємодію та досліджувати їх вплив на параметри пристрою

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

1. Основна заробітна плата кожного із дослідників $З_О$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_О = \frac{М}{Т_р} * t \text{ (грн)} \qquad (5.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для розробки методу та програмного засобу для відлагодження пристроїв під керування операційної системи Android необхідно залучити програміста з посадовим окладом 9000 грн. Кількість робочих днів у місяці складає 22, а кількість робочих днів програміста складає 30. Зведемо сумарні розрахунки до таблиця 5.4.

Таблиця 5.4 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	12000	545,5	5	2727
Програміст	9000	409,1	30	12273
Всього				15000

2. Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{дод}}{100\%} \quad (5.2)$$

$$З_д = 0{,}11 * 15000 = 1650 \text{ (грн)}$$

3. Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою (5.3):

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100} \text{ (грн)} \quad (5.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді:

$$Н_{ЗП} = (15000 + 1650) * \frac{22}{100} = 3663 \text{ (грн)}$$

4. Витрати на матеріали М що були використані під час виконання даного етапу роботи, розраховуються по кожному виду матеріалів за формулою:

$$М = \sum_{1}^{n} Н_i \cdot Ц_i \cdot К_i - \sum_{1}^{n} В_i \cdot Ц_в \text{ грн.,} \quad (5.4)$$

де $Н_i$ – витрати матеріалу *i*-го найменування, кг;

$Ц_i$ – вартість матеріалу *i*-го найменування, грн./кг.;

$К_i$ – коефіцієнт транспортних витрат, $К_i$ = (1,1…1,15);

$В_i$ – маса відходів матеріалу *i*-го найменування, кг;

$Ц_в$ – ціна відходів матеріалу *i*-го найменування, грн/кг;

n – кількість видів матеріалів.

Таблиця 5.5 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
USB-накопичувач	350	1	350
USB-кабель	80	1	80
Упаковка паперу	120	1	120
Ручки	15	2	30
Всього			580
З врахуванням коефіцієнта транспортування			638

5. Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Таблиця 5.6 – Витрати на придбання програмних засобів по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість
Microsoft Visual Studio	1	1200	1200

6. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$A = \frac{Ц \cdot T}{Ткор \cdot 12} \quad [грн], \tag{5.5}$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункта 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 12000 грн.

$$А = \frac{12000 \cdot 1}{2 \cdot 12} = 500$$

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i} \tag{5.6}$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$\eta_і$ – коефіцієнт корисної дії обладнання, $\eta_і < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}3 \cdot 250 \cdot 4{,}1 \cdot 0{,}5}{0{,}8} = 192{,}19$$

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \qquad (5.7)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 15000 \cdot \frac{100}{100\%} = 15000 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

$$В = 15000+1650+3663+638+1200+500+192{,}19+15000=37843{,}2$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.8)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,9.

Звідси:

$$ЗВ = \frac{37843,2}{0,9} = 42047,99 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_{1}^{n} \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N \right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100} \right) \quad (5.9)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2021 році – 18%.

Припустимо, що при впроваджені результатів наукової розробки покращується якість програмного продукту для формування індивідуальних тренувань. Припустимо, що ціна від зросте на 50 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 1000 шт., протягом другого року – на 900 шт., протягом третього року на 850 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до складає 250 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [50 \cdot 1 + (250 + 50) \cdot 1000] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 51256{,}49 \text{ грн.}$$

$$\Delta П_2 = [50 \cdot 1 + (250 + 50) \cdot (1000 + 900)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 97421{,}12 \text{ грн.}$$

$$\Delta П_3 = [50 \cdot 1 + (250 + 50) \cdot (1000 + 900 + 850)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 140981{,}86 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (5.10)$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}} = 2…5$).

$$PV = 2 \cdot 42047{,}99 \ = 84095{,}97$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \tag{5.11}$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{5.12}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$\text{ПП} = \frac{51256{,}49}{(1+0{,}2)^1} + \frac{97421{,}11}{(1+0{,}2)^2} + \frac{140981{,}86}{(1+0{,}2)^3} = 192333{,}49 \text{ грн.}$$

$$\text{Е}_{\text{абс}} = (192333{,}49 - 84095{,}97\ ) = 108237{,}51 \text{грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$\mathrm{E_в} = \sqrt[3]{1 + \frac{108237,51}{84095,97}} - 1 = 0,53 = 53\%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau_{\min} = 0,18 + 0,05 = 0,23.$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{5.15}$$

$$\mathrm{T_{ок}} = \frac{1}{0,53} = 1,9 \text{ роки}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

5.5 Висновки до економічного розділу

Було проведено оцінку комерційного потенціалу розробки методу та програмного засобу для відлагодження пристроїв під керування операційної системи Android, який є на високому рівні. Порівнюючи нову розробку з аналогом, вона є набагато кращою і випереджає його за багатьма показниками.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 37843,2 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 42047,99 грн.

Вкладені інвестиції в даний проект окупляться через 1,9 роки при прогнозованому прибутку 192333,49 грн. за три роки.

ВИСНОВКИ

У магістерській дипломній роботі було розроблено засіб для відлагодження пристроїв під керування операційної системи Android.

Було проаналізовано стан даної проблеми на сьогоднішній день. Розглянуто основні аналоги, визначено їх особливості та недоліки і розроблено порівняння з власним програмним продуктом.

Обрано мови програмування Python та середовища програмування Visual Studio Code для реалізації поставленої задачі.

Були вирішенні наступні задачі:

- розроблено метод відлагодження мобільних пристроїв;
- розроблено модель відлагодження мобільних пристроїв;
- розроблено інтерфейс програмного продукту, який буде легким для розуміння та інтуїтивним користувачу;
- розроблено програмний продукт, призначений для вирішення проблеми;
- проведено тестування програмного продукту для перевірки всіх можливих варіантів використання

Запропоновано новий метод налагодження пристроїв при розробці мобільних додатків. Розроблено модель програмної системи відлагодження.

Також було розроблено структуру програми. При реалізації програми було обрано бібліотеку Tkinter, що відповідає за графічний інтерфейс користувача.

Тестування програми довело повну працездатність даного програмного продукту та відповідність поставленому технічному завданню. Розроблено інструкцію користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ЕЛЕКТРОННІ ІНФОРМАЦІЙНІ РЕСУРСИ: СТВОРЕННЯ, ВИКОРИСТАННЯ, ДОСТУП – Суми/Вінниця: Суми, НІКО, 2021. – 225 с. – (УДК 004). – (ISBN 978-617-7422-16-6).
2. McWherter J. Professional Mobile Application Development / J. McWherter, S. Growell. – Indianapolis: Wrox, 2012. – 432 с.
3. Franceschi H. Android App Development / Hervé J. Franceschi. – Burlington: Jones & Bartlett Learning. – 682 с.
4. Thompson S. Debugging Code / Sam Thompson. – New York: Gareth Stevens Publishing, 2021. – 240 с.
5. Stroustrup B. The C++ Programming Language / Bjarne Stroustrup. – Boston: Addison-Wesley, 2013. – 480 с.
6. Richter J. CLR Via C# / Jeffrey Richter. – Seatle: Microsoft, 2012. – 863 с.
7. Ramalho L. Fluent Python / Luciano Ramalho. – Sebastopol: O'Reilly Media, Inc., 2013. – 792 с.
8. Urma R. Java 8 in Action. Lambdas, streams, and functional-style programming / R. Urma, M. Fusco, A. Mycroft. – Shelter Island: Manning Publications, 2014. – 424 с.
9. Microsoft Visual Studio [Електронний ресурс] – Режим доступу до матеріалу: https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio.
10. Швець О. Занурення в паттерни проектування. / Олександ Швець. – Кам’янець-Подільський: Refactoring.Guru, 2021. – 393 с.
11. Software Architecture: The Hard Parts: Modern Trade-Off Analyses for Distributed Architectures / N.Ford, M. Richards, P. Sadalage, Z. Dehghani., 2021. – 464 с.
12. Графічний інтерфейс користувача [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Графічний_інтерфейс_користувача.
13. Single document interface [Електронний ресурс]: Режим доступу до матеріалу: https://ru.wikipedia.org/wiki/Single_document_interface.
14. Multiple document interface [Електронний ресурс]: Режим доступу до матеріалу: https://ru.wikipedia.org/wiki/Multiple_document_interface.

15. Metro (design language) [Електронний ресурс] – Режим доступу до матеріалу: https://en.wikipedia.org/wiki/Metro_(design_language).
16. Tkinter [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Tkinter.
17. Алгоритм [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Алгоритм.
18. Діаграма класу [Електронний ресурс] – Режим доступу: https://uk.uzvisit.com/323-class-diagram.
19. Тестування програмного забезпечення [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення.
20. Petschenik N. System Testing With an Attitude / Nathan Petschenik. – New York: Dorset House, 2004. – 306 с
21. Калбертсон Р. Быстрое тестирование. / Р. Калбертсон, К. Браун, Г. Кобб. – М.: «Вильямс»,2002. – 374 с.
22. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2021 р.

Технічне завдання

на магістерську кваліфікаційну роботу

«Розробка методу і програмного засобу відлагодження пристроїв під керування операційної системи Android»

за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____к.т.н., доц. Ракитянська Г. Б.

"____"_____2021 р.

Виконав:

___студент гр. 1ПІ-20м Савицький Д.С.

"____"_____2021 р.

Вінниця – 2021 рік

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмного засобу для відлагодження пристроїв під керування операційної системи Android».

Галузь застосування – розробка мобільних додатків.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ №277 від 24.09.2020 ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою розробки є надання можливості ефективного способу відлагодження мобільних пристроїв під керування операційної системи Android, шляхом розробки та програмної реалізації алгоритмів відлагодження.

Практична цінність отриманих результатів полягає у тому, що отримані в магістерській роботі методи використано для розробки та реалізації алгоритмів відлагодження мобільних пристроїв.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. McWherter J. Professional Mobile Application Development / J. McWherter, S. Growell. – Indianapolis: Wrox, 2012. – 432 с.
2. Franceschi H. Android App Development / Hervé J. Franceschi. – Burlington: Jones & Bartlett Learning. – 682 с.
3. Thompson S. Debugging Code / Sam Thompson. – New York: Gareth Stevens Publishing, 2021. – 240 с.

5. Технічні вимоги

Розробити програмний продукт, призначений для відлагодження пристроїв під керування операційної системи Android

6. Конструктивні вимоги

Програмний засіб повинен реалізувати налагодження пристрою за допомогою зняття системної інформації пристрою, читання логів та відслідковування стану програм. Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз сучасного стану питання та обґрунтування задачі	15.09. 2021 – 03.10.2021
2	Розробка методу відлагодження і моделі програмного додатку	03.10.2021 – 18.10.2021
3	Розробка програмного додатку	18.10.2021 – 05.11.2021
4	Тестування програмного додатку	05.11.2021 – 19.11.2021
5.	Економічна частина	19.11.2021 - 30.11.2021

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б. Протокол перевірки роботи

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методу і програмного засобу відлагодження пристроїв під керуванням операційної системи Android.**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ – 20м

Науковий керівник: к.т.н. доц. Ракитянська Г. Б.

Unicheck	
Оригінальність	**90,6 %**
Схожість	9,4 %

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлена з повним звітом подібності, який був згенерований Системою щодо роботи «Розробка методу і програмного засобу відлагодження пристроїв під керуванням операційної системи Android».

Автор _____ Савицький Дмитро Сергійович

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г. О.
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

Додаток В. Лістинг програми

```
from tkinter import *
from HomePage1 import *

root = Tk()
root.title('Android Assistant')

HomePage1(root)
root.mainloop()

from tkinter import *
from view import *

class MainPage(object):
    def __init__(self, master=None):
        self.root = master
        self.root.geometry('%dx%d'% (600, 450))
        self.createPage()

    def createPage(self):
        self.inputPage = InputFrame(self.root)
        self.installPage=InstallFrame(self.root)
        self.queryPage = QueryFrame(self.root)
        self.countPage = CountFrame(self.root)
        self.aboutPage = AboutFrame(self.root)
        self.seniorPage = SeniorFrame(self.root)
        self.seniorPage.pack()
        menubar = Menu(self.root)
        menubar.add_command(label='System Info', command=self.senior)
        menubar.add_command(label='System Logs',
command=self.countData)
        menubar.add_command(label='Install APK',
command=self.installData)
        self.root['menu'] = menubar
```

```
def inputData(self):
    self.inputPage.pack()
    self.installPage.pack_forget()
    self.queryPage.pack_forget()
    self.countPage.pack_forget()
    self.aboutPage.pack_forget()

    self.seniorPage.pack_forget()
def installData(self):
    self.inputPage.pack_forget()
    self.installPage.pack()
    self.queryPage.pack_forget()
    self.countPage.pack_forget()
    self.aboutPage.pack_forget()
    self.seniorPage.pack_forget()

def queryData(self):
    self.inputPage.pack_forget()
    self.queryPage.pack()
    self.countPage.pack_forget()
    self.aboutPage.pack_forget()
    self.installPage.pack_forget()
    self.seniorPage.pack_forget()

def countData(self):
    self.inputPage.pack_forget()
    self.queryPage.pack_forget()
    self.countPage.pack()
    self.aboutPage.pack_forget()
    self.installPage.pack_forget()
    self.seniorPage.pack_forget()

def aboutDisp(self):
    self.inputPage.pack_forget()
```

```
        self.queryPage.pack_forget()
        self.countPage.pack_forget()
        self.aboutPage.pack()
        self.installPage.pack_forget()
        self.seniorPage.pack_forget()

    def helpMe(self):
        self.inputPage.pack_forget()
        self.queryPage.pack_forget()
        self.countPage.pack_forget()
        self.aboutPage.pack_forget()
        self.installPage.pack_forget()
        self.seniorPage.pack_forget()

    def senior(self):
        self.inputPage.pack_forget()
        self.queryPage.pack_forget()
        self.countPage.pack_forget()
        self.aboutPage.pack_forget()
        self.installPage.pack_forget()
        self.seniorPage.pack()

from tkinter import *
from tkinter import ttk
from tkinter.messagebox import *
import os
import subprocess
import datetime
import tkinter as tk
import re
import tkinter.filedialog

class InputFrame(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
```

```
        self.root = master
        self.itemName = StringVar()
        self.importPrice = StringVar()
        self.sellPrice = StringVar()
        self.deductPrice = StringVar()
        self.createPage()

    def createPage(self):
        Label(self).grid(row=0, stick=W, pady=10)
        Button(self,text='screenshot',width=25,height=5,command=self.
screenshot).grid(row=1, stick=W, pady=10)
        self.switchscrBtn = Button(self, text='start recording',
width=25, height=5, command=self.switchscreenrecord)
        self.switchscrBtn.grid(row=2, stick=W, pady=10)

    def switchscreenrecord(self):
        if self.switchscrBtn['text'] =='Start recording':
            self.switchscrBtn['text'] ='End video'
            self.startrecord()
        else:
            self.switchscrBtn['text'] ='Start recording'
            self.endrecord()

    def startrecord(self):
        nowtime = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
        self.name1 = nowtime +'test.mp4'
        out ='adb shell screenrecord/sdcard/test.mp4'
        self.pro = subprocess.Popen(out, stderr=subprocess.PIPE)

    def endrecord(self):
        self.pro.kill()
        out2 = subprocess.getstatusoutput('adb pull/sdcard/test.mp4
.\{}'.format(self.name1))

    def screenshot(self):
```

```
        nowtime=datetime.datetime.now().strftime('%Y%m%d%H%M%S')
        name=nowtime+'.png'
        out = subprocess.getstatusoutput('adb shell screencap -
p/sdcard/screen.png')
        out1 = subprocess.getstatusoutput('adb pull/sdcard/screen.png
.\{}'.format(name))
        if out[0]==0 and out1[0]==0:
            pass
        else:
            tk.messagebox.showinfo("Message", "Unknown reason,
screenshot failed")

class InstallFrame(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.root = master
        self.itemName = StringVar()
        self.importPrice = StringVar()
        self.sellPrice = StringVar()
        self.deductPrice = StringVar()
        self.number = StringVar()
        self.createPage()

    def createPage(self):
        Label(self).grid(row=0, stick=W, pady=10)
        Button(self, text='Find installation file', width=25,
height=5, command=self.search).grid(row=1, stick=W, pady=10)
    def search(self):
        default_dir = r"C:\Users\lenovo\Desktop"
        self.filename =
tkinter.filedialog.askopenfilename(title=u"Select
File",initialdir=(os.path.expanduser(default_dir)),filetypes=[("apk
format", "apk")])
        Label(self,text='The path is:').grid(row=2,stick=W,pady=10)
        Label(self, text=self.filename).grid(row=3, pady=10, stick=W)
```

```
        self.B = Button(self, text='Start installation', width=12,
command=self.switch)
        self.B.grid(row=6, stick=E, pady=10)

    def switch(self):
        print(self.B)

        if self.B['text'] =='Start installation':
            self.B['text'] ='End installation'
            self.startinstall()
        else:
            self.B['text'] ='Start installation'
            self.endinstall()

    def startinstall(self):
        tk.messagebox.showinfo("Message", "Start the installation
now")
        out1=self.filename
        out ='adb install -r'
        out2=out+out1
        self.pro = subprocess.Popen(out2, stderr=subprocess.PIPE)
    def endinstall(self):
        self.pro.kill()

class QueryFrame(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.root = master
        self.packageName=StringVar()
        self.pressureNum=StringVar()
        self.seedNum=StringVar()
        self.logLevel=StringVar()
        self.throttle=StringVar()
        self.ignoreCrash=StringVar()
        self.ignoreANR=StringVar()
```

```
        self.ignoreTimeout=StringVar()
        self.createPage()

    def createPage(self):
        Label(self).grid(row=0, stick=W, pady=10)
        Label(self, text='Package name:').grid(row=1, stick=W,
pady=10)
        Entry(self, textvariable=self.packageName).grid(row=1,
column=1, stick=E)
        Label(self, text='Pressure:').grid(row=2, stick=W, pady=10)
        Entry(self, textvariable=self.pressureNum).grid(row=2,
column=1, stick=E)
        self.press = Entry(self, textvariable=self.pressureNum)
        self.press.insert(10, 1000)
        Label(self, text='Mark:').grid(row=3, stick=W, pady=10)
        Entry(self, textvariable=self.seedNum).grid(row=3, column=1,
stick=E)
        self.seed =Entry(self, textvariable=self.seedNum)
        self.seed.insert(10, 1)
        Label(self, text='Log level:').grid(row=4, stick=W, pady=10)
        numberChosen = ttk.Combobox(self, width=12,
textvariable=self.logLevel, state='readonly')
        numberChosen['values'] = (1, 2, 3)
        numberChosen.grid(column=1, row=4,stick=E)
        numberChosen.current(2)
        Label(self, text='throttle').grid(row=5, stick=W, pady=10)
        numberChosen = ttk.Combobox(self, width=12,
textvariable=self.throttle, state='readonly')
        numberChosen['values'] = (100,150,200,250,300,350,400)
        numberChosen.grid(column=1, row=5, stick=E)
        numberChosen.current(0)
        Checkbutton(self,text='ignore
crash',onvalue=1,offvalue=0,variable =self.ignoreCrash).grid(row=6,
column=1, stick=E)
```

```
        Checkbutton(self, text='ignore
timeout',onvalue=1,offvalue=0,variable
=self.ignoreTimeout).grid(row=7, column=1, stick=E)
        self.startBtn=Button(self, text='Start pressure
test',command=self.switch)
        self.startBtn.grid(row=8, column=5, stick=E, pady=5)

        Button(self, text='Report Analysis').grid(row=9, column=5,
stick=E, pady=5)

    def switch(self):

        if self.startBtn['text'] =='Start pressure test':
            self.startBtn['text'] ='Forcibly terminate'
            self.monkey()
        else:
            self.startBtn['text'] ='Start pressure test'
            self.killPro()

    def monkey(self):
        packageName=self.packageName.get()
        print('package name{)'.format(packageName))
        pressureNum=int(self.pressureNum.get())
        print('Number of pressures: ()'.format(pressureNum))
        seedNum=int(self.seedNum.get())
        print('Mark: {}'.format(seedNum))
        logLevel=int(self.logLevel.get())
        print('Log Level: ()'.format(logLevel))
        ignoreCrash=self.ignoreCrash.get()
        print('Ignore the crash: ()'.format(ignoreCrash))
        ignoreTimeout=self.ignoreTimeout.get()
        throttle=int(self.throttle.get())
        print('Time interval: ()'.format(throttle))
        print('Ignore timeout: ()'.format(ignoreTimeout))
        nowtime = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
```

```
        filename ='Monkey'+nowtime + ".txt"
        logcat_file = open(filename,'w')
        logcmd = r'adb shell monkey -p {} -s {} -v -v -v --throttle
{} --ignore-crashes --ignore-timeouts --ignore-security-exceptions --
ignore-native-crashes --monitor-native-crashes
{}'.format(packageName,seedNum,throttle,pressureNum)
        self.pro = subprocess.Popen(logcmd, stdout=logcat_file,
stderr=subprocess.PIPE)

    def some_adb_cmd(self):
        p = subprocess.Popen('adb shell cd sdcard && cd Android && cd
data && ps |grep monkey',stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
        return_code = p.poll()
        while return_code is None:
            line = p.stdout.readline()
            return_code = p.poll()
            line = line.strip()
            line1=str(line,"utf-8")
            pattern = re.compile(r'[^\d]+(\d+)[^\d]+')
            res = re.findall(pattern, line1)
            res1=res[0]
            a='adb shell cd sdcard && cd Android && cd data'
            a1='kill'+res1
            a2=a+" && "+a1
            p1 = subprocess.Popen(a2,stdout=subprocess.PIPE,
stderr=subprocess.PIPE)
            break

    def killPro(self):

        self.some_adb_cmd()

class CountFrame(Frame):
    def __init__(self, master=None):
```

```
        Frame.__init__(self, master)
        self.root = master
        self.filterTag=StringVar()
        self.filterStr=StringVar()
        self.filterRegular=StringVar()
        self.logLevel = StringVar()
        self.filterFormat=StringVar()
        self.createPage()

    def createPage(self):
        Label(self, text='Filter by string:').grid(row=2, stick=W,
pady=10)
        Entry(self, textvariable=self.filterStr).grid(row=2,
column=1, stick=E)
        Label(self, text='Filter by log level:').grid(row=5, stick=W,
pady=10)
        Label(self).grid(row=0, stick=W, pady=10)
        filterFormat = ttk.Combobox(self, width=12,
textvariable=self.filterFormat, state='readonly')
        filterFormat['values'] =
('Verbose','Debug','Warn','Error','Fatal')
        filterFormat.grid(column=1, row=5, stick=E)
        filterFormat.current(1)
        self.B=Button(self, text='start',
width=12,command=self.switch)
        self.B.grid(row=6, stick=E, pady=10)

    def switch(self):
        print(self.B)
        if self.B['text'] =='Start':
            self.B['text'] ='End'
            self.logCat()
        else:
            self.B['text'] ='Start'
            self.killPro()
```

```
    def logCat(self):
        filterStr=self.filterStr.get()
        print('Filter string: ()'.format(filterStr))
        filterFormat=self.filterFormat.get()
        print('Filter format item: ()'.format(filterFormat))
        nowtime = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
        filename = nowtime + ".txt"
        logcat_file = open(filename,'w')
        if filterFormat!='' and filterStr!='':
            logcmd = r'adb shell cd sdcard && cd Android && cd data
&& logcat *:{} | grep {}'.format(filterFormat,filterStr)
        elif filterFormat!='' and filterStr=='':
            logcmd = r'adb shell cd sdcard && cd Android && cd data
&& logcat *:{}'.format(filterFormat)
        elif filterFormat=='' and filterStr !='':
            logcmd = r'adb shell cd sdcard && cd Android && cd data
&& logcat | grep {}'.format(filterStr)
        else:
            logcmd ='adb logcat -v time'
        self.pro = subprocess.Popen(logcmd, stdout=logcat_file,
stderr=subprocess.PIPE)

    def killPro(self):
        self.pro.kill()

class AboutFrame(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.root = master
        self.createPage()

    def createPage(self):
        Button(self, text='Memory Monitoring', width=12,
command=self.getMemory).grid(row=1, stick=E, pady=10, column=1)
```

```
        Button(self, text='Mobile CPU monitoring', width=12,
command=self.allCpu).grid(row=2, stick=E, pady=10, column=1)
        Button(self, text='current application CPU', width=12,
command=self.myCpu).grid(row=2, stick=E, pady=10, column=1)
        Button(self, text='start time', width=12,
command=self.startTime).grid(row=4, stick=E, pady=10, column=1)

    def getMemory(self):
        import re
        out = subprocess.getstatusoutput('adb shell dumpsys window |
findstr mCurrentFocus')
        str1 =out[1]
        pattern = re.compile(r'u0\s*(.+?)\/')
        res = re.findall(pattern, str1)
        self.package=res[0]
        nowtime = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
        filename = "startime" + nowtime + ".xls"
        startime_file = open(filename,'w')
        order = "adb shell dumpsys meminfo {}".format(self.package)
        self.pro = subprocess.Popen(order, stdout=startime_file,
stderr=subprocess.PIPE)

    def allCpu(self):
        top = tk.Toplevel()
        top.title('Memory Monitoring')
        top.geometry('%dx%d'% (700, 1400))
        t = Text(top, width=700, height=100)
        t.pack(fill=tkinter.X, side=tkinter.BOTTOM)
        top.mainloop()
        logcmd='adb shell&&top -m 10 -s cpu'
        out=subprocess.Popen(logcmd,stderr=subprocess.PIPE)
        t.insert(tkinter.END,"{}".format(out[1]))
        t.see(tkinter.END)
        t.update()
```

```
    def myCpu(self):
        import re
        out = subprocess.getstatusoutput('adb shell dumpsys window |
findstr mCurrentFocus')
        str1 = out[1]
        pattern = re.compile(r'u0\s*(.+?)\/')
        res = re.findall(pattern, str1)
        self.packageName = res[0]
        out = subprocess.getstatusoutput('adb shell dumpsys cpuinfo |
find "{}"'.format(self.packageName))
        top = tk.Toplevel()
        top.title('Current Application')
        top.geometry('%dx%d'% (300, 100))
        t = Text(top, width=300, height=100)
        t.insert('1.0', "{}".format(out[1]))
        t.pack()
        top.mainloop()

    def startTime(self):
        pass

class SeetingFrame(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.root = master
        self.createPage()

    def createPage(self):
        Label(self, text='Work in Progress').pack()

class SeniorFrame(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.root = master
        self.createPage()
```

```
    def createPage(self):
        Label(self, text='Android version:').grid(row=0, sticky=W,
pady=10)
        Label(self, text=self.systemversion()).grid(row=0, column =
1, sticky=W, pady=10)
        Button(self, text='Current package', width=12,
command=self.currentPackage).grid(row=4 ,sticky=W+E, pady=10, column
= 0, columnspan=2)
        Button(self, text='Package name list', width=12,
command=self.listPackage).grid(row=3, sticky=W+E, pady=10, column =
0, columnspan=2)
        Label(self, text='Device resolution:').grid(row=2, stick=W,
pady=10)
        Label(self, text=self.resolution()).grid(row=2, column = 1,
sticky=W, pady=10)
        Label(self, text='Device id:').grid(row=1, stick=W, pady=10)
        Label(self, text=self.deviceid()).grid(row=1, column = 1,
sticky=W, pady=10)
        Button(self, text='Charge information', width=12,
command=self.charge).grid(row=5, stick=E, pady=10, column=0,
columnspan=2, sticky=W+E)

    def currentPackage(self):
        out = subprocess.getstatusoutput('adb shell dumpsys window |
findstr mCurrentFocus')
        top = tk.Toplevel()
        top.title('Current Application')
        top.geometry('%dx%d'% (700,100))
        t = Text(top, width=700, height=100)
        t.insert('1.0', "{}".format(out[1]))
        t.pack()
        top.mainloop()

    def listPackage(self):
```

```
        nowtime = datetime.datetime.now().strftime('%Y%m%d%H%M%S')
        filename ='package'+nowtime + ".txt"
        logcat_file = open(filename,'w')
        logcmd ='adb shell pm list packages'
        self.pro = subprocess.Popen(logcmd, stdout=logcat_file,
stderr=subprocess.PIPE)
        out = subprocess.getstatusoutput('adb shell pm list
packages')
        top = tk.Toplevel()
        top.title('Package name list')
        top.geometry('%dx%d'% (400, 1200))
        t=Text(top,width=400,height=900)
        t.insert('1.0',"{}".format(out[1]))
        t.pack()
        top.mainloop()

    def resolution(self):
        out = subprocess.getstatusoutput('adb shell wm size')
        return format(out[1])

    def systemversion(self):
        out = subprocess.getstatusoutput('adb shell getprop
ro.build.version.release')
        return format(out[1])

    def deviceid(self):
        out = subprocess.getstatusoutput('adb get-serialno')
        return format(out[1])

    def charge(self):
        print('battery information')
        out = subprocess.getstatusoutput('adb shell dumpsys battery')
        top = tk.Toplevel()
        top.title('battery information')
        top.geometry('%dx%d'% (600, 600))
```

```
        t = Text(top, width=600, height=600)
        t.insert('1.0', "{}".format(out[1]))
        t.pack()
        top.mainloop()

class HelpFrame(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.root = master
        self.createPage()

    def createPage(self):
        Label(self, text="None").grid(row=1, stick=W, pady=10)



from tkinter import *
from tkinter.messagebox import *
import tkinter as tk
from MainPage import *
import os
import re
import subprocess

class HomePage1(object):
    def __init__(self, master=None):
        self.root = master
        self.root.geometry('%dx%d'% (300, 180))
        self.createPage()

    def createPage(self):
        self.page = Frame(self.root)
        self.page.pack()
        Button(self.page, text='Connect to phone',width=12,
command=self.loginCheck).grid(row=1, stick=W, pady=10,column=1)
```

```
        Button(self.page, text='Quit',width=12,
command=self.page.quit).grid(row=2, column=1, stick=E)

    def loginCheck(self):
        out = subprocess.getstatusoutput('adb devices -l')
        out1 = subprocess.getstatusoutput('adb devices')
        if out[0]==0:
            if'device product' in out[1]:
                self.page.destroy()
                MainPage(self.root)
            else:
                self.sayTry()
        else:
            if out1[0]==0:
                if'device' in out1[1]:
                    self.page.destroy()
                    MainPage(self.root)
                else:
                   self.sayTry()
            else:
                self.sayNoadb()
    def connectPhone(self):
        self.page.destroy()

    def sayTry(self):
        tk.messagebox.showinfo("Message", "Phone connection failed,
please try to reconnect")

    def sayFail(self):
        tk.messagebox.showinfo("Message", "Mobile phone connection
failed, unknown error")
    def sayNoadb(self):
        tk.messagebox.showinfo("Message", "adb is not installed or
the adb environment variable is not configured")
```

Додаток Г

ІЛЮСТРАТИВНА ЧАСТИНА

Розробка методу і програмного засобу відлагодження пристроїв під керування операційної системи Android

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра ПЗ

Магістрська дипломна робота на тему:
Розробка методу і програмного засобу відлагодження пристроїв під керування операційної системи Android
Спеціальності: 121 – Інженерія програмного забезпечення

ВИКОНАВ: СТУДЕНТ 2-ГО КУРСУ ГРУПИ 1ПІ-20М
САВИЦЬКИЙ ДМИТРО СЕРГІЙОВИЧ
КЕРІВНИК: К.Т.Н., ДОЦЕНТ КАФЕДРИ ПЗ
к.т.н., доц. каф. ПЗ Ракитянська Г.Б

Рисунок Г.1 – Тема роботи

Мета та наукова новизна роботи

Мета та завдання дослідження. Метою розробки є підвищення надійності процесу відлагодження та розширення функціональних можливостей за рахунок нового методу збору та опрацювання даних, шляхом розробки та програмної реалізації алгоритмів відлагодження.

Наукова новизна отриманих результатів. Подальшого розвитку дістав метод відлагодження мобільних пристроїв, який, на відміну від існуючих, оптимізує зв'язок мобільного пристрою та робочого комп'ютера розробника та забезпечує підвищення ефективності та надійності процесу пошуку та виправлення проблем та суттєво знижує ризики, пов'язані з недостатньою якістю розроблюваних мобільних додатків.

Подальшого розвитку дістала модель системи, яка, на відміну від існуючих, забезпечує універсальний підхід до пристроїв та мобільних додатків, що дозволяє скоротити час розробки, тестування та відлагодження додатків, знизити вартість процесу та підвищити його ефективність.

Рисунок Г.2 – Мета та наукова новизна

РОЗДІЛ І
ОБГУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКИ ЗАДАЧІ ДОСЛІДЖЕННЯ

Оскільки програмне забезпечення та електронні системи загалом стали складнішими, різноманітні загальні методи налагодження розширилися за допомогою нових методів виявлення аномалій, оцінки впливу та планування виправлення програмного забезпечення або повного оновлення системи.

Налагодження – це процес виявлення та видалення наявних і потенційних помилок (також званих «багами») у програмному коді, які можуть призвести до неочікуваної поведінки або збою системи. Щоб запобігти некоректній роботі програмного забезпечення або системи, налагодження використовується для пошуку та усунення помилок або дефектів. Налагодження є важливою частиною визначення, чому операційна система, програма чи програма працюють неправильно. Коли різні підсистеми або модулі тісно пов'язані, налагодження стає значно важчим, оскільки будь-яка зміна в одному модулі може призвести до появи нових помилок в іншому.

Рисунок Г.3 – Розділ 1

Аналіз існуючих рішень

Згідно таблиці порівняльних характеристик можна побачити, що розробка програмного продукту є доцільною. В виконання проекту маємо отримати продукт, що буде являти собою краще оптимізовану і біль універсальною версією на відміну від існуючих рішень.

Параметр	Додаток			
	Android Remote Debugger	Android Studio	Android Debugging Bridge	SuperDebug
Графічний інтерфейс	1	1	0	1
Універсальність	0	1	1	1
Кросплатформовість	1	1	1	1
Низька ресурсозатратність	0	0	1	1
Загальний коефіцієнт	2	3	3	4

Рисунок Г.4 – Аналіз існуючих рішень

Висновок до першого розділу

У результаті аналізу аналогів розроблюваного додатку було зроблено висновок про доцільність розробки власного додатку для відлагодження пристроїв під керування операційної системи Android. Було розроблено основні завдання, які є необхідними для розробки програмного продукту.

У результаті аналізу об’єктно-орієнтованих мов програмування було обрано мову програмування Python для розробки програмного продукту.

У результаті аналізу середовищ розробки було обрано середовище розробки Microsoft Visual Studio Code.

Рисунок Г.5 – Висновок до першого розділу

РОЗДІЛ 2
РОЗРОБКА МЕТОДУ ВІДЛАГОДЖЕННЯ І МОДЕЛІ ПРОГРАМНОГО ДОДАТКУ

Для того, щоб відагодити програму, користувач повинен почати з проблеми, виділити вихідний код проблеми, а потім виправити її. Користувач програми повинен знати, як вирішити проблему, оскільки очікується знання про аналіз проблеми. Коли помилку виправлено, програмне забезпечення готове до використання. Інструменти відлагодження (так звані відлагоджувачі) використовуються для виявлення помилок кодування на різних етапах розробки. Вони використовуються для відтворення умов, в яких сталася помилка, а потім для вивчення стану програми на той момент і визначення причини. Програмісти можуть відстежувати виконання програми крок за кроком, оцінюючи значення змінних, і зупиняти виконання там, де це потрібно, щоб отримати значення змінних або скинути змінні програми.

Рисунок Г.6 – Розділ 2

Завдяки розроблюваній програмній системі, є можливість враховувати декілька різних факторів, що виникають під час розробки та тестування мобільних додатків, в програмі поєднується декілька функцій та методів, щоб підвищити надійність та ефективність процесу відлагодження. Такий підхід буде більш привабливим для потенційних користувачів, адже він об'єднує в собі декілька різних підходів, для чого раніше вимагалось використання декількох програмних систем.

Помилка	Частота виникнення	Ймовірність виявлення і виправлення	Витрати на тестування і доробку
Помилки в графічному інтерфейсі додатку	Дуже часто	0.95 – 0.99	Низькі
Серверні помилки	Дуже часто	0.90 – 0.95	Середні
Помилки в передачі даних між клієнтом та серверною частиною	Часто	0.70 – 0.80	Середні
Помилки в бізнес-логіці та функціях додатку	Часто	0.80 – 0.99	Середні
Системні помилки та аварійне завершення програми	Нечасто	0.99 – 1	Високі
Помилки безпеки пристрою та додатку	Рідко	0.3 – 0.7	Дуже високі
Витоки пам'яті	Рідко	0.3 – 0.5	Високі
Висока енерговитрата спричинена функціоналом додатку	Дуже рідко	0.4 – 0.6	Високі

Рисунок Г.7 – Модель системи

Висновок до другого розділу

В даному розділі було описано методи відлагодження різних мобільних пристроїв та додатків, призначених для них. На основі описаних методів побудовано вдосконалену модель відлагодження цих пристроїв. Наведено діаграми процесу, таблиця поширеності помилок при розробці мобільних додатків та формати системних даних які повертає пристрій.

Наведено та описано схему роботи програми, у вигляді діаграм компонентів та класів, а також розглянуто архітектуру системи, обрано патерн (шаблон) проектування для розробки програмної системи.

Рисунок Г.8 – Висновок до другого розділу

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ДОДАТКУ

Графічний візуальний інтерфейс користувача – тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, що засновані на використанні тексту, текстовому наборі команд та текстовій навігації.

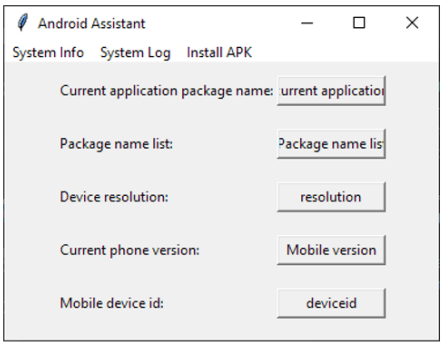

Рисунок Г.9 – Розділ 3

Розробка модулів програмного додатку

Кожне вікно програми має свої події (наприклад натискання кнопки в межах вікна, чи натискання певної клавіші на клавіатурі). Через подію програма викликає певні методи програми. Наприклад, подія початку процесу запису логі викликає метод logcat(), що оцінює орієнтовну кількість файлів в теках для правильної роботи індикатору прогресу.

При запуску додатку створюється головне вікно та ініціалізуються основні методи програмного додатку. Далі додаток обробляє введений користувачем запит і здійснює пошук, виводячи результат(перелік тек та файлів) у відповідне поле головного віка.

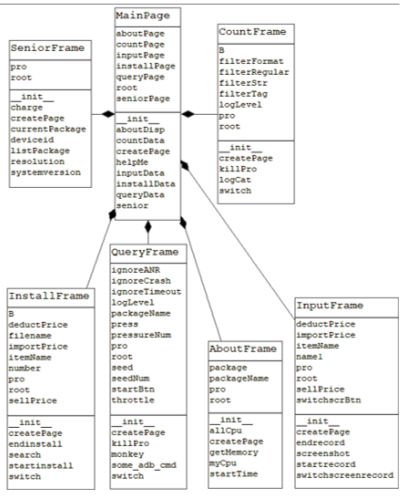

Рисунок Г.10 – Розробка модулів програмного додатку

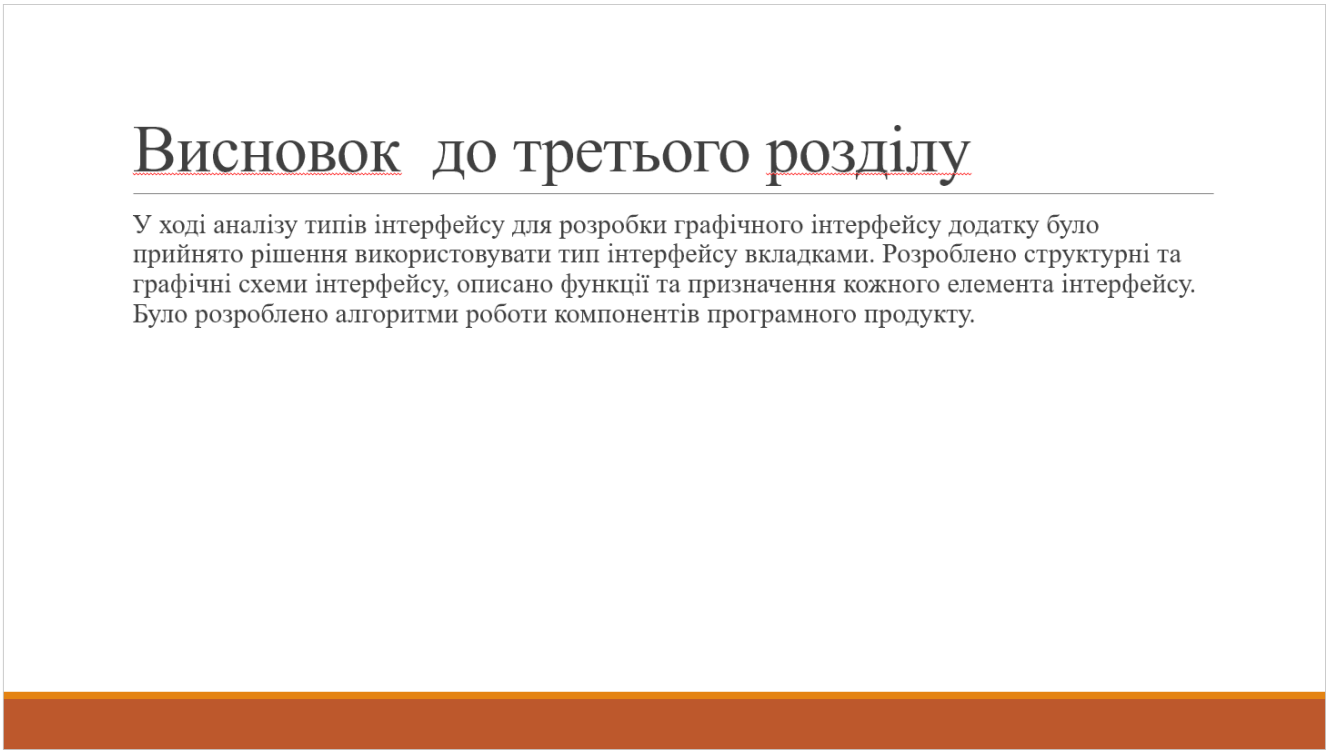

Рисунок Г.11 – Висновок до третього розділу

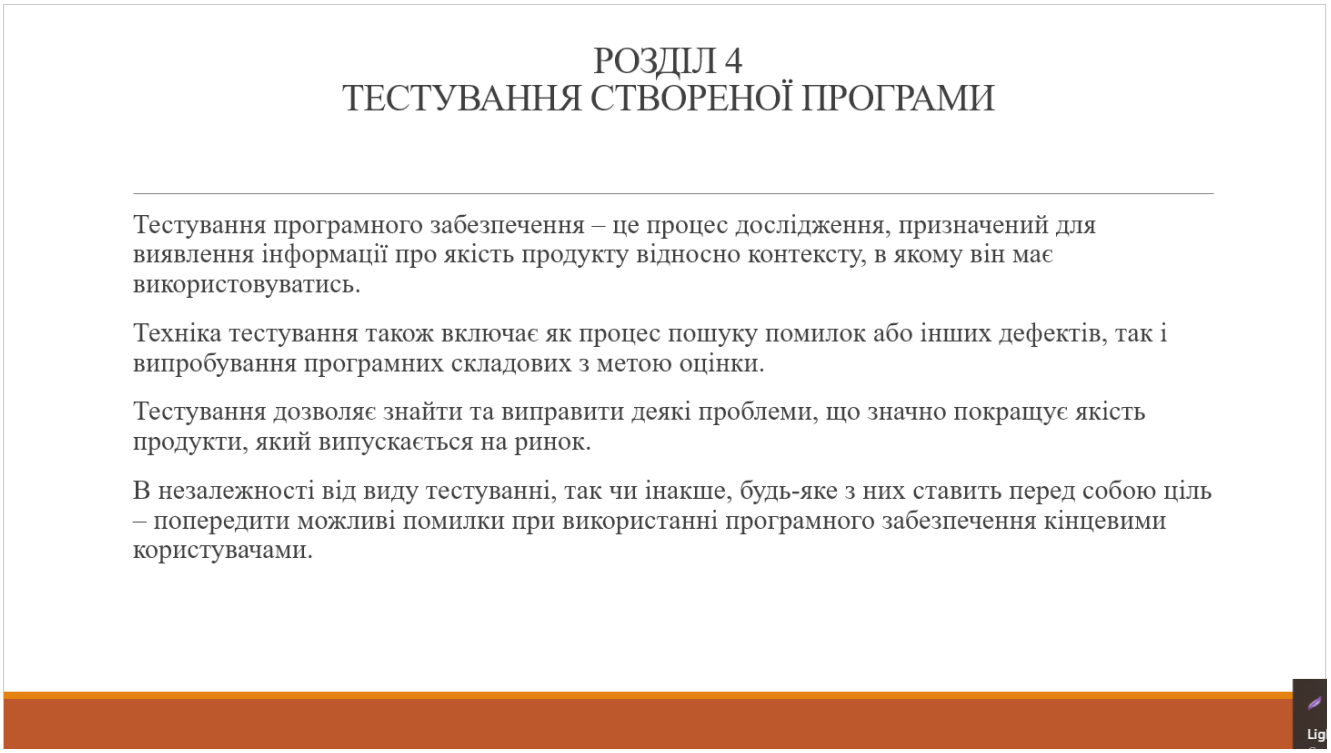

Рисунок Г.12 – Розділ 4

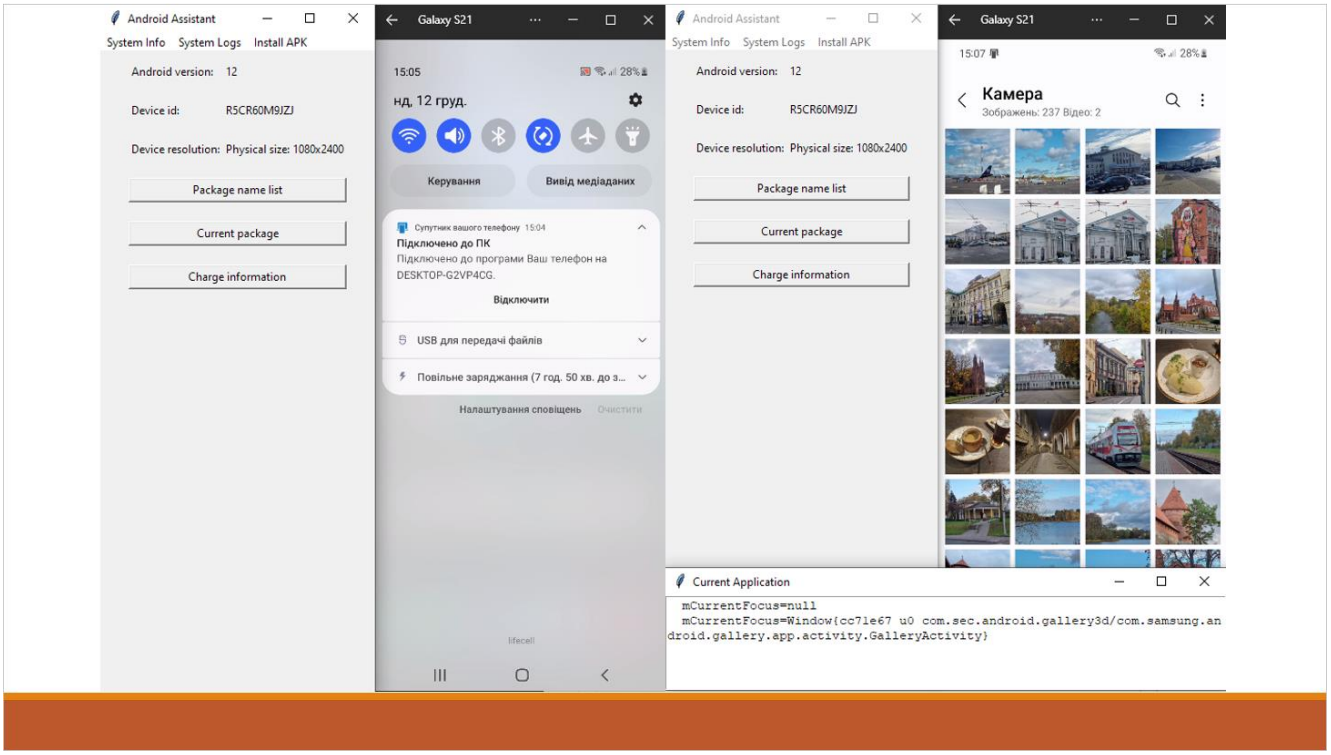

Рисунок Г.13 – Приклади роботи

Висновок до четвертого розділу

В цьому розділі було проведено тестування як в ручному так і в автоматичному режимі, тестування програми показало повну її працездатність там ефективність роботи алгоритмів реалізованих в програмі.

Згідно аналізу роботи програмного додатку на різних системах були складені системні вимоги для двох можливих конфігурацій персональних комп'ютерів користувачів. Ці вимоги відповідають приблизним сучасним конфігураціям комп'ютерів, тому проблем з реалізацією додатку не має бути.

Рисунок Г.14 – Висновки до четвертого розділу

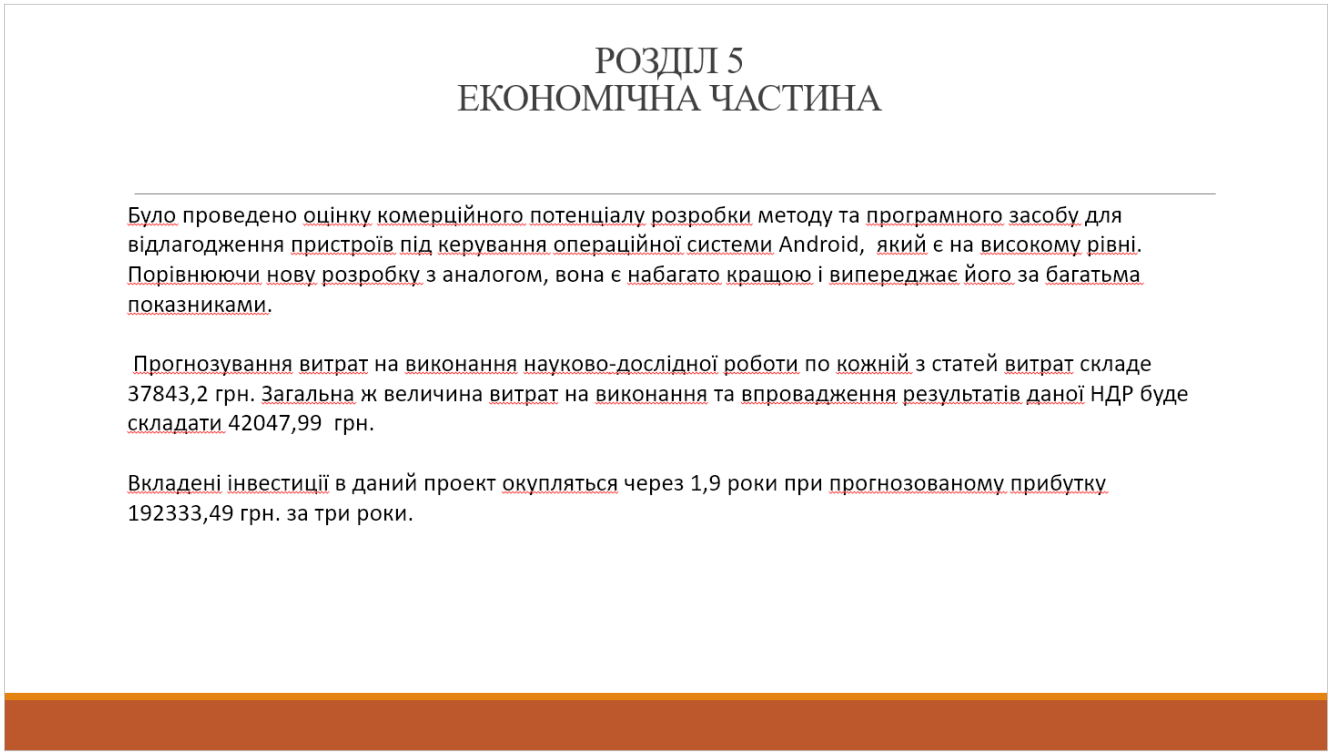

Рисунок Г.15 – Розділ 5

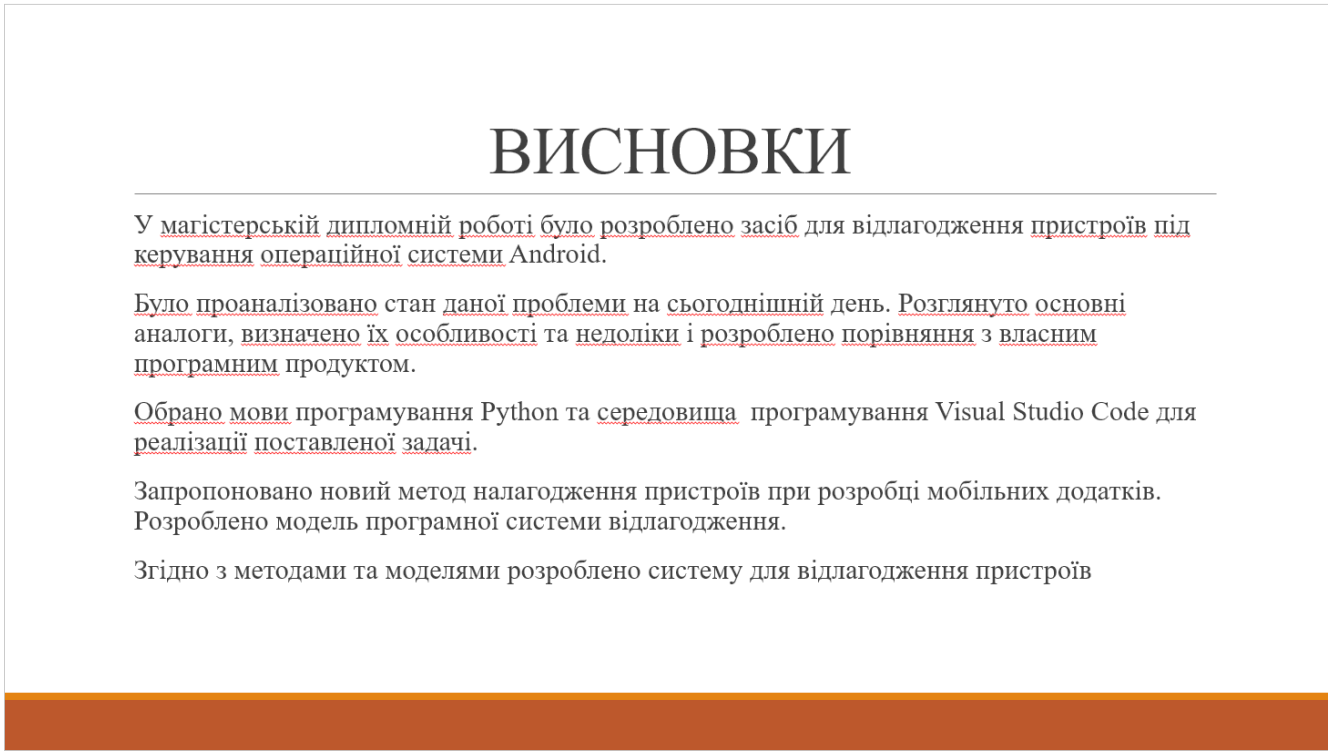

Рисунок Г.16 - Висновки