Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка програмного забезпечення для налаштування Windows з використанням системного реєстру»

Виконав: студент 4 курсу
групи 1ПІ-21б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Ткачук В.І.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Майданюк В.П.
(прізвище та ініціали)
Рецензент: д.ф., ст. викл. каф. ОТ Обертюх М. Р.
(прізвище та ініціали)

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
«09» 06 2025 р.

Вінниця ВНТУ – 2025 рік

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н._____
«21» березня 2025 р.

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

Ткачуку Владиславу Іллічу

1. Тема роботи – «Розробка програмного забезпечення для налаштування Windows з використанням системного реєстру»

Керівник роботи: Майданюк Володимир Павлович, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: операційна система – Windows; середовище розробки – Visual Studio Code; мова програмування – Python.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану розвитку програмних застосунків для редагування реєстру та обгрунтування задач розробки; розробка методів, моделі та алгоритму роботи програмного забезпечення; розробка програмного забезпечення; тестування програмного забепечення; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження, новизна, практична цінність одержаних результатів; аналоги; порівняльний аналіз аналогів; використані технології; тестування програмного забезпечення; висновки; публікації.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Майданюк В.П., к.т.н., доцент кафедри ПЗ	24.03.25	02.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану розвитку програмних застосунків для редагування реєстру та обгрунтування задач розробки	25.03.25 – 31.03.25	вик.
2	Розробка методів, моделі та алгоритму роботи програмного забезпечення	03.04.25 – 14.04.25	вик.
3	Варіантний аналіз та вибір мови програмування розробки	17.04.25 – 21.04.25	вик.
4	Розробка програмного забезпечення	24.04.25 – 12.05.25	вик.
5	Тестування програмного забезпечення	15.05.25 – 19.05.25	вик.
6	Оформлення матеріалів до захисту БКР	22.05.25 – 30.05.25	вик.

Студент _____ Ткачук В. І.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Майданюк В. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.4

Ткачук В. І. Розробка програмного забезпечення для налаштування Windows з використанням системного реєстру : бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 58 с.

На укр. мові. Бібліогр. : 26 назв ; рис. : 28; табл. 3.

У бакалаврській кваліфікаційній роботі проведено аналіз методів доступу до системного реєстру операційної системи Windows, а також досліджено основні категорії параметрів, що впливають на конфіденційність, персоналізацію та продуктивність системи.

Розроблено програмне забезпечення з графічним інтерфейсом користувача, що реалізує функції резервного копіювання, зміни системних параметрів та ведення журналу змін. Для реалізації інтерфейсу використано мову програмування Python та бібліотеку PyQt5, а для взаємодії з системним реєстром – модуль winreg.

Розроблений додаток підвищує зручність і безпеку налаштування системи Windows та може бути використаний як кінцевими користувачами, так і в освітніх цілях при вивченні основ адміністрування.

Ключові слова: Windows, Python, PyQt5, системний реєстр, налаштування, безпека, графічний інтерфейс.

ABSTRACT

UDC 004.4

Tkachuk V.I. Development of software for Windows configuration using the system registry. Vinnytsia: VNTU, 2025. 58 p.

In Ukrainian language. Bibliography: 26 titles ; fig. : 28 ; tabl. 3.

In the bachelor's thesis, an analysis of access methods to the Windows system registry was conducted, and typical categories of settings affecting privacy, personalization, and system performance were studied.

Software with a graphical user interface was developed, which implements functions for backup, parameter modification, and change logging. The interface was developed using Python programming language and the PyQt5 library, while interaction with the registry was implemented using the winreg module.

The developed application enhances the convenience and security of the Windows configuration process and can be useful both for end users and in educational settings when studying the basics of Windows administration.

Keywords: Windows, Python, PyQt5, system registry, configuration, security, graphical interface.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний етап розвитку комп’ютерних технологій характеризується зростаючою складністю операційних систем, а також підвищеними вимогами до їх безпеки, стабільності та персоналізації. У цьому контексті особливого значення набуває можливість гнучкого керування параметрами операційної системи Windows. Одним із ключових механізмів конфігурування системи є системний реєстр – централізована база даних, яка містить критично важливу інформацію про конфігурацію самої ОС, драйверів та встановлених програм [1].

Редагування системного реєстру за допомогою стандартних засобів Windows потребує глибоких технічних знань і передбачає певний рівень ризику. Некоректні дії можуть призвести до збоїв у роботі системи або навіть до її повного виходу з ладу. У зв’язку з цим виникає потреба у створенні спеціалізованого програмного забезпечення, яке забезпечуватиме зручний, інтуїтивно зрозумілий та безпечний інтерфейс для взаємодії користувача із системним реєстром.

Зростання значущості конфіденційності, продуктивності та кастомізації системи робить питання налаштування Windows все більш актуальним навіть для пересічного користувача. Надання доступу до ключових параметрів у зручній формі дозволяє підвищити рівень контролю над системою, оптимізувати її роботу, зменшити навантаження на ресурси та забезпечити захист персональних даних.

Особливої важливості набуває реалізація функціоналу резервного копіювання та фіксації змін, що дозволяє мінімізувати ризики та забезпечує можливість швидкого відновлення системних параметрів у разі виникнення помилок. На сьогоднішній день більшість існуючих рішень або є платними, або мають надмірно складний інтерфейс, що обмежує їх використання недосвідченими користувачами [2].

Таким чином, актуальність обраної теми обумовлена необхідністю створення доступного, безпечного та функціонального інструменту для гнучкого налаштування параметрів Windows із використанням системного реєстру. Це

забезпечує підвищення зручності та безпеки користування операційною системою без потреби в спеціальних технічних знаннях.

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

Мета та завдання дослідження. Метою роботи є підвищення зручності та безпеки процесу налаштування операційної системи Windows шляхом розробки програмного забезпечення з інтуїтивним графічним інтерфейсом, що дозволяє змінювати параметри системи через системний реєстр.

Основними задачами дослідження є:

– розробити архітектуру додатку;

– розробити користувацький інтерфейс;

– розробити модуль для взаємодії з системним реєстром т;

– розробити модуль для введення журналу змін;

– розробити алгоритми роботи застосунку;

– провести тестування розробленого застосунку.

Об'єкт дослідження – процес розробки програмного забезпечення для налаштування параметрів операційної системи Windows за допомогою системного реєстру.

Предмет дослідження – методи та засоби програмної взаємодії з системним реєстром Windows для зміни системних параметрів.

Методи дослідження. У процесі досліджень використовувались: теорія алгоритмів, методи взаємодії з системним реєстром, методи побудови графічного інтерфейсу, методи тестування програмного застосунку.

Новизна отриманих результатів. Подальшого розвитку отримав метод розробки програмного забезпечення для налаштування операційної системи Windows, який, на відміну від існуючих, забезпечує централізований доступ до широкого спектру параметрів системного реєстру з можливістю резервного копіювання, відновлення та ведення журналу змін, що підвищує надійність і безпечність внесення змін.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в бакалаврській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для налаштування Windows з використанням системного реєстру.

Особистий внесок здобувача. Усі результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. У друкованій праці [3], опублікованій у співавторстві, автору належать такі результати: налаштування Windows з використанням системного реєстру.

Апробація результатів роботи. Основні положення бакалаврської кваліфікаційної роботи доповідалися та обговорювалися на конференції «IV Міжнародна науково-практична конференція «Інформаційні технології в освіті та науці» (Вінниця, 2025 р.).

Публікації. Результати роботи були опубліковані в матеріалах конференції «IV Міжнародна науково-практична конференція «Інформаційні технології в освіті та науці» (Вінниця, 2025 р.).

1 АНАЛІЗ СТАНУ РОЗВИТКУ ПРОГРАМНИХ ЗАСТОСУНКІВ ДЛЯ РЕДАГУВАННЯ РЕЄСТРУ ТА ОБГРУНТУВАННЯ ЗАДАЧ РОЗРОБКИ

1.1 Аналіз стану розвитку програмних застосунків для редагування реєстру

Сучасний розвиток комп'ютерних технологій і операційних систем призводить до зростання потреб користувачів у гнучкому, точному та безпечному налаштуванні середовища Windows відповідно до власних вимог.

Традиційна робота з системним реєстром за допомогою редактора реєстру (regedit.exe) є складною та потенційно небезпечною – навіть незначна помилка може призвести до порушення стабільності системи. У зв'язку з цим виникає потреба у створенні програмних засобів, які забезпечать безпечну взаємодію з реєстром через зручний графічний інтерфейс [1].

Окремі сторонні утиліти надають подібний функціонал, проте часто мають суттєві недоліки: обмежений набір функцій, відсутність можливості створення резервних копій, закритий код, відсутність підтримки української мови, а також платна ліцензія.

Потреба в інструменті з відкритою архітектурою, модульною структурою, підтримкою локалізації, резервного копіювання і логування змін – усе це формує основу для розробки нового рішення, орієнтованого на кінцевого користувача.

Розробка такого застосунку дозволяє не лише поліпшити зручність роботи із системою Windows, а й підвищити безпеку внесення змін, зберігаючи гнучкість і контроль за кожною дією. Крім того, впровадження тем оформлення, розподіл функціоналу за категоріями та простий механізм відкату змін сприяють покращенню користувацького досвіду та персоналізації ОС [2].

Таким чином, аналіз сучасного стану засобів для налаштування Windows вказує на актуальність розробки власного інструменту, який поєднує відкритість, зручність, функціональність і безпечну взаємодію з системним реєстром. Це дозволяє користувачам швидко адаптувати систему до своїх потреб, не ризикуючи її стабільністю.

1.2 Порівняльний аналіз аналогів

Наявні програмні засоби, які взаємодіють із системним реєстром Windows, мають низку обмежень – як функціональних, так і технічних.

Серед найбільш відомих аналогів можна відзначити такі застосунки:

– Winaero Tweaker;

– O&O ShutUp10++;

– RegistryEditor;

– Glary Utilities;

– Advanced SystemCare;

Winaero Tweaker – багатофункціональний безкоштовний застосунок для зміни системних параметрів, в основному через редагування реєстру [4]. Має велику кількість налаштувань, але не підтримує створення резервних копій і має складну навігацію (див. рисунок 1.1).

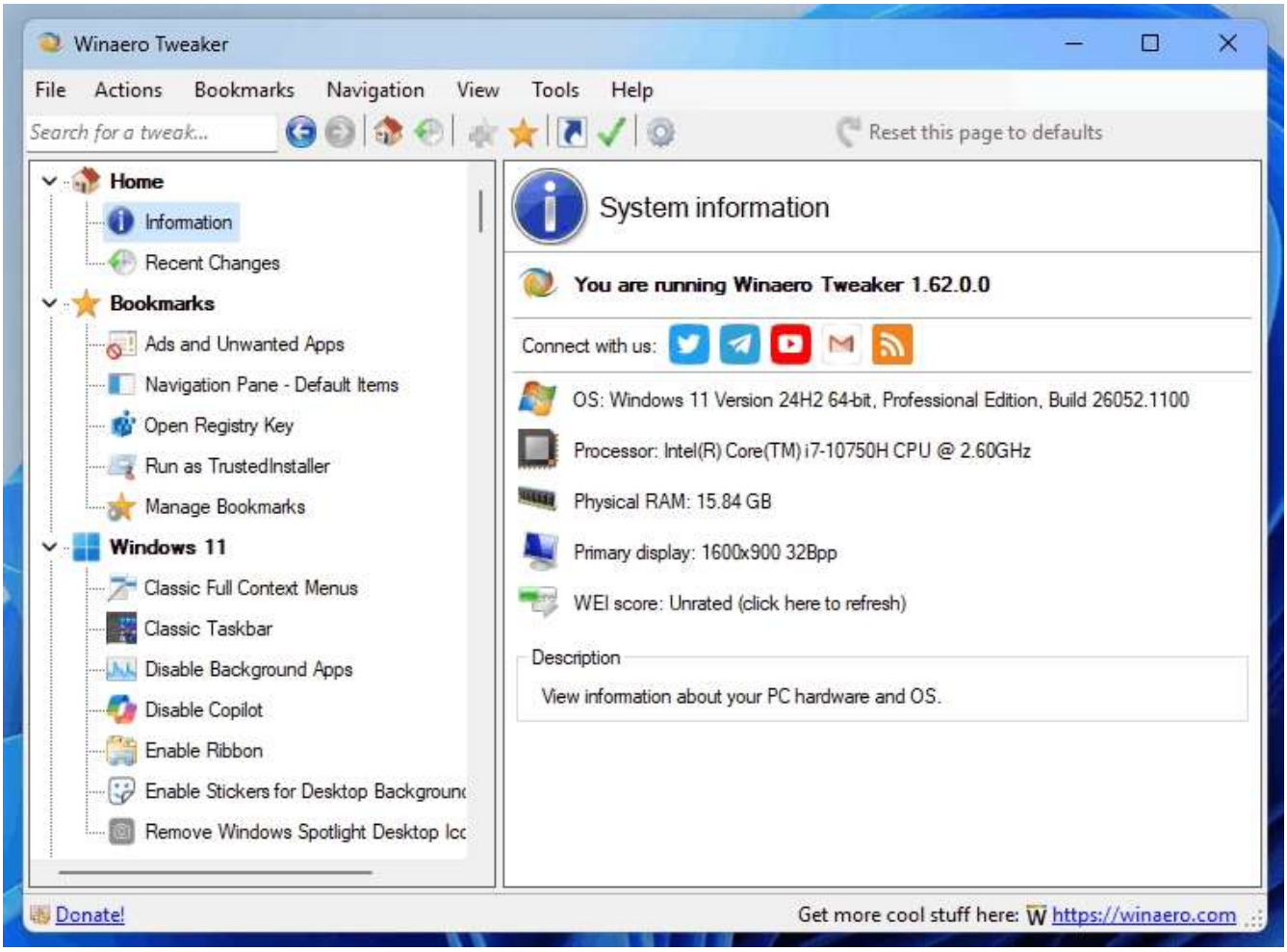

Рисунок 1.1 – Приклад роботи застосунку Winaero Tweaker

O&O ShutUp10++ – утиліта для керування параметрами конфіденційності в Windows (див. рисунок 1.2). Основний фокус – відключення телеметрії та небажаних сервісів. Програма зручна у використанні, але обмежена виключно параметрами конфіденційності, не надає змоги кастомізувати інтерфейс чи оптимізувати систему [5].

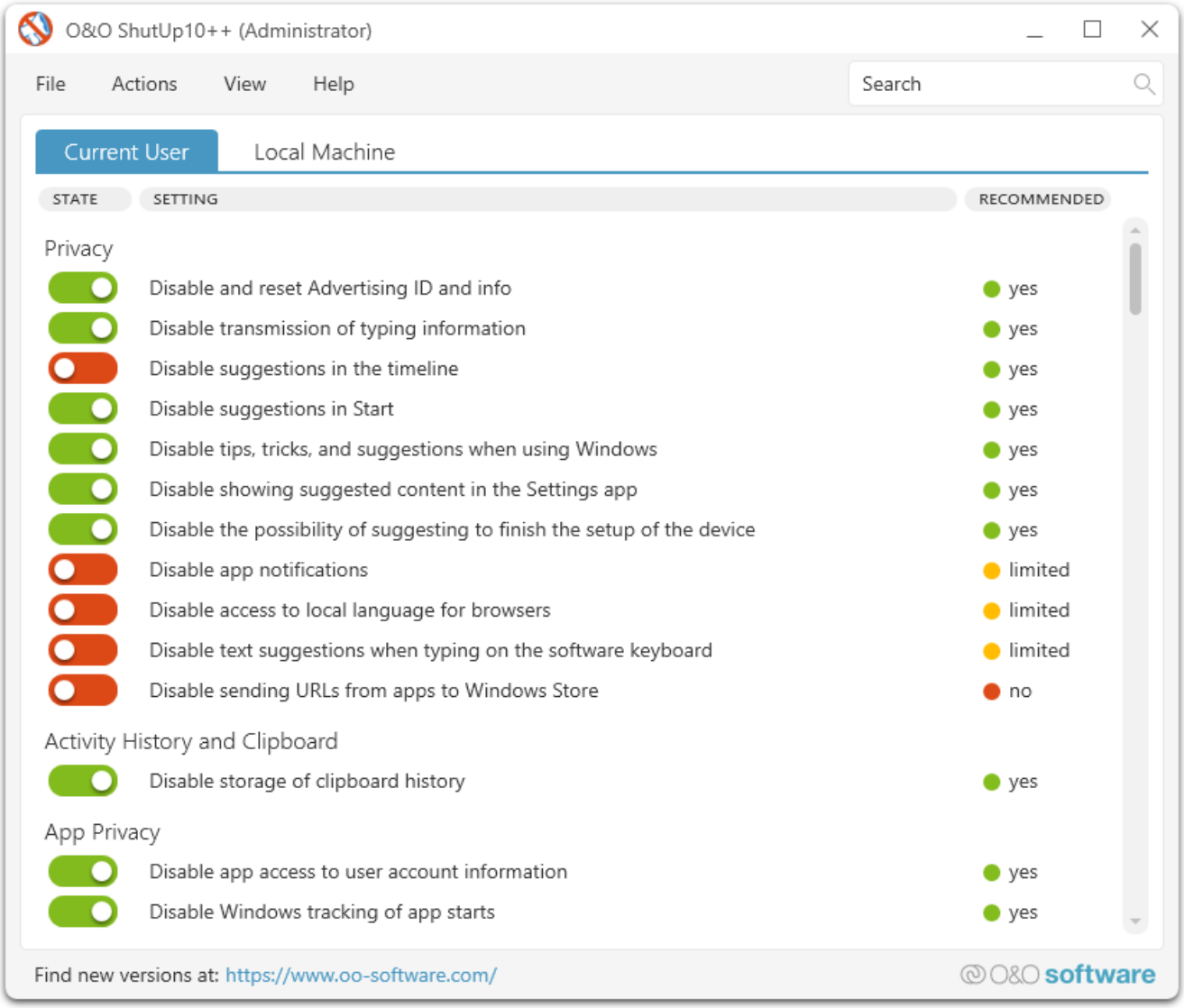

Рисунок 1.2 – Головне вікно O&O ShutUp10++

RegistryEditor – це стандартний вбудований інструмент операційної системи Windows, який надає повний доступ до системного реєстру (див. рисунок 1.3). Його інтерфейс представляє собою ієрархічне дерево ключів, у якому користувач може вручну змінювати, створювати або видаляти записи [6].

Вбудований інструмент операційної системи дозволяє виконувати всі базові операції з реєстром, пошук значень, редагування параметрів різних типів (DWORD, STRING, BINARY тощо). Однак він не передбачає захисту від помилок, не попереджає про можливі наслідки, не веде журнал змін та має незручний інтерфейс – це робить його потужним, але небезпечним інструментом у руках недосвідченого користувача.

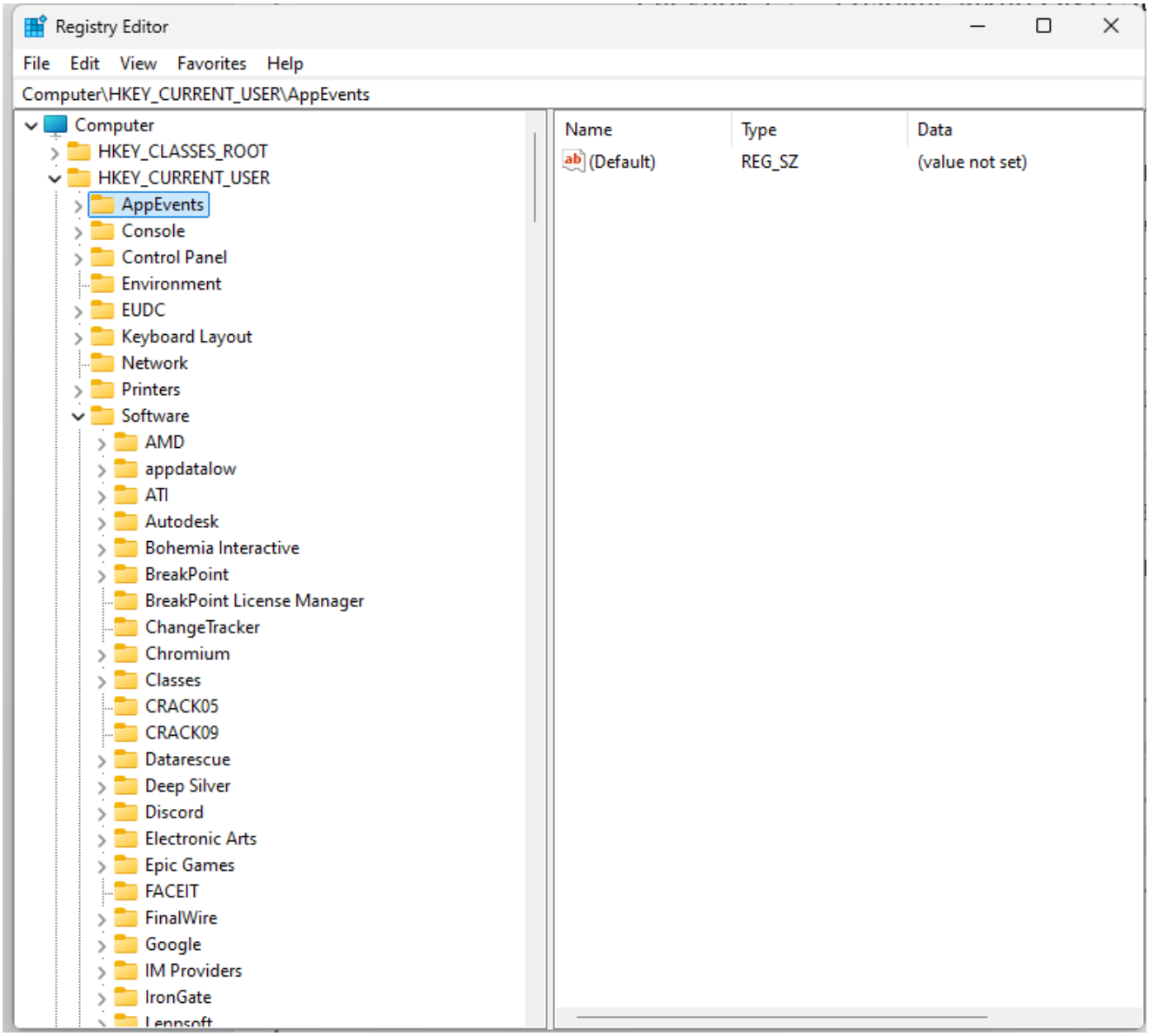

Рисунок 1.3 – Приклад роботи застосунку RegistryEditor

Glary Utilities – це універсальний інструмент для обслуговування та оптимізації операційної системи Windows (див. рисунок 1.4). Програма об'єднує у собі функції очищення реєстру, управління автозавантаженням, дефрагментації

дисків, пошуку дубльованих файлів та інших сервісних задач. Glary Utilities має інтуїтивно зрозумілий інтерфейс і підтримує пакетне обслуговування системи за один клік. Однак, більшість розширених функцій доступні лише у платній версії, що обмежує можливості безкоштовного користування [7].

Рисунок 1.4 – Головне вікно Glary Utilities

Advanced SystemCare – це комплексне програмне забезпечення для оптимізації та прискорення роботи Windows-систем (див. рисунок 1.5). Основний функціонал включає очищення реєстру, оптимізацію параметрів мережі, захист конфіденційності, оновлення драйверів та інші інструменти для підвищення продуктивності ПК. Інтерфейс програми простий і дозволяє виконувати базове обслуговування системи у декілька кліків. Як і у випадку з Glary Utilities, значна частина розширених можливостей доступна тільки у преміум-версії [8].

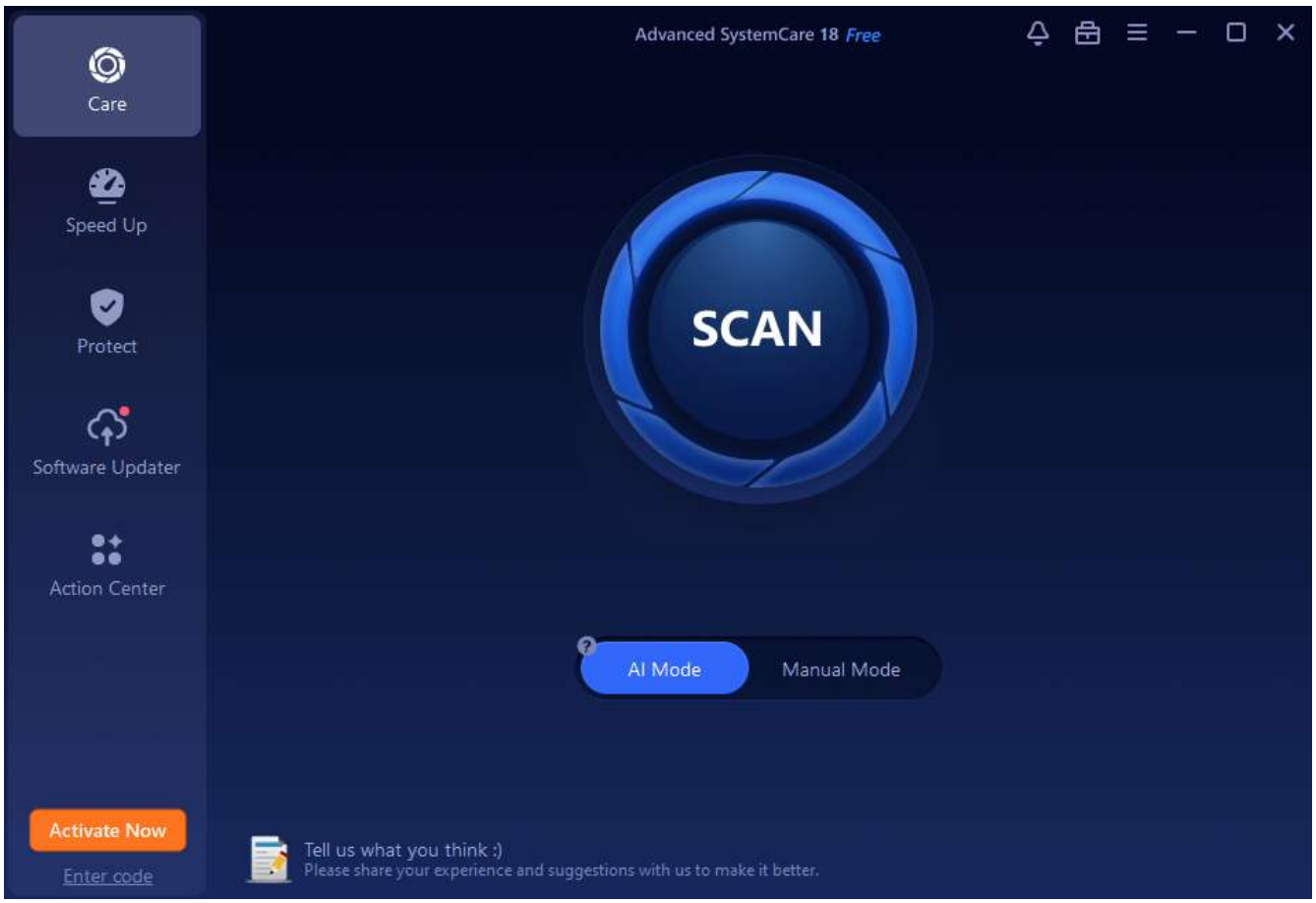

Рисунок 1.5 – Приклад роботи застосунку Advanced SystemCare

Було зроблено порівняльну характеристику розроблюваного продукту та наявних аналогів, результати порівняння зведено в таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	Winaero Tweaker	O&O ShutUp 10++	Registry Editor	Glary Utilities	Advanced SystemCare	Власна розробка
Підтримка резервного копіювання	-	+	-	+	+	+
Логування змін	-	-	-	-	-	+
Пояснення до кожного параметра	+	+	-	+	+	+

Продовження таблиці 1.1

Критерій	Winaero Tweaker	O&O ShutUp 10++	Registry Editor	Glary Utilities	Advanced SystemCare	Власна розробка
Можливість зміни теми (світла, темна)	+	-	+	-	+	+
Створення резервної копії за категорією налаштувань	-	-	-	-	-	+
Загальна оцінка	40%	40%	20%	40%	60%	100%

Як видно з таблиці, жоден із наявних аналогів не надає повного спектра можливостей для налаштування системи Windows, зокрема в аспектах резервного копіювання змін, логування, зміни тем оформлення та розділення функціональності за категоріями (оптимізація, конфіденційність, кастомізація). Саме тому доцільною є розробка власного застосунку, який би враховував ці аспекти й забезпечував безпечну та зручну роботу з системним реєстром.

Таким чином, створення універсального та доступного засобу для конфігурації Windows через реєстр є актуальним і перспективним напрямком розробки програмного забезпечення, орієнтованого на кінцевого користувача.

1.3 Аналіз методів розв’язання задачі

Розробка програмного забезпечення для налаштування параметрів операційної системи Windows через системний реєстр потребує комплексного підходу до вирішення завдань, пов’язаних із взаємодією з критичними системними компонентами, графічною інтерфейсною взаємодією та забезпеченням безпеки змін. Існує кілька підходів до розробки подібних програмних рішень, кожен з яких має свої переваги і недоліки. Вибір оптимального методу багато в чому залежить від поставлених цілей, обмежень проєкту, вимог до зручності користування, масштабованості, швидкодії та рівня

технічної підготовки кінцевого користувача.

Одним із традиційних методів є створення консольних утиліт або скриптів (наприклад, PowerShell або .bat-файли), які змінюють налаштування реєстру напряму. Цей підхід швидкий у реалізації, проте має суттєві обмеження – він не забезпечує зручний інтерфейс, не підтримує модульність та резервне копіювання, а також не забезпечує зворотний зв’язок з користувачем. Крім того, відсутність візуального представлення дій користувача часто ускладнює роботу з такими інструментами для пересічного користувача, підвищує ймовірність помилкових змін і, як наслідок, порушення стабільності роботи ОС.

Альтернативним підходом є використання мов програмування високого рівня (наприклад, C++, C# або Python) у поєднанні з графічними бібліотеками. Такий підхід дозволяє реалізувати гнучке, масштабоване та зручне у використанні рішення з підтримкою GUI. Зокрема, у рамках кваліфікаційної роботи було обрано мову програмування Python та бібліотеку PyQt5 для реалізації графічного інтерфейсу. Такий вибір зумовлений високою читабельністю коду, великою кількістю готових бібліотек, активною спільнотою підтримки та хорошою інтеграцією з Windows API через модуль winreg.

Робота з системним реєстром у Python здійснюється за допомогою стандартного модуля winreg, який дозволяє читати, записувати та створювати ключі у гілках реєстру. Основна перевага такого методу – повна сумісність з Windows API та можливість реалізації як простих, так і складних сценаріїв налаштування системи. Крім того, використання Python значно спрощує процес розробки, забезпечуючи швидку ітерацію прототипів, зменшуючи час на відлагодження та полегшуючи реалізацію логіки ведення журналу змін.

З метою впорядкування архітектури проєкту було реалізовано чітке розділення відповідальностей у програмному коді. Основу взаємодії із системним реєстром Windows становлять два ключові компоненти – клас для роботи з реєстром (RegistryManager) та клас для логування дій (Logger). Саме вони формують ядро програмного функціоналу застосунку. Такий поділ дозволяє уникнути дублювання коду, забезпечує більшу гнучкість при оновленнях, а також

спрощує виявлення та виправлення помилок під час тестування і налагодження.

Графічний інтерфейс програми структуровано за категоріями налаштувань: конфіденційність, оптимізація, кастомізація. Кожна категорія об’єднує набір змін, які користувач може внести до реєстру за допомогою відповідного інтерфейсу. Усі функції згруповані за логічними ознаками, що дозволяє користувачу інтуїтивно орієнтуватися в можливостях програми. Крім того, для кожної операції передбачено короткий опис її впливу на систему, що підвищує прозорість налаштувань.

Такий підхід до структурування застосунку забезпечує зрозуміле представлення функцій для користувача, зручність у підтримці проєкту, а також можливість розширення – наприклад, додавання нових категорій налаштувань або вдосконалення існуючих компонентів. Завдяки цьому програма легко адаптується до нових версій Windows або розширюється під потреби корпоративних користувачів без необхідності суттєвих змін у базовому коді.

У результаті обрана технологічна база та архітектурна модель дозволили створити програмне забезпечення, яке є не лише функціональним, але й безпечним та зрозумілим у використанні. Комплексність, модульність і прозорість реалізації стали визначальними факторами у формуванні надійного інструменту для тонкого налаштування параметрів Windows, орієнтованого на широкий спектр користувачів – від новачків до системних адміністраторів.

1.4 Постановка задач дослідження

Провівши детальний аналіз методів розв’язання поставленої задачі, було розглянуто як класичні, так і сучасні підходи до реалізації програмного забезпечення, яке взаємодіє із системним реєстром Windows. Було проаналізовано переваги та недоліки використання консольних скриптів, графічних інтерфейсів, а також різних мов програмування, що застосовуються у подібних проєктах. Особливу увагу приділено питанням безпеки, стабільності роботи, масштабованості архітектури та зручності користування кінцевим продуктом. Також було здійснено огляд існуючих програм-аналогів, що надають користувачам

функціональність налаштування параметрів Windows, зокрема утиліт на кшталт Winaero Tweaker, Ultimate Windows Tweaker та інших. Порівняння з цими інструментами дозволило визначити їхні сильні сторони, а також виявити обмеження, які слід враховувати при реалізації власного рішення. На основі проведеного аналізу методів та практичного досвіду роботи з аналогічними програмами було сформульовано такі основні задачі дослідження:

– розробити архітектуру додатку;

– розробити користувацький інтерфейс;

– розробити модуль для взаємодії з системним реєстром т;

– розробити модуль для введення журналу змін;

– розробити алгоритми роботи застосунку;

– провести тестування розробленого застосунку.

Технічне завдання на розробку наведено в додатку А.

1.5 Висновки

У першому розділі було розглянуто сучасний стан програмних засобів для налаштування параметрів операційної системи Windows через системний реєстр. Проаналізовано існуючі рішення, зокрема такі як Winaero Tweaker, O&O ShutUp10++, Glary Utilities, Advanced SystemCare та стандартний редактор реєстру Windows – Registry Editor. Кожен із цих застосунків має свої переваги та недоліки, які були ретельно досліджені.

2 РОЗРОБКА МЕТОДІВ, МОДЕЛІ ТА АЛГОРИТМУ РОБОТИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз інформаційного забезпечення програми

Інформаційна архітектура застосунку для налаштування параметрів операційної системи Windows через системний реєстр побудована на основі чітко визначених потоків даних [9]. Такий підхід дозволяє структурувати внутрішні процеси взаємодії програмного забезпечення з системними ресурсами, забезпечуючи узгодженість усіх етапів обробки даних – від моменту отримання вхідної інформації до формування кінцевого результату.

Вхідні дані включають поточні значення ключів реєстру Windows, які зчитуються при старті застосунку, обрані користувачем параметри для редагування (конфіденційність, оптимізація, кастомізація), а також інформацію про бажану тему оформлення інтерфейсу. У процесі взаємодії з користувачем ці дані постійно оновлюються відповідно до поточних дій, забезпечуючи гнучке та інтуїтивне налаштування системи.

Проміжні дані формуються у вигляді тимчасових копій змінених параметрів реєстру перед їх застосуванням. Вони зберігаються у пам’яті для попередньої перевірки, аналізу на конфлікти та забезпечення можливості скасування змін у разі виявлення помилок або некоректних значень. Цей підхід мінімізує ймовірність пошкодження критичних налаштувань операційної системи.

Вихідними даними є безпосередньо змінені записи реєстру, файли резервних копій змінених параметрів і журнал змін, що зберігається для аналізу та контролю. Журнал змін (лог-файл) фіксує усі ключові дії користувача: час і дату виконання, тип операції, старе і нове значення параметра, а також статус її завершення (успішно чи з помилкою). Наявність такого механізму логування є важливою складовою інформаційної безпеки та дозволяє здійснювати аудит внесених змін у разі потреби.

Така побудова потоків даних забезпечує керованість процесів налаштування системи, підвищення надійності та безпеки змін, а також можливість відкату до

попередніх станів [10]. Завдяки продуманій структурі взаємодії між модулями програми досягається високий рівень стабільності навіть при інтенсивному використанні програмного забезпечення.

Для ефективної роботи із системним реєстром застосунок використовує стандартні формати збереження даних:

– внутрішні формати представлені у вигляді Python-структур (словники, списки), що містять дані про шлях до ключа, тип значення та його нове значення; вони дозволяють здійснювати швидку обробку та передачу даних між модулями програми;

– файлові формати включають .reg для резервного копіювання налаштувань, а також .log, де фіксуються дії користувача. Це забезпечує як відновлення системи до попереднього стану, так і зручність у проведенні діагностики.

Інформаційне забезпечення розробленого застосунку для налаштування Windows являє собою оптимально побудовану структуру, що поєднує зручність взаємодії, безпечну обробку даних та мінімізацію ризиків під час роботи із критичними елементами операційної системи. Завдяки цьому користувачі отримують гнучкий, функціональний і водночас безпечний інструмент для тонкого налаштування системи відповідно до своїх потреб і вподобань.

2.2 Розробка методу взаємодії з системним реєстром

Системний реєстр Windows є ієрархічною базою даних, яка зберігає налаштування та параметри як самої операційної системи, так і встановлених програм. Робота з реєстром вимагає обережності, оскільки неправильне редагування його ключів може призвести до дестабілізації системи або її некоректної роботи. Саме тому для розробки програмного забезпечення, яке взаємодіє з реєстром, необхідно реалізувати надійний, безпечний і контрольований метод доступу до даних.

Основною вимогою до методу взаємодії з системним реєстром є забезпечення читання, створення, редагування та видалення ключів і значень з урахуванням прав доступу, типів даних (наприклад, REG_SZ, REG_DWORD,

REG_BINARY) та структури реєстру. Крім того, необхідно враховувати можливість попереднього збереження резервної копії перед внесенням змін, а також ведення журналу для відстеження усіх дій користувача.

У межах розробки даного застосунку було використано стандартний модуль winreg мови програмування Python, який надає інтерфейс до API Windows Registry. Завдяки цьому модулю забезпечується прямий доступ до основних гілок реєстру (HKEY_LOCAL_MACHINE, HKEY_CURRENT_USER, тощо), а також підтримка основних операцій: відкриття ключів, читання значень, створення нових записів і видалення існуючих.

Для організації коду було реалізовано клас RegistryManager, який інкапсулює всі функції взаємодії з реєстром. Такий підхід дозволяє відокремити логіку роботи з системним реєстром від інших частин програми, спрощуючи обслуговування та розширення застосунку. Кожен метод класу має обробку винятків (try-except) для запобігання аварійного завершення роботи програми у разі помилок доступу чи некоректних параметрів.

Таким чином, запропонований метод взаємодії з системним реєстром забезпечує безпечне, контрольоване та зручне внесення змін до конфігурацій Windows, що є критично важливим для створення стабільного інструменту налаштування операційної системи. Завдяки поєднанню структурованого підходу, обробки винятків і чіткого розмежування функціональних компонентів, досягається висока стійкість програми до помилок, збереження цілісності даних та зменшення ризиків, пов’язаних із втручанням у критичні параметри системи. Це дозволяє користувачу зосередитися на налаштуванні необхідних функцій без загрози порушення стабільної роботи ОС.

2.3 Розробка методу побудови графічного інтерфейсу

Графічний інтерфейс користувача (GUI) є критичним компонентом програмного забезпечення, який забезпечує інтуїтивно зрозумілу взаємодію користувача з функціоналом програми. У контексті розробки застосунку для зміни параметрів операційної системи Windows через системний реєстр особливо

важливим є створення зручного, адаптивного та логічно структурованого інтерфейсу, що дозволяє користувачеві швидко знаходити необхідні налаштування та контролювати зміни без зайвого ризику.

Для реалізації графічного інтерфейсу було обрано бібліотеку PyQt5, яка є однією з найпотужніших і водночас доступних для використання бібліотек у середовищі Python. Вона дозволяє створювати кросплатформені GUI-застосунки із широкими можливостями кастомізації, підтримкою подій, валідації введення, і роботи з шаблонами інтерфейсу.

Архітектурно інтерфейс побудовано за принципом поділу на функціональні категорії: конфіденційність, оптимізація та кастомізація. Для кожної категорії створено окрему вкладку, яка містить групи параметрів із відповідними перемикачами (CheckBox), поясненнями (Label), та кнопками дій (Button). Такий поділ дозволяє зменшити навантаження на користувача та забезпечити логічне структурування інформації.

Інтерфейс також має адаптивну структуру, яка дозволяє коректно масштабуватись при зміні розміру вікна. Для цього використовуються гнучкі менеджери компонування (наприклад, QVBoxLayout, QGridLayout), що дозволяють елементам автоматично змінювати положення та розмір залежно від доступного простору.

У результаті реалізований метод побудови графічного інтерфейсу забезпечує не лише зручність використання, а й сприяє підвищенню безпеки та ефективності взаємодії з системним реєстром. Такий підхід дозволяє мінімізувати помилки користувача і забезпечити комфортну роботу з програмним забезпеченням.

2.4 Розробка моделі програми

Для кращого розуміння роботи застосунку було вирішено розробити модель функціонування системи на основі UML-діаграми діяльності (див. рисунок 2.1), що дозволяє наочно відобразити послідовність виконання дій, прийняття рішень та обробку даних у межах реалізованого функціоналу.

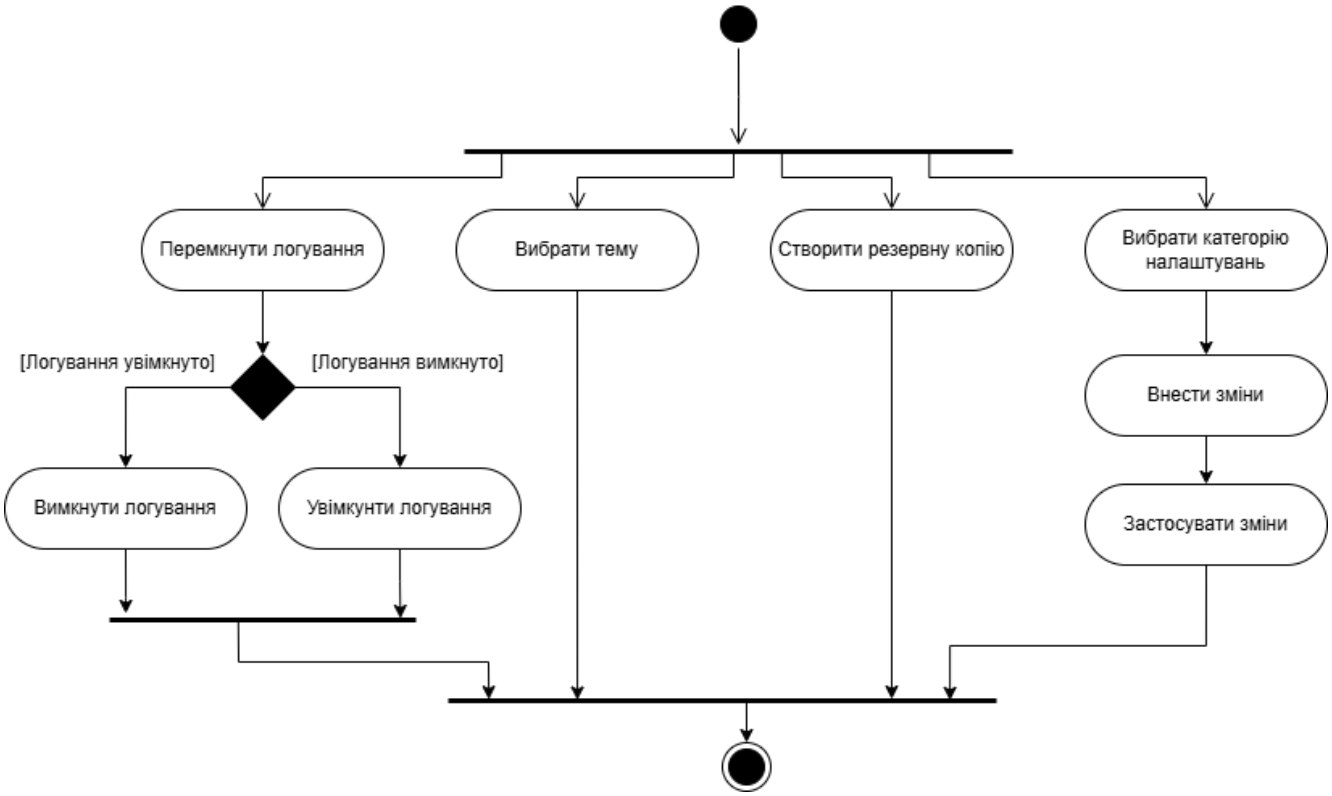

Рисунок 2.1 – Модель системи у вигляді діаграми діяльності застосунку

Згідно з розробленою діаграмою, робота користувача із застосунком починається із запуску програми та вибору потрібної категорії налаштувань. Користувач отримує доступ до переліку параметрів, які згруповані відповідно до обраної категорії.

На першому етапі користувач може переглянути поточний стан налаштувань та обрати необхідні зміни. Перед застосуванням змін програма пропонує створити резервну копію поточних параметрів реєстру для забезпечення можливості відновлення у разі необхідності.

Після підтвердження змін користувачем, застосунок вносить відповідні зміни до системного реєстру за допомогою внутрішнього модуля взаємодії з реєстром. Всі внесені зміни автоматично фіксуються у журналі логування для подальшого контролю та аналізу.

У випадку успішного внесення змін програма повідомляє користувача про завершення операції. Користувач також має можливість переглянути лог змін, відновити налаштування із резервної копії або змінити тему оформлення

інтерфейсу для більш комфортної роботи.

Розробка діаграми діяльності була виконана у веб-сервісі Draw.io [11].

2.5 Розробка алгоритму роботи програми

При розробці програмного застосунку для налаштування параметрів Windows через системний реєстр було сформовано загальний алгоритм роботи додатку (див. рисунки 2.2 – 2.3).

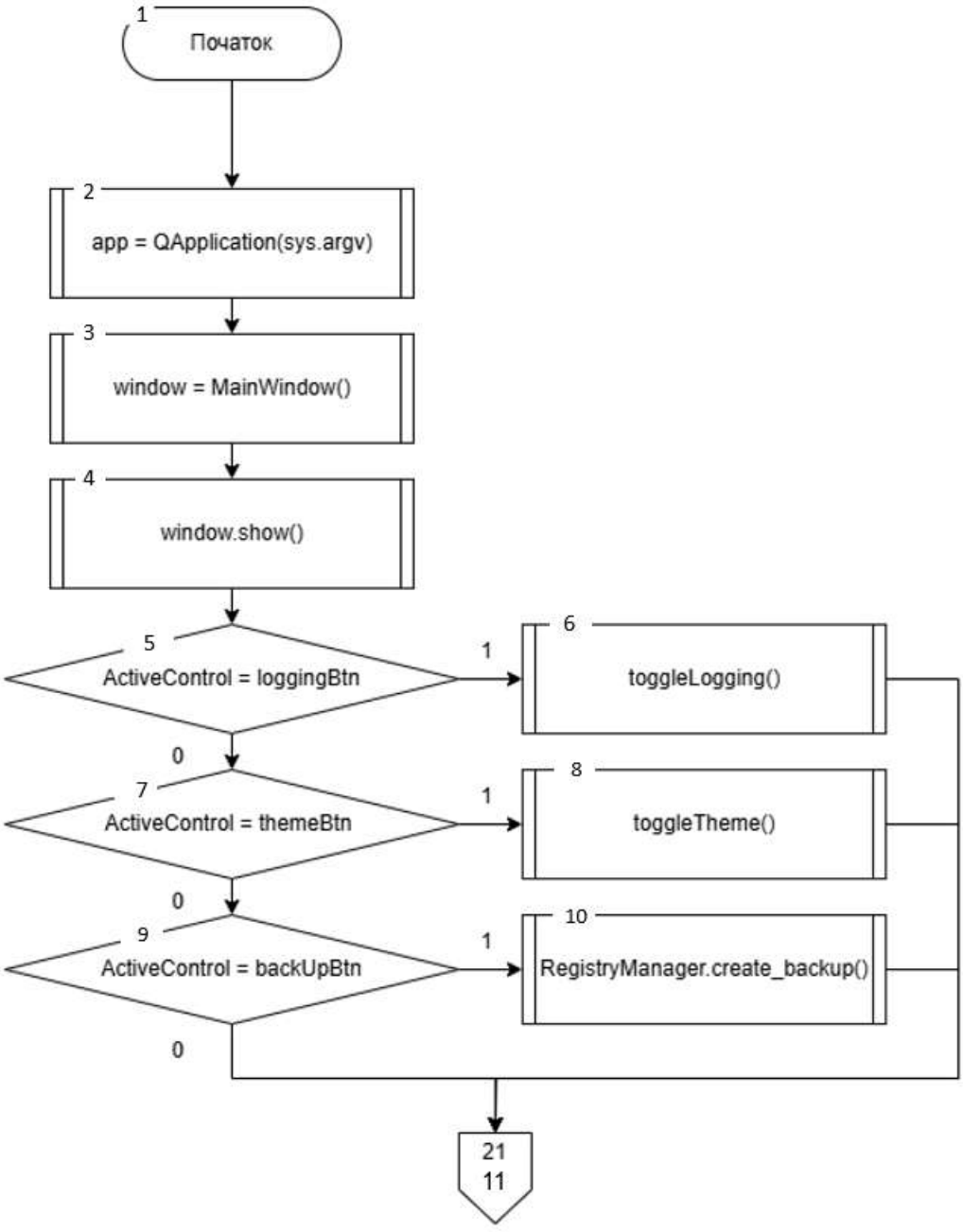

Рисунок 2.2 – Загальний алгоритм роботи програми

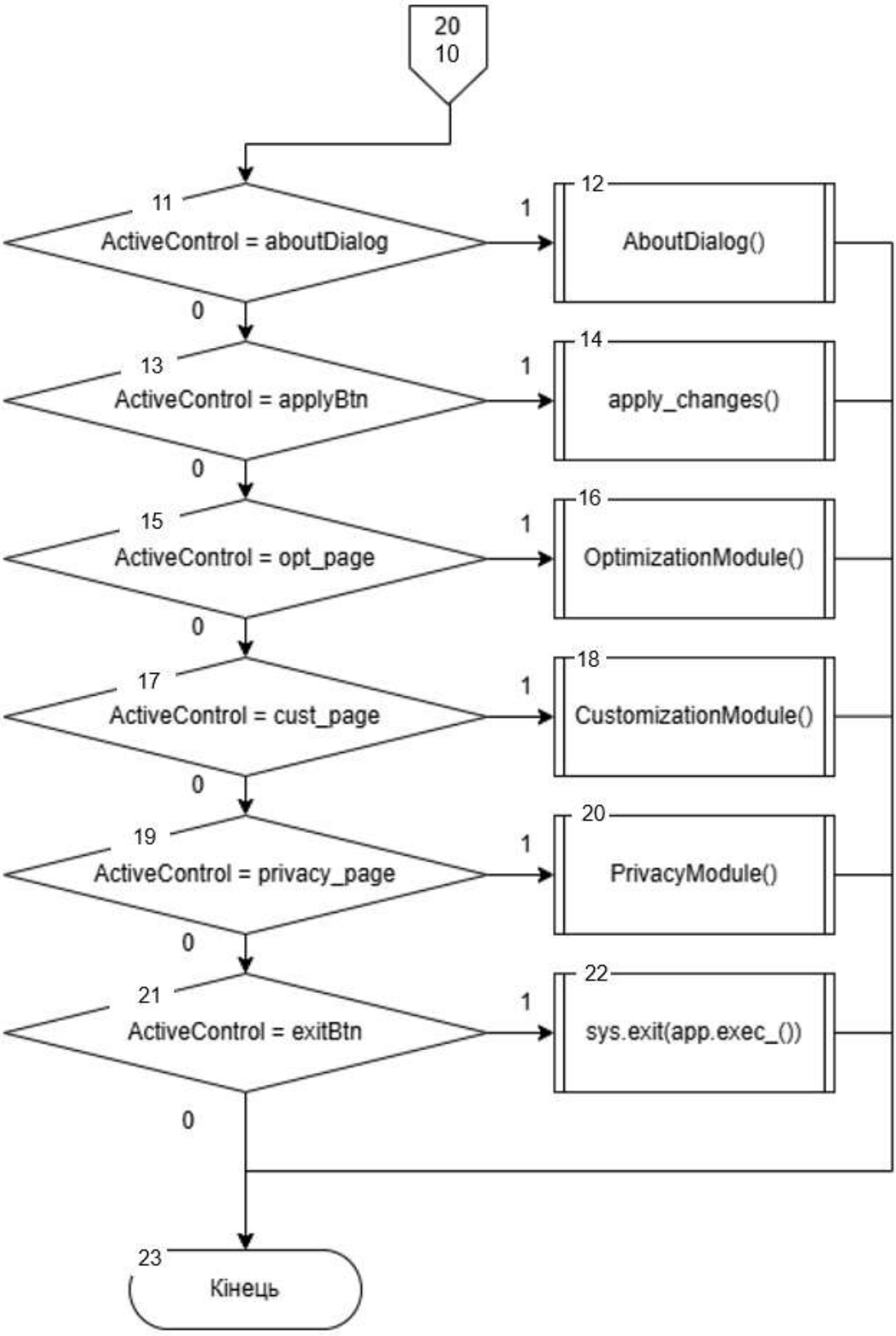

Рисунок 2.3 – Продовження загального алгоритму програми

Відповідно до розробленого алгоритму, першим етапом є завантаження головного вікна застосунку. Під час цього процесу виконується ініціалізація

ключових модулів системи, необхідних для стабільної роботи програми. До таких модулів належать, зокрема, компонент, відповідальний за взаємодію із системним реєстром, а також модуль фіксації та журналювання змін, що відбуваються протягом сеансу користувача. Ініціалізація згаданих елементів критично важлива для забезпечення надійного функціонування застосунку.

Головне вікно виступає центральною ланкою взаємодії між користувачем і програмним забезпеченням. Саме через нього забезпечується доступ до всього функціоналу, що реалізовано через зручний і зрозумілий графічний інтерфейс. Після завершення завантаження інтерфейсу та приведення усіх внутрішніх компонентів у робочий стан користувач отримує змогу розпочати роботу із застосунком.

На цьому етапі передбачено вибір однієї з кількох категорій налаштувань, що згруповані за функціональним принципом. Перша категорія – конфіденційність , що надає параметри, які дозволяють контролювати збір персональних даних та телеметрії. Друга – оптимізація, яка охоплює налаштування, що впливають на продуктивність, пришвидшують роботу системи та знижують ресурсне навантаження. Третя категорія – кастомізація, що дає змогу змінювати візуальні та поведінкові аспекти Windows, адаптуючи її під власні вподобання. Вибір відбувається через графічний інтерфейс, після чого користувачеві пропонується перелік параметрів для редагування.

Користувач, ознайомившись із доступними параметрами, обирає необхідні налаштування та переходить до підтвердження змін. Подальше впровадження обраних значень здійснюється за допомогою класу RegistryManager, який відповідає за безпосередню взаємодію з реєстром системи. Такий механізм дозволяє централізовано керувати конфігурацією, забезпечуючи узгодженість змін та зменшуючи ризик помилок під час їх застосування. Це сприяє стабільній роботі програмного забезпечення та спрощує подальше адміністрування.

Паралельно з внесенням змін усі дії, що впливають на реєстр або окремі його ключі, автоматично фіксуються у журналі подій. Цю функцію виконує модуль Logger, який послідовно зберігає інформацію про кожну зміну – зокрема,

дату, час, конкретний ключ та нове значення. Це дозволяє підтримувати повну історію взаємодії з системою для подальшого моніторингу або аналізу.

Ключовим елементом у функціонуванні програми є файл запуску, адже саме з нього починається виконання всього застосунку. У ньому визначено порядок ініціалізації основних компонентів, включно зі створенням графічного інтерфейсу, завантаженням головного вікна та активацією механізмів обробки подій. Коректна структура цього файлу гарантує стабільний запуск та взаємодію з користувачем.

На початку виконання відбувається імпорт необхідних елементів: класу QApplication із бібліотеки PyQt5, який відповідає за управління життєвим циклом застосунку, та класу MainWindow, що реалізує головне вікно програми. Це є стандартним кроком при створенні графічних інтерфейсів у середовищі PyQt.

Далі здійснюється перевірка, чи файл є головним модулем запуску. Це реалізується через конструкцію if __name__ == "__main__", яка гарантує виконання коду лише у разі прямого запуску, а не при імпорті з іншого модуля.

Після цього створюється екземпляр застосунку (app = QApplication(sys.argv)), що обробляє системні аргументи та готує середовище для запуску інтерфейсу. Потім створюється об'єкт головного вікна (window = MainWindow()), який містить основну логіку взаємодії користувача із застосунком. За допомогою методу show() головне вікно виводиться на екран, забезпечуючи доступ до функціоналу.

Завершальний етап – запуск основного циклу обробки подій через sys.exit(app.exec_()). Цей цикл відповідає за реагування програми на дії користувача в режимі реального часу, зокрема натискання кнопок, переміщення вікон або інші взаємодії з інтерфейсом. Саме це забезпечує повноцінну роботу графічного застосунку в середовищі операційної системи Windows.

У результаті виконання всіх зазначених етапів формується повноцінне середовище, готове до взаємодії з користувачем. Чітка послідовність ініціалізації компонентів, створення графічного інтерфейсу та запуску обробника подій дозволяє забезпечити коректну роботу застосунку з моменту запуску й протягом усього сеансу. Такий підхід гарантує стабільність, передбачуваність поведінки

програми та зручність подальшої підтримки й розширення функціоналу.

2.6 Висновки

У другому розділі було проведено детальний аналіз інформаційного забезпечення програми, призначеної для налаштування параметрів Windows через системний реєстр. У межах цього аналізу було розглянуто ключові компоненти, необхідні для стабільної та ефективної роботи застосунку. Зокрема, було розроблено методи взаємодії з системним реєстром, а також побудови графічного інтерфейсу користувача.

Також було створено загальну модель роботи застосунку, що ілюструє основні етапи його функціонування. Ця модель представлена у вигляді UML-діаграми діяльності, яка чітко демонструє послідовність дій з боку користувача та відповідні реакції системи. Такий підхід дозволив структурувати логіку роботи програми та візуалізувати її ключові процеси.

Крім того, було сформовано основний алгоритм роботи застосунку, що відображає логіку взаємодії користувача з інтерфейсом, починаючи від запуску програми і завершуючи застосуванням обраних налаштувань. Це дало змогу краще зрозуміти внутрішню організацію застосунку та забезпечити його узгоджену й послідовну роботу.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програми

При розробці програмного забезпечення для налаштування параметрів Windows через системний реєстр важливо обрати найбільш оптимальні технології для реалізації всіх компонентів. Основними критеріями вибору стали: простота реалізації, надійність, доступність бібліотек для роботи з реєстром та створення графічного інтерфейсу, а також можливість швидкої розробки й масштабованості застосунку.

Мову програмування для розробки програмного продукту було обрано серед декількох популярних варіантів, що забезпечують необхідну функціональність для роботи з системним реєстром, створення графічного інтерфейсу користувача та реалізації резервного копіювання налаштувань. Розглядалися такі мови програмування:

Python – сучасна мова програмування, яка відзначається високою швидкістю розробки, простотою синтаксису та наявністю великої кількості бібліотек для роботи з Windows API, у тому числі з системним реєстром. Підтримує створення графічних інтерфейсів за допомогою бібліотек таких як PyQt5, Tkinter тощо, також забезпечує зручну інтеграцію з операційною системою Windows, що дозволяє ефективно працювати з її API та різними системними компонентами, такими як реєстр, файлову систему та мережеві ресурси. Це робить Python ідеальним вибором для створення програмного забезпечення, яке потребує тісної взаємодії з Windows. Завдяки наявності численних бібліотек, таких як pywin32, розробка програм для Windows стає швидшою та зручнішою, а також дозволяє знизити складність завдяки готовим рішенням для інтеграції з основними системними функціями [12].

C++ – потужна компільована мова програмування, що надає прямий доступ до системних ресурсів та високий рівень контролю за пам'яттю. Проте, розробка застосунків на C++ вимагає значно більше часу через складніший синтаксис та

управління ресурсами. Створення графічного інтерфейсу також є більш трудомістким порівняно з іншими мовами [13].

Java – популярна мова програмування, яка надає кросплатформеність і безпеку. Проте робота з системним реєстром Windows у Java є обмеженою і потребує використання додаткових сторонніх бібліотек або нативних викликів через JNI (Java Native Interface), що ускладнює розробку [14].

C# – мова програмування, тісно інтегрована з платформою Windows та середовищем .NET [15]. Забезпечує зручний доступ до системного реєстру та дозволяє швидко створювати графічні інтерфейси за допомогою Windows Forms або WPF.

Для обґрунтованого вибору було сформовано порівняльну характеристику мов програмування (див. таблицю 3.1).

Таблиця 3.1 – Порівняльна характеристика мов програмування

Критерій	Python	C++	Java	C#
Простота синтаксису	+	–	_	+
Наявність вбудованих бібліотек для GUI	+	–	+	+
Наявність вбудованих бібліотек для роботи з реєстром	+	+	–	+
Швидкість розробки	+	–	+	-
Складність розгортання застосунку	+	–	–	+
Загальний бал	5	1	2	4

На основі проведеного аналізу видно, що Python отримав найвищий бал серед розглянутих мов. Він забезпечує просту та швидку розробку, має вбудовані засоби для роботи із системним реєстром і побудови інтерфейсу.

Для розробки програмного продукту було розглянуто п’ять популярних середовищ розробки програмного забезпечення: Visual Studio Code, PyCharm, Sublime Text, Atom та Eclipse. Основними критеріями вибору були: зручність ро-

боти з проєктами на Python, наявність підтримки розширень, швидкість роботи, простота налаштування та інтеграція з системою контролю версій.

Visual Studio Code (VSCode) – безкоштовне середовище розробки від компанії Microsoft, яке є одним із найпопулярніших середовищ для розробки програмного забезпечення на різних мовах, включаючи Python. VSCode має велику кількість розширень, зокрема офіційне розширення для Python, підтримує налагодження коду, автодоповнення, роботу з Git і терміналом безпосередньо з вікна редактора. Висока швидкість роботи, можливість тонкої настройки середовища, відкритий вихідний код та велика кількість безкоштовних плагінів роблять VSCode дуже привабливим для розробників різного рівня [16].

PyCharm – середовище розробки від компанії JetBrains, спеціалізоване на розробці застосунків мовою Python. Має зручний інтерфейс, потужну підтримку налагодження, рефакторингу, тестування коду та інтеграції з базами даних. PyCharm пропонує безліч професійних інструментів для розробників, включно з інтеграцією Docker, системами контролю версій та можливістю роботи з вебтехнологіями [17].

Однак повноцінна версія PyCharm є платною (Professional Edition), а безкоштовна версія (Community Edition) має обмежений функціонал, що може створити труднощі при роботі над великими або складними проєктами.

Sublime Text – легковажний та дуже швидкий текстовий редактор із підтримкою плагінів для розробки програмного забезпечення на багатьох мовах, включно з Python [18]. Sublime Text має мінімалістичний інтерфейс і високий рівень продуктивності, що дозволяє працювати навіть на слабких комп’ютерах.

Однак сам по собі редактор має обмежену базову функціональність, і для повноцінної роботи з Python потрібно встановлювати додаткові плагіни.

Atom – текстовий редактор з відкритим вихідним кодом, розроблений компанією GitHub. Atom орієнтований на гнучкість і можливість глибокої персоналізації за допомогою плагінів. Він підтримує роботу з багатьма мовами програмування, включаючи Python, має вбудовану інтеграцію з Git та GitHub.

Проте Atom поступається за швидкістю роботи іншим середовищам, особливо на великих проєктах, що може знизити продуктивність розробника. Також проєкт Atom зараз знаходиться у стані припинення активної розробки [19].

Eclipse – класичне середовище розробки, що орієнтоване на мову Java, але за допомогою розширення PyDev підтримує і розробку на Python. Eclipse має розвинену інфраструктуру плагінів, потужні засоби для налагодження та роботи з великими проєктами [20].

Разом з тим, Eclipse є досить важким середовищем: тривала інсталяція, високе споживання ресурсів і складність налаштування можуть ускладнити роботу.

Для обґрунтування вибору середовища розробки було сформовано порівняльну характеристику (див. таблицю 3.2).

Таблиця 3.2 – Порівняльна характеристика середовищ розробки

Критерій	VSCode	PyCharm	Sublime Text	Atom	Eclipse
Швидкість роботи	+	–	+	–	–
Простота налаштування	+	+	+	+	–
Наявність безкоштовної повної версії	+	–	–	+	+
Інтеграція з Git	+	+	–	+	+
Підтримка розширень і плагінів	+	+	+	+	+
Підтримка тестування (Unit Tests, PyTest)	+	+	–	–	+
Гнучкість у налаштуванні комбінацій клавіш	+	+	–	+	+
Загальний бал	7	5	3	6	5

На основі аналізу видно, що Visual Studio Code набрав найбільшу кількість балів за всіма критеріями. Він поєднує у собі простоту, гнучкість, швидкість роботи та повну безкоштовність без обмеження функціоналу.

Для створення графічного інтерфейсу користувача у процесі розробки застосунку було розглянуто дві популярні бібліотеки для Python: Tkinter та PyQt5. Обидві бібліотеки надають необхідний базовий функціонал для розробки віконних застосунків, проте мають свої особливості, переваги та обмеження.

Tkinter є стандартною бібліотекою для створення графічних інтерфейсів у Python. Вона поставляється разом із мовою Python і не потребує окремого встановлення. Tkinter дозволяє створювати базові графічні елементи, такі як кнопки, мітки, текстові поля, списки тощо. Бібліотека має простий та зрозумілий API, що дозволяє швидко створювати невеликі віконні застосунки. Інтерфейси, створені за допомогою Tkinter, є простими та легкими, проте вигляд додатків може бути дещо застарілим порівняно з сучасними вимогами до дизайну. Крім цього, можливості стилізації елементів у Tkinter є обмеженими: зміна вигляду віджетів зазвичай обмежується базовими властивостями кольору, шрифту та розміру, що ускладнює створення професійних інтерфейсів без сторонніх бібліотек [21].

PyQt5 – це одна з найбільш потужних бібліотек для створення графічних інтерфейсів у Python, яка базується на фреймворку Qt [22]. PyQt5 забезпечує багатий набір віджетів та розширені можливості для створення складних та сучасних інтерфейсів. Важливою перевагою є підтримка використання стилів через QSS (Qt Style Sheets), що за своєю структурою та логікою нагадують CSS, що значно спрощує налаштування зовнішнього вигляду елементів. PyQt5 також підтримує роботу з анімаціями, вкладками, діалоговими вікнами та іншими сучасними елементами інтерфейсу.

У результаті порівняння було обрано бібліотеку PyQt5, оскільки вона забезпечує більш сучасний вигляд застосунків, розширені можливості стилізації за допомогою QSS, а також дозволяє створювати гнучкі та адаптивні інтерфейси відповідно до сучасних вимог до користувацького досвіду.

Для реалізації функціоналу взаємодії із системним реєстром Windows у програмному застосунку було розглянуто декілька варіантів бібліотек та модулів Python. Серед основних кандидатів були: winreg, pywin32 та ctypes.

Winreg – це стандартний модуль мови програмування Python, який надає функціональні можливості для безпосередньої роботи з реєстром Windows. Модуль дозволяє виконувати основні операції, такі як відкриття ключів реєстру, читання значень параметрів, створення та видалення ключів і значень, а також зміну прав доступу. Оскільки winreg входить до складу стандартної бібліотеки Python, його використання не потребує встановлення додаткових пакетів або бібліотек, що значно спрощує процес розробки та розгортання застосунку. Крім того, winreg має чіткий і зрозумілий інтерфейс функцій, що полегшує реалізацію базових операцій з реєстром [23].

Pywin32 – це потужна бібліотека для доступу до функціоналу Windows API з мови Python, включаючи роботу з реєстром. Вона надає розширені можливості для більш глибокої взаємодії із системою, однак є досить об'ємною та вимагає додаткового встановлення. До того ж, використання pywin32 ускладнює деплоймент програмного продукту, оскільки потрібно враховувати додаткові залежності і середовище виконання [24].

CtTypes – стандартна бібліотека Python для роботи з нативними викликами системних функцій через динамічні бібліотеки. Використовуючи ctypes, можна безпосередньо звертатися до функцій Windows API для роботи з реєстром. Проте це вимагає ручної роботи з адресами пам’яті, структурами та викликами функцій на низькому рівні, що підвищує складність розробки та ризик появи помилок [25].

Аналіз можливих рішень показав, що для задач даного проєкту найбільш доцільним є використання саме winreg. Цей модуль забезпечує необхідний набір функцій для ефективної роботи з системним реєстром без зайвих ускладнень і додаткових залежностей. Стандартний характер бібліотеки гарантує її доступність у будь-якому середовищі Python для Windows, що значно спрощує розгортання та підтримку програмного продукту.

Отже, для реалізації взаємодії із системним реєстром Windows у кваліфікаційній роботі було обрано бібліотеку winreg.

Кожен із вибраних інструментів має свої сильні сторони:

- Python забезпечує швидку розробку та доступ до API Windows.

- PyQt5 надає можливості створення привабливого графічного інтерфейсу.

- winreg гарантує надійний доступ до системного реєстру без необхідності встановлення сторонніх бібліотек.

- VS Code оптимізує процес розробки завдяки зручному інтерфейсу, автоформатуванню та розширенням для роботи з Python.

Поєднання цих технологій дозволяє ефективно реалізувати застосунок, який об'єднує функціональність змін налаштувань Windows, їх резервного копіювання, логування змін і гнучкого управління через сучасний інтерфейс.

Крім того, обраний стек технологій забезпечує високу масштабованість проєкту: у разі необхідності функціонал можна розширити без суттєвих змін архітектури програми.

Отже, вибір стеку технологій на основі Python, PyQt5, winreg і Visual Studio Code є оптимальним рішенням для розробки програмного забезпечення, що поєднує в собі простоту та надійність.

3.2 Розробка модуля для взаємодії з системним реєстром

У рамках розробки застосунку для налаштування параметрів операційної системи Windows через системний реєстр було реалізовано окремий модуль – клас RegistryManager (див. рисунок 3.1). Цей клас інкапсулює функціонал взаємодії із системним реєстром, дозволяючи безпечно читати, змінювати та резервувати параметри.

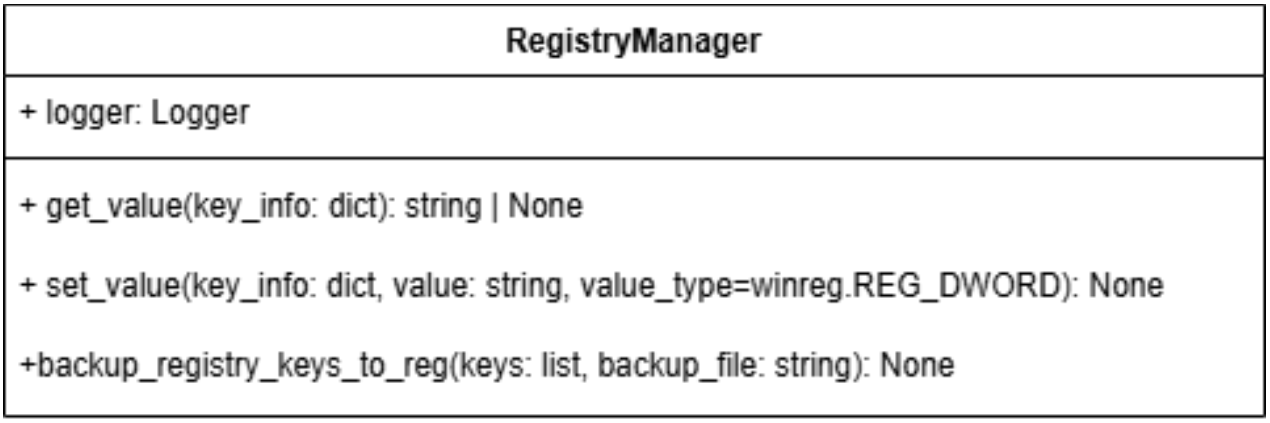

Рисунок 3.1 – Клас RegistryManager

В класі реалізовано такі основні методи:

1. get_value – метод для зчитування значення певного параметра реєстру. Він приймає на вхід словник даних, що містить інформацію про розділ реєстру, шлях та ім'я параметра. Метод намагається відкрити відповідний ключ у режимі читання та отримати значення параметра. У разі відсутності ключа або помилки в процесі зчитування, система генерує запис у журналі попереджень і повертає None.

На рисунку 3.2 продемонстровано алгоритм роботи метода.

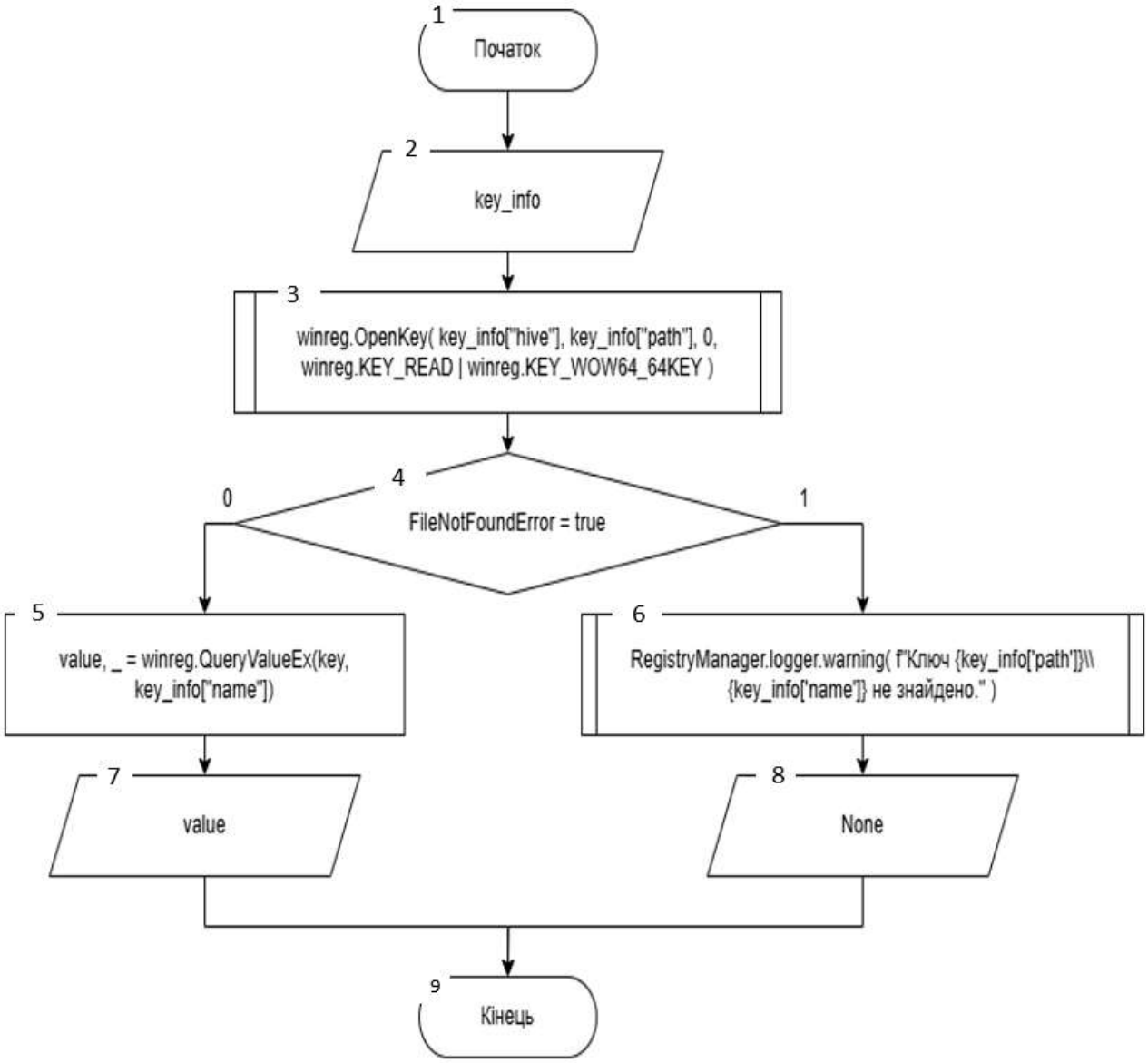

Рисунок 3.2 – Алгоритм роботи метода get_value

2. set_value – метод для створення або оновлення параметра в реєстрі. Він приймає словник із даними про ключ реєстру, нове значення та тип параметра. Спочатку відбувається спроба відкрити ключ для зміни. Якщо такий ключ не існує, створюється новий. Після чого встановлюється нове значення. Всі дії фіксуються у журналі: успішне оновлення параметра або опис помилки у випадку її виникнення.

На рисунку 3.3 продемонстровано алгоритм роботи метода.

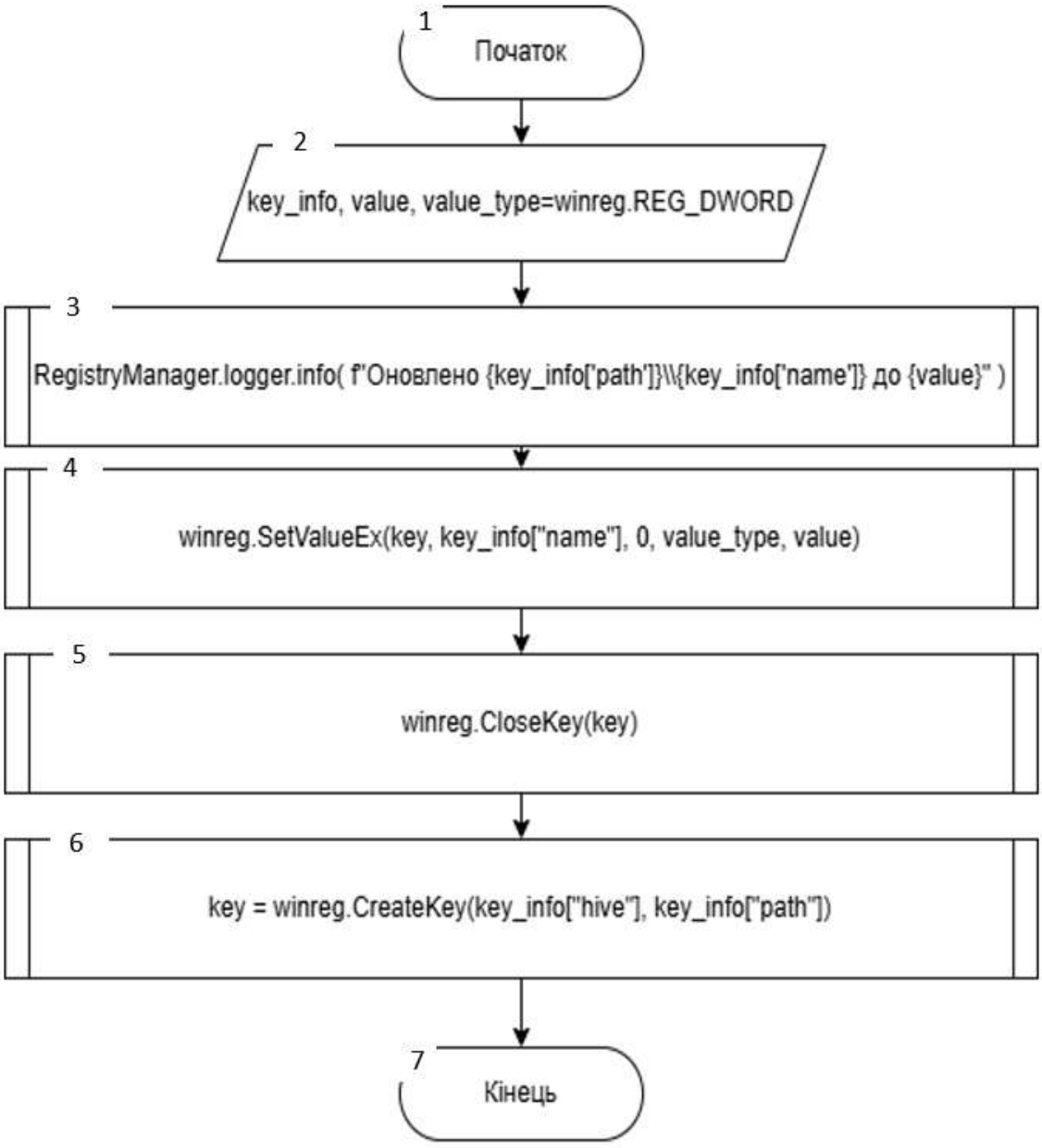

Рисунок 3.3 – Алгоритм роботи метода set_value

3. backup_registry_keys_to_reg – метод для створення резервної копії групи реєстрових ключів у файл формату .reg (див. рисунок 3.4).

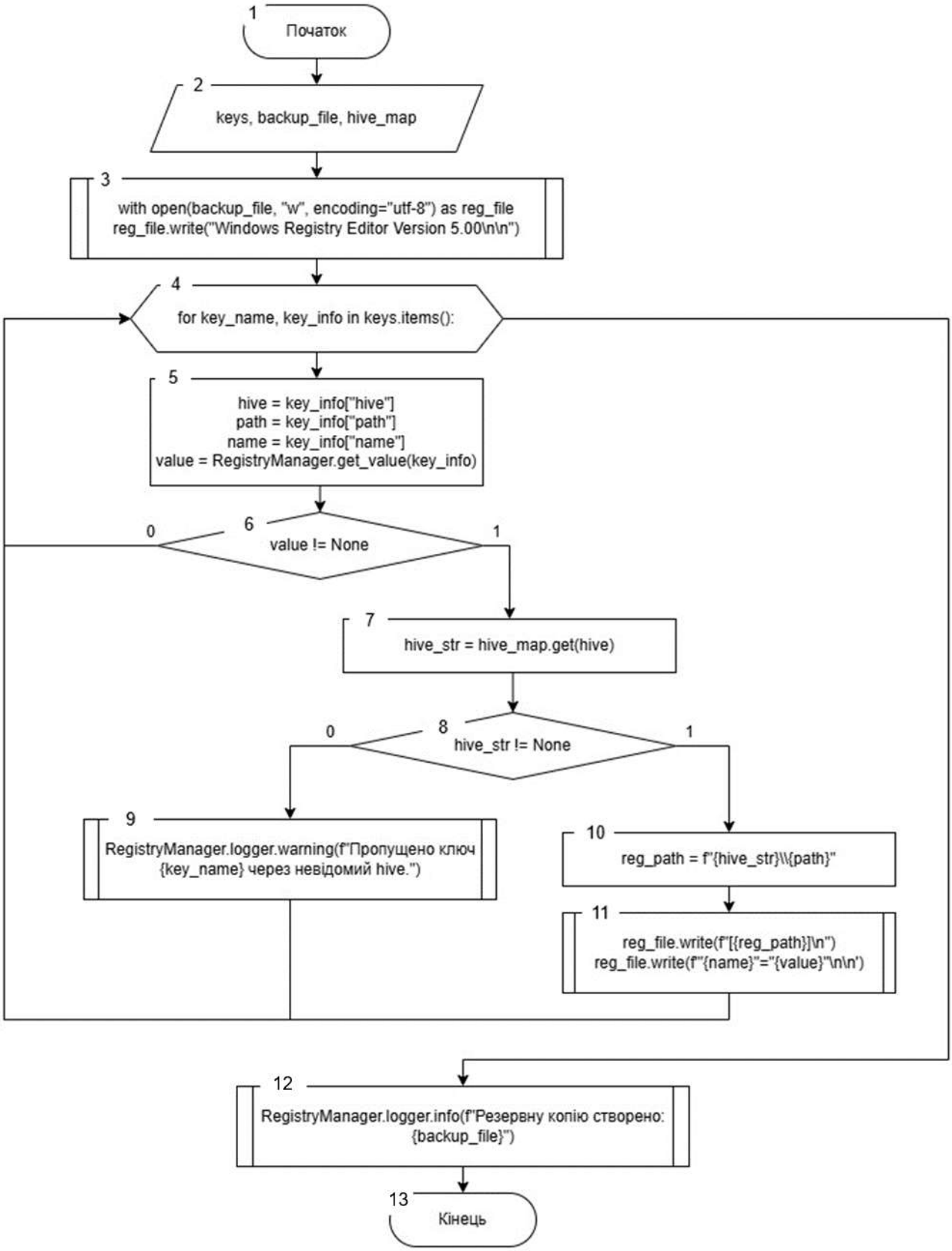

Рисунок 3.4 – Алгоритм роботи метода backup_registry_keys_to_reg

Метод приймає набір ключів і шлях до файлу збереження. Він формує файл резервної копії у форматі, сумісному з Windows Registry Editor (.reg), з усіма зчитаними значеннями параметрів. Успішне створення резервної копії або помилки фіксуються в журналі застосунку.

Крім того, у застосунку визначена константа REGISTRY_KEYS, яка містить повний перелік реєстрових ключів, що використовуються для налаштування параметрів системи.

Константа реалізована у вигляді словника, де для кожного налаштування вказано:

– гілку реєстру (hive), наприклад HKEY_CURRENT_USER або HKEY_LOCAL_MACHINE;

– шлях до розділу (path);

– ім'я параметра (name);

– тип налаштування (type), що вказує на приналежність до певної категорії (кастомізація, оптимізація або конфіденційність).

Такий підхід дозволяє централізовано управляти переліком налаштувань, які можна змінювати через інтерфейс застосунку, а також спрощує створення резервних копій та виконання масових операцій із реєстром.

Таким чином, реалізований модуль дозволяє безпечно та ефективно змінювати параметри системного реєстру Windows без необхідності прямого використання стандартних засобів редагування

3.3 Розробка модуля для фіксації внесених змін до реєстру

У сучасних програмних розробках важливим аспектом є ефективний контроль за виконанням операцій та моніторинг змін у системі. Це дозволяє не лише забезпечити прозорість процесів, а й швидко реагувати на непередбачувані ситуації або помилки. Зокрема, у програмах, які взаємодіють із критичними компонентами операційної системи, такими як системний реєстр Windows, виникає потреба в надійному механізмі фіксації всіх змін і подій, що відбуваються в процесі роботи. Це особливо актуально для програм, які надають користувачеві

розширені можливості налаштування системи, адже будь-яке неконтрольоване втручання може призвести до збоїв у роботі ОС або втрати даних.

Для розв’язання цих завдань було розроблено модуль логування, що забезпечує реєстрацію всіх ключових подій, які відбуваються під час взаємодії з системним реєстром Windows. Цей модуль став невід’ємною складовою архітектури застосунку, оскільки надає змогу фіксувати як стандартні події (наприклад, зміна параметрів), так і виняткові ситуації (зокрема, помилки доступу чи некоректні значення). Модуль дозволяє зберігати важливу інформацію щодо змін у реєстрі, попереджень і помилок, що виникають під час роботи програми, тим самим підвищуючи надійність та безпеку системи. Наявність журналу змін також дозволяє користувачеві в подальшому аналізувати історію дій та за потреби скасувати або повторити певні налаштування.

Для забезпечення контролю за змінами, які вносяться до системного реєстру, у межах роботи було реалізовано окремий модуль – клас Logger. Цей клас відповідає за централізоване ведення журналу подій і є прикладом використання принципів модульного програмування. Завдяки чіткій структурі реалізації, логування можна легко масштабувати або адаптувати під інші проєкти. На рисунку 3.5 продемонстровано діаграму класу, яка ілюструє основні компоненти модуля та їх взаємозв’язки.

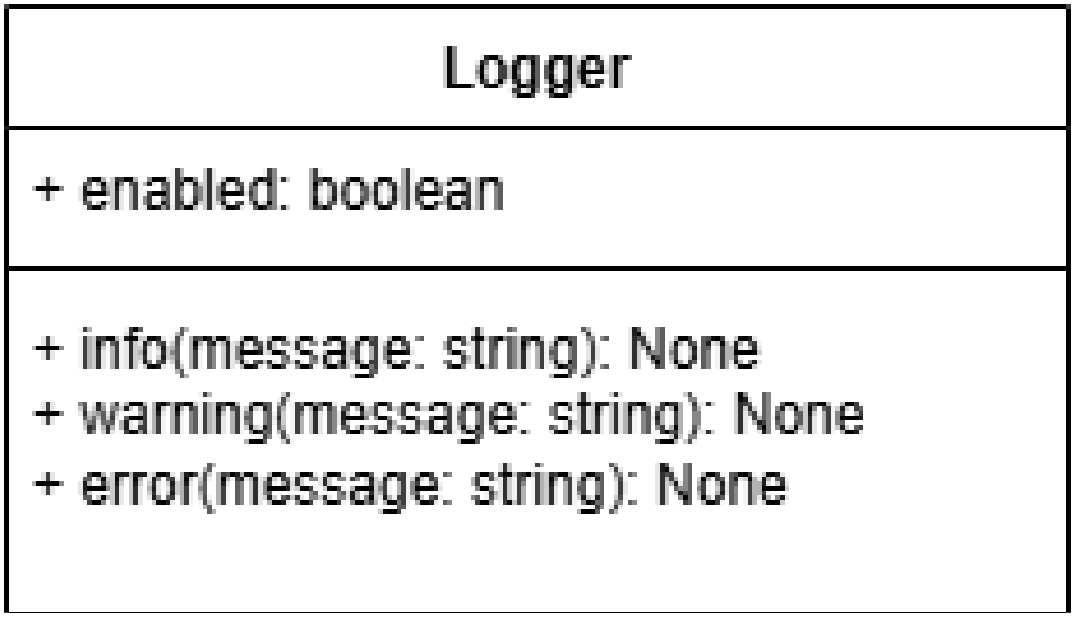

Рисунок 3.5 – Діаграма класу

В класі реалізовано такі основні можливості:

1. Автоматичне створення каталогу для збереження лог-файлів. При ініціалізації об’єкта перевіряється наявність директорії для журналів. Якщо така відсутня, вона створюється автоматично.

2. Створення окремого лог-файлу для кожного дня роботи застосунку. Назва файлу формується динамічно відповідно до поточної дати. Це дозволяє впорядковано зберігати всі події у розрізі часу.

3. Реалізовано запис інформаційних повідомлень про виконані операції. Метод для запису повідомлень інформує про успішне внесення змін, резервне копіювання або інші ключові події під час роботи програми (див. рисунок 3.6).

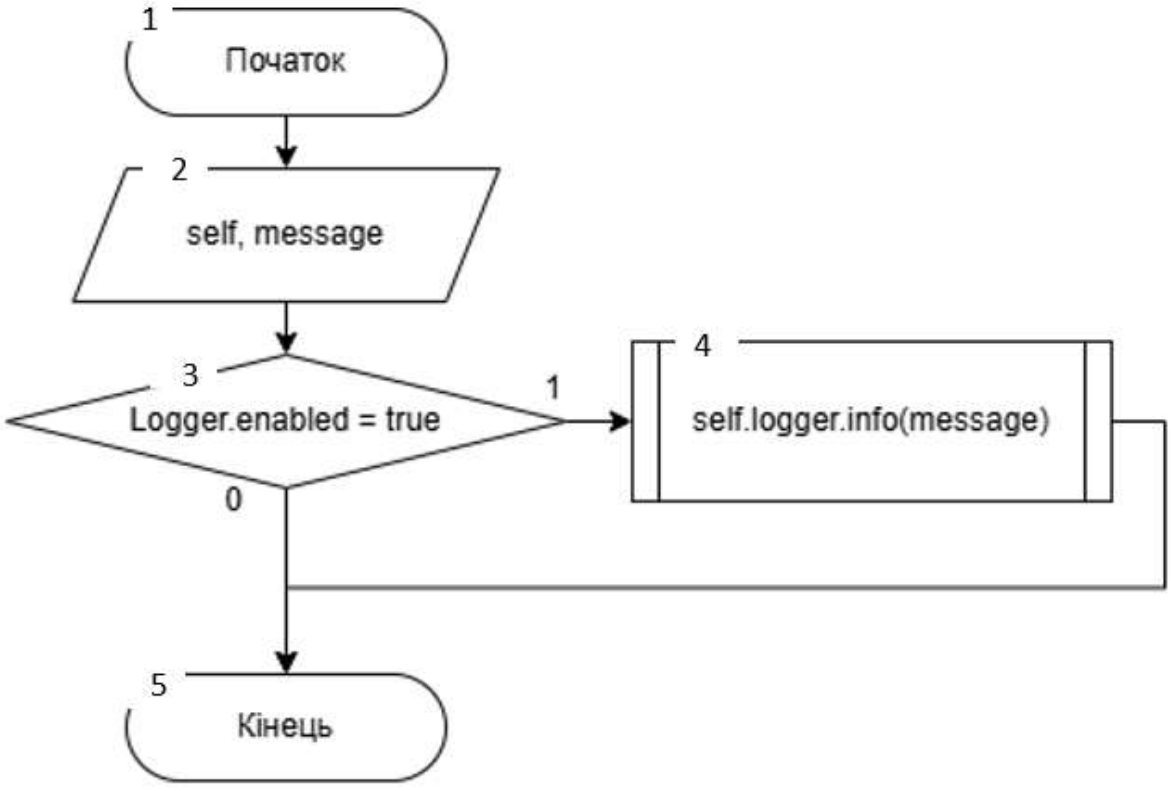

Рисунок 3.6 – Алгоритм роботи методу info

4. Реалізовано механізм фіксації попереджень. Метод дозволяє зафіксувати потенційно небезпечні ситуації, наприклад, відсутність потрібного ключа реєстру або некоректне значення (див. рисунок 3.7).

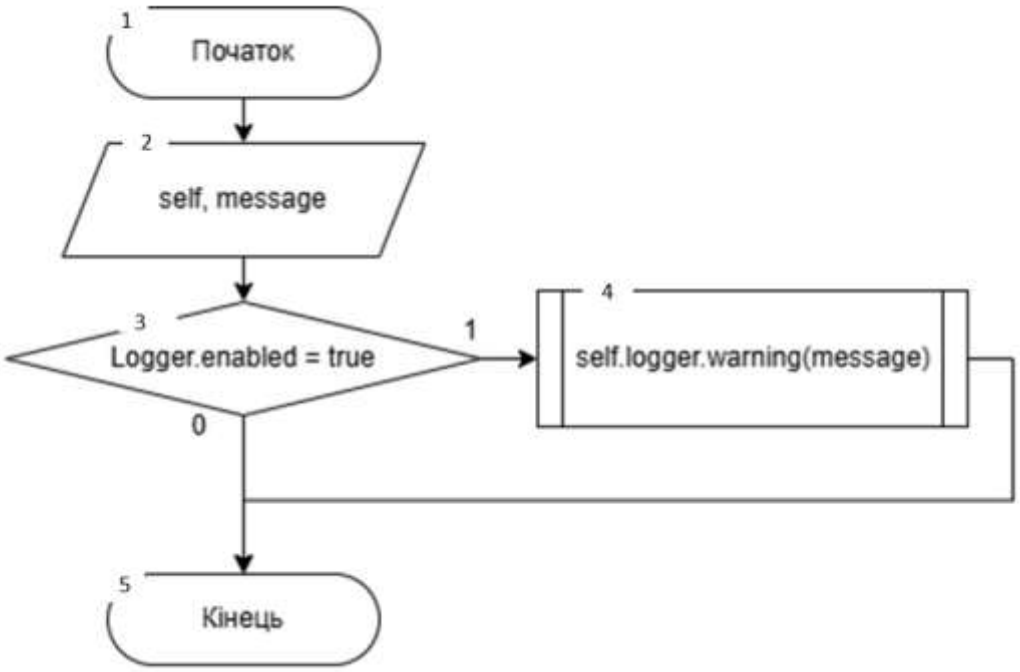

Рисунок 3.7 – Алгоритм роботи методу warning

5. Реалізовано обробку та запис помилок. Метод дозволяє детально фіксувати всі помилки, які виникають у процесі роботи програми (див. рисунок 3.8).

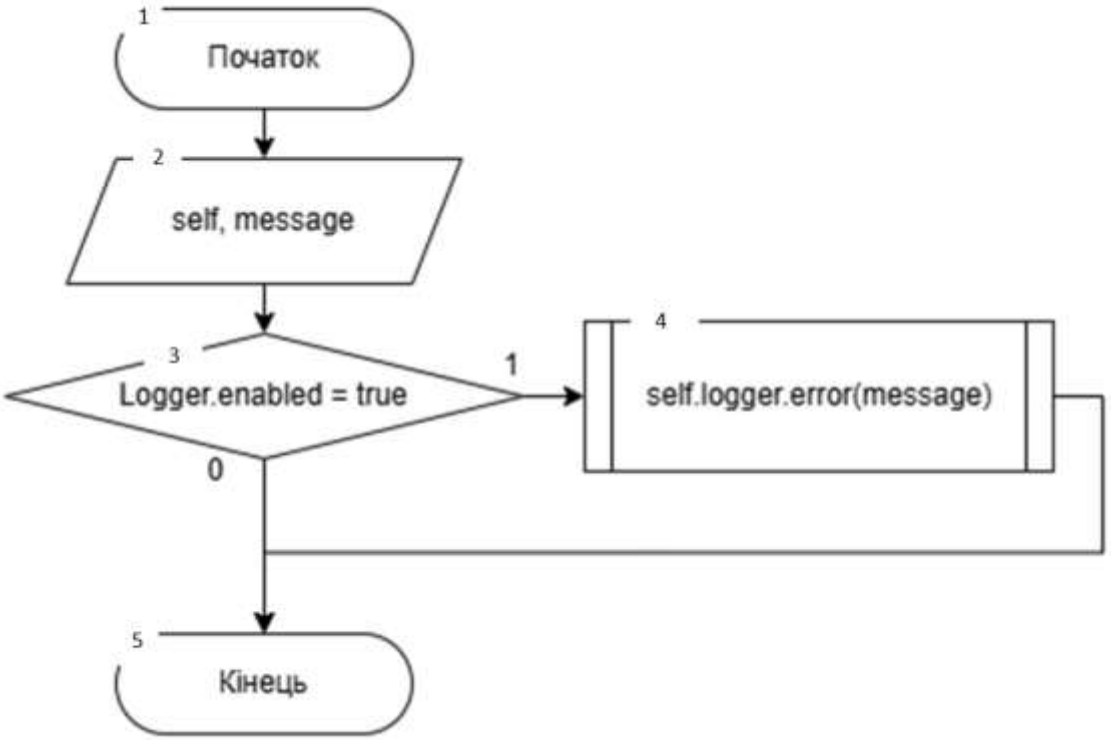

Рисунок 3.8 – Алгоритм роботи методу error

Окрім базового запису подій, модуль логування був спроєктований із урахуванням вимог до масштабованості та зручності аналізу даних. У процесі реалізації враховувалися потреби як пересічного користувача, так і розробника, який може надалі здійснювати технічну підтримку або удосконалення застосунку. Використання окремих методів для інформаційних повідомлень, попереджень та помилок дозволяє класифікувати всі події за рівнем важливості, що значно полегшує подальший аудит змін. Це розмежування дає змогу швидко виявити критичні помилки, не загубившись у загальному потоці подій.

Структурований підхід до створення лог-файлів спрощує процес пошуку потрібної інформації під час діагностики роботи застосунку. Зокрема, передбачено логування з поділом за датами, що дозволяє легко відслідковувати хронологію змін. Такий підхід сприяє швидкому виявленню причин виникнення несправностей і дає змогу оперативно вжити заходів для їх усунення. Завдяки впорядкуванню логів за датами та їх автоматичному створенню користувач або розробник може швидко відстежити історію змін системного реєстру.

Таким чином, розроблений модуль логування є критично важливою частиною системи, яка забезпечує прозорість усіх змін у реєстрі, підвищує безпеку використання програми та полегшує процес виявлення й усунення помилок. Його реалізація дозволяє досягти високого рівня стабільності та надійності всього програмного комплексу, що відповідає сучасним вимогам до системного програмного забезпечення.

3.4 Висновки

У третьому розділі обґрунтовано вибір технологій та інструментів, що використовувалися для розробки програмних модулів застосунку для налаштування параметрів Windows через системний реєстр.

На основі аналізу вимог до функціональності та зручності роботи було обрано мову програмування Python та бібліотеку PyQt5 для створення графічного інтерфейсу користувача. Для безпосередньої роботи із системним реєстром було використано стандартний модуль winreg, який забезпечує стабільну інтеграцію із

Windows API.

У межах розділу також було представлено опис розробки основних програмних модулів системи. Зокрема, одним із ключових компонентів є модуль для взаємодії з реєстром, який виконує функції безпечного читання, редагування, а також резервного збереження параметрів, що змінюються. Така реалізація дозволяє уникнути пошкодження системних налаштувань та забезпечує можливість відновлення до попереднього стану у випадку виникнення помилок або збоїв.

Окрім того, був реалізований модуль логування змін, який відповідає за фіксацію всіх дій, що здійснюються у реєстрі. Цей модуль відіграє важливу роль у забезпеченні прозорості та контролю за змінами, що вносяться в систему. Він дозволяє зберігати інформацію про кожну операцію, яка була виконана, включно з часом її здійснення та конкретними параметрами, що були змінені. Такий підхід полегшує подальший аналіз і діагностику, а також сприяє підвищенню рівня безпеки застосунку.

Таким чином, вибір відповідних технологій та інструментальних засобів, а також продумана розробка окремих функціональних модулів забезпечили надійну основу для створення ефективного, безпечного та зручного у використанні програмного продукту, здатного виконувати поставлені завдання з налаштування параметрів Windows на високому рівні якості.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз методів тестування

Тестування програмного забезпечення є невід'ємною складовою процесу його розробки, оскільки дозволяє виявити та усунути помилки ще до етапу впровадження, а також гарантувати стабільність, надійність і відповідність програмного продукту вимогам користувача. Якість тестування безпосередньо впливає на рівень довіри до програмного забезпечення, особливо у випадку, коли воно взаємодіє із критичними компонентами операційної системи, як-от системний реєстр.

Методи тестування, що застосовуються, повинні відповідати особливостям функціоналу програмного забезпечення, специфіці даних, з якими воно працює, а також способу взаємодії з кінцевим користувачем. У випадку розробки програмного забезпечення, що виконує конфігурацію операційної системи Windows, ключовим є забезпечення стабільної і безпечної взаємодії із системним реєстром. Це вимагає застосування спеціалізованих підходів до перевірки всіх операцій, пов’язаних із читанням, записом, модифікацією та збереженням параметрів реєстру.

Основним видом тестування у цьому контексті є функціональне тестування, яке забезпечує перевірку правильності виконання ключових функцій програми. Сюди входить перевірка коректного зчитування параметрів із системного реєстру, їх редагування, видалення, а також можливість додавання нових записів. Особлива увага приділяється перевірці працездатності механізмів створення резервних копій і їх подальшого відновлення, що гарантує захист від можливих помилок та втрати важливих даних.

Окрім функціонального, важливим є тестування на відмову. Такий вид тестування дозволяє оцінити, як програмне забезпечення поводиться у нестандартних або критичних ситуаціях, наприклад, при відсутності доступу до певних директорій, відсутності прав адміністратора чи виникненні внутрішніх системних збоїв. Це дає змогу перевірити, наскільки ефективно реалізовано

обробку винятків, і чи здатна програма уникнути аварійного завершення роботи.

Ще одним аспектом є тестування швидкодії. Продуктивність програмного забезпечення стає особливо важливою при виконанні ресурсоємних задач, таких створення резервних копій великого обсягу. Затримки у виконанні таких операцій можуть призводити до незадоволення користувачів, тому їх необхідно своєчасно виявляти та оптимізовувати. Для зручності користувача важливо, щоб усі операції виконувалися швидко та без збоїв навіть на комп'ютерах середнього рівня продуктивності.

У разі, якщо програмне забезпечення має графічний інтерфейс користувача (GUI), обов'язковим є тестування його елементів. Перевіряється правильність відображення графічних елементів, логіка переходу між вкладками, відгук на дії користувача, а також адаптація інтерфейсу до різних розмірів екрана чи змін теми оформлення. Недоліки в роботі GUI можуть значно знизити зручність використання програми та створити негативне враження, навіть якщо її функціональна частина працює бездоганно.

Крім того, актуальним є тестування сумісності із різними версіями операційної системи Windows. Оскільки архітектура системного реєстру може частково відрізнятись між версіями, а також змінюватись із оновленнями, необхідно переконатися, що програмне забезпечення однаково ефективно працює як на Windows 10, так і на Windows 11 та подібних платформах.

Таким чином, застосування комплексного підходу, що включає функціональне тестування, тестування на відмову, перевірку швидкодії, тестування графічного інтерфейсу та сумісності з різними ОС, дозволяє забезпечити високу якість, стабільність та зручність програмного забезпечення у повсякденному використанні.

4.2 Тестування програмного застосунку

Для підтвердження працездатності та відповідності функціональним вимогам було проведено тестування програмного застосунку Windows Tuner. Останню

версію програми було інстальовано на тестовий комп’ютер із попередньо налаштованою операційною системою Windows 11 Home (версія 23H2).

Першим етапом було функціональне тестування, що охоплювало перевірку ключових можливостей програми. Було проаналізовано:

– точність зчитування існуючих параметрів системного реєстру;

– правильність внесення змін до реєстру та їх збереження;

– коректність генерації резервних копій у форматі .reg;

– відповідність текстових повідомлень реальним результатам виконання операцій.

За результатами тестування встановлено, що застосунок виконує усі заплановані дії згідно зі специфікацією. У жодному з випадків не спостерігалося збоїв, аварійного завершення роботи чи неправильного відображення результатів.

На наступному етапі було проведено тестування на відмову. Для цього були змодельовані потенційно проблемні ситуації:

– спроба створити резервну копію в директорію без прав запису;

– спроба змінити параметри реєстру без відповідних адміністративних прав.

Програма належним чином відреагувала на всі виняткові ситуації, без аварійного завершення роботи. Користувач отримував відповідні повідомлення через графічні вікна або логи. Це свідчить про ефективну реалізацію системи обробки помилок та логування, що значно підвищує надійність застосунку.

Далі було виконано тестування швидкодії, що дало змогу оцінити продуктивність програми при виконанні базових операцій:

– зміна окремого параметра у реєстрі відбувалася миттєво (менше 0,1 секунди);

– створення повної резервної копії реєстру (приблизно 30 ключів) займало до 0,5 секунди;

– повторне відновлення даних також здійснювалося без затримок.

Програма демонструвала стабільну продуктивність навіть на обладнанні середнього рівня, що підтверджує оптимальність реалізованих алгоритмів.

На окремому етапі було проведено тестування інтерфейсу користувача. У ході цього тестування:

– перевірено роботу світлої та темної тем оформлення;

– протестовано навігацію між вкладками «Оптимізація», «Кастомізація» та «Конфіденційність»;

– перевірено роботу кнопок та інтерактивних елементів (перемикачів, чекбоксів, списків) у всіх розділах програми;

– перевірено роботу меню та відповідність пунктів меню заявленому функціоналу;

– перевірено логіку відображення активних/неактивних пунктів залежно від поточного контексту;

– перевірено масштабування інтерфейсу на екранах із різними роздільними здатностями.

Інтерфейс працював стабільно, усі елементи були доступними та коректно відображались незалежно від налаштувань системи.

Тестування сумісності проводилося на двох актуальних версіях Windows:

– Windows 10 Pro (22H2);

– Windows 11 Home (23H2).

На обох платформах програма працювала стабільно: усі функції виконувалися без помилок, резервні копії створювалися і відновлювалися, інтерфейс не мав ознак некоректної роботи.

Таким чином, результати всебічного тестування програмного застосунку Windows Tuner підтверджують його відповідність вимогам до функціональності, стабільності, сумісності та продуктивності. Реалізовані механізми логування змін, обробки помилок та резервного копіювання дозволяють забезпечити високу надійність при щоденному використанні.

4.3 Розробка інструкції для користувача програмного забезпечення

Дана інструкція призначена для користування програмним продуктом «WindowsTuner» – застосунок для налаштування Windows за допомогою

системного реєстру.

Для інсталяції застосунка необхідно виконати такі кроки:

1. Завантажте програмний застосунок.

2. Запустіть програму, після чого з’явиться головне вікно програмного застосунку (див. рисунок 4.1).

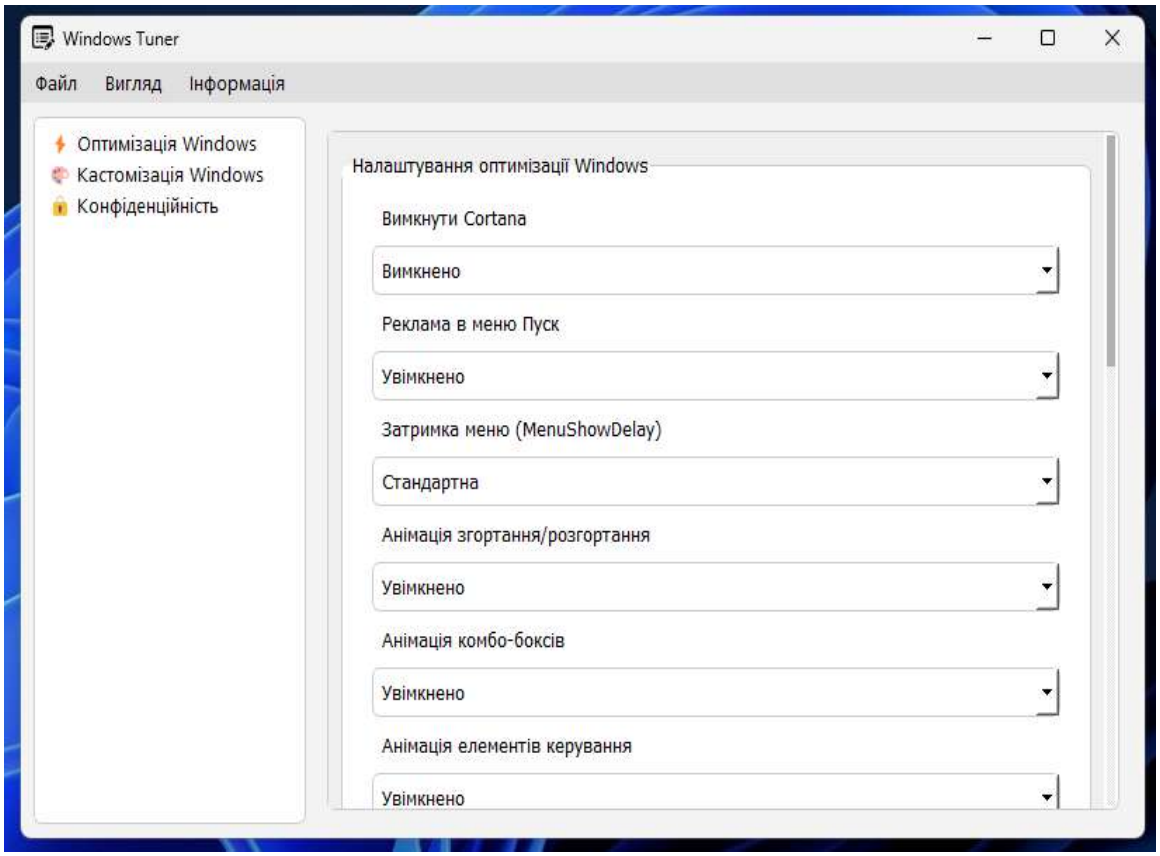

Рисунок 4.1 – Головне вікно

Основними можливостями застосунку є зміна параметрів реєстру , створення резервної копії налаштувань перед внесенням змін, введення журналу змін, а також можливість вибору теми оформлення інтерфейсу.

Використання функцій додатку можливе за допомогою пунктів меню, дані елементи інтерфейсу наведено на рисунку 4.2.

Меню складається з 3 пунктів:

– «Файл»;

– «Вигляд»;

– «Інформація».

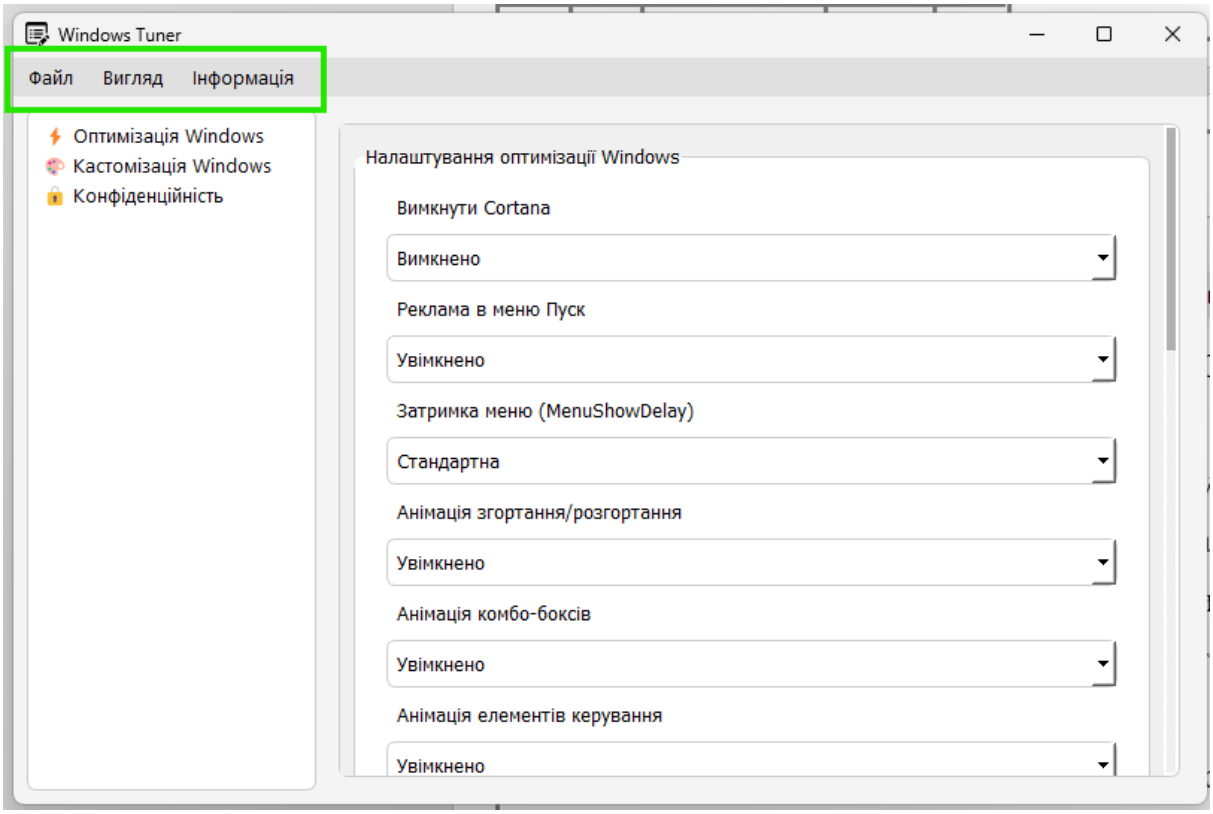

Рисунок 4.2 – Стрічка меню

Кожний з них має свої підпункти, пункт «Файл» складається з 3 підпунктів:

– «Логування» – вимикає або вмикає фіксацію змін;

– «Зберегти резервну копію» – даний підпункт має власний список, де слід обрати по якій категорії налаштувань системи, буде зроблено резервну копію;

– «Вихід» – закриває програму.

На рисунку 4.3 продемонстровано розгорнутий підпункт «Файл».

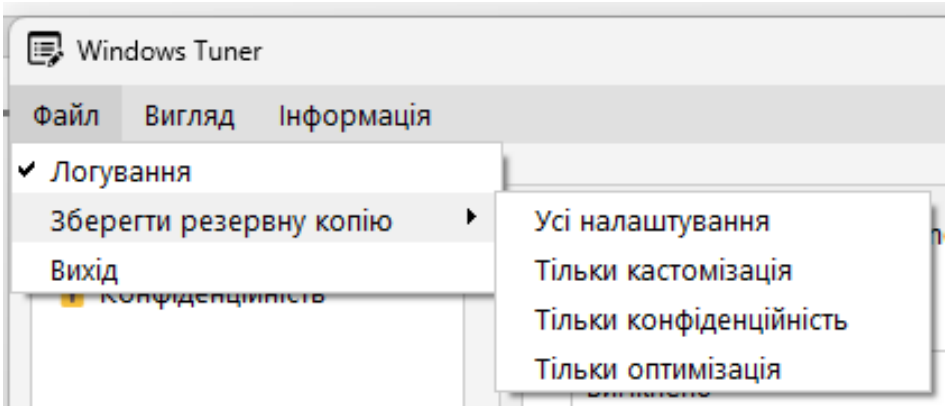

Рисунок 4.3 – Розгорнутий підпункт меню «Файл»

Підпункт меню «Вигляд» має функціонал для зміни кольорової схеми додатку (див. рисунок 4.4).

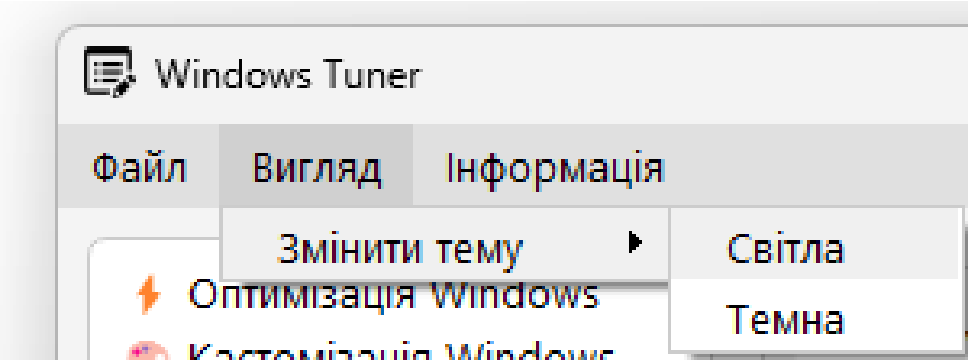

Рисунок 4.4 – Розгорнутий підпункт меню «Вигляд»

На рисунку 4.5 продемонстровано темну кольорову схему додатку, на прикладі головного вікна.

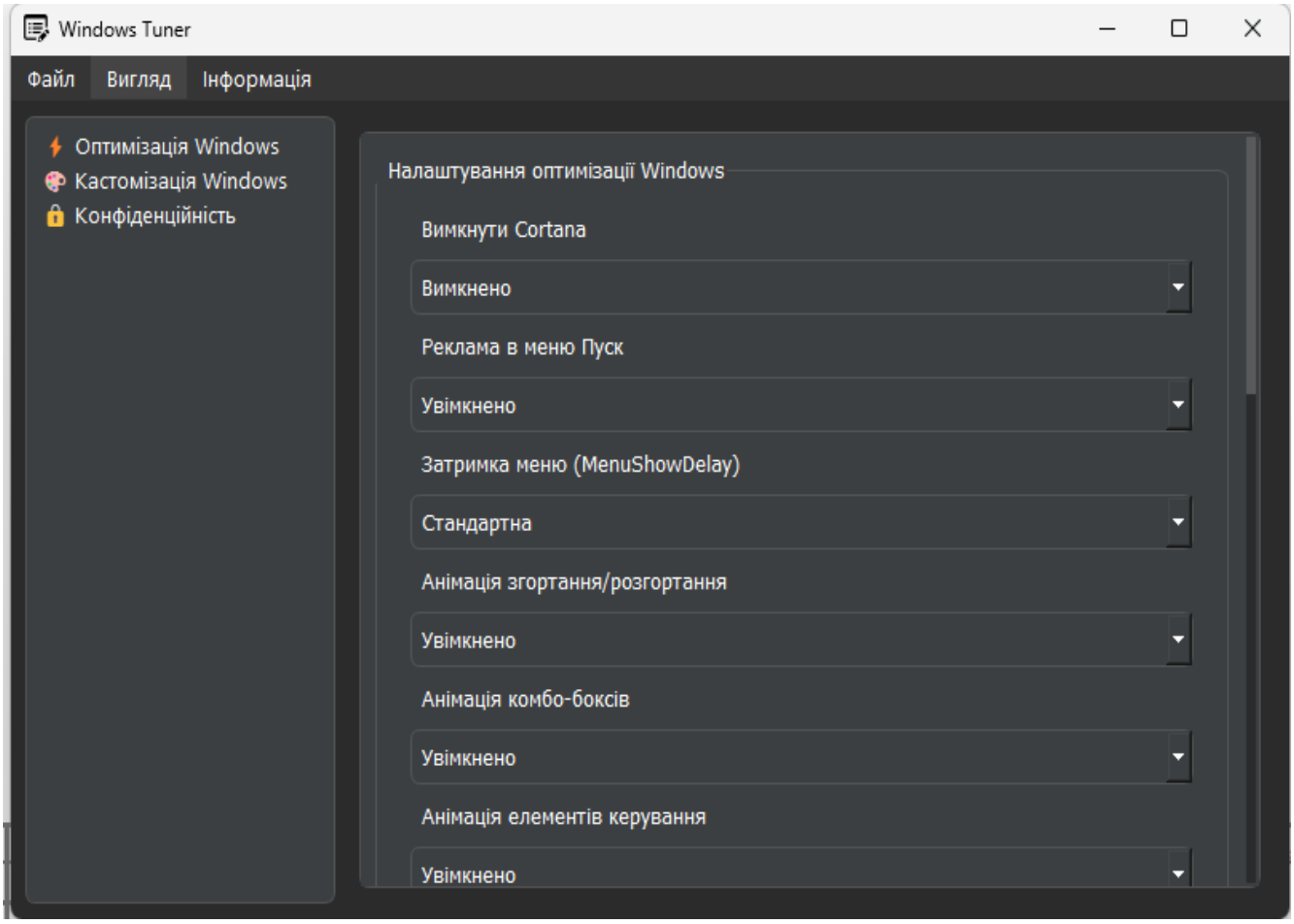

Рисунок 4.5 – Головне вікно у темному режимі

Підпункт меню «Інформація» надає можливість переглянути інформацію про застосунок (назва, версія, розробник, призначення), на рисунку 4.6 продемонстровано розгорнутий підпункт меню.

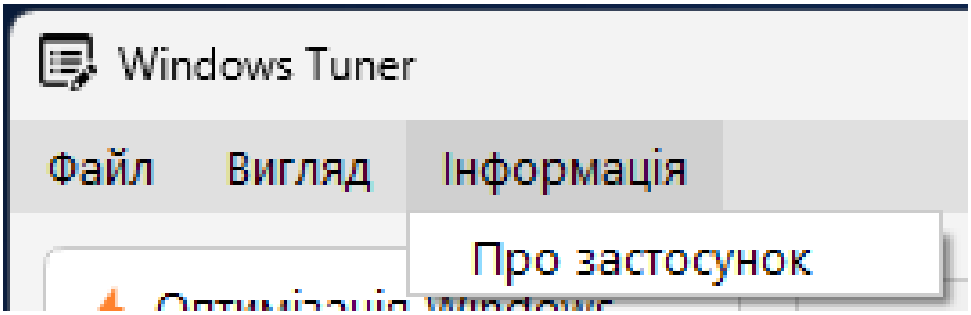

Рисунок 4.6 – Розгорнутий підпункт меню «Інформація»

На рисунку 4.7 продемонстровано модульне вікно, з інформацією про застосунок.

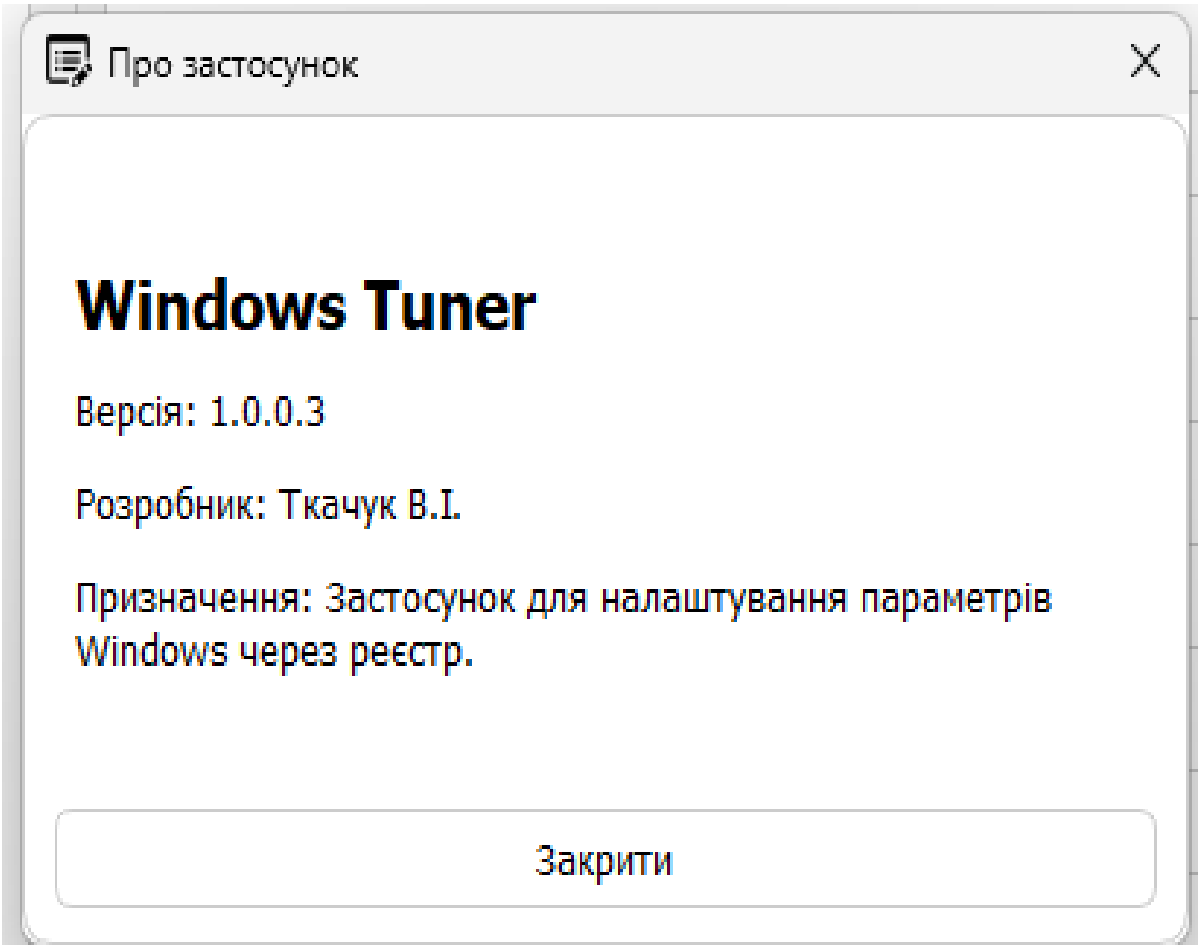

Рисунок 4.7 – Модульне вікно «Про застосунок»

Щоб працювати з параметрами реєстру, потрібно спочатку обрати категорію налаштувань в навігації додатку, після чого в робочій області програми будуть відображенні відповідні елементи керування (див. рисунок 4.8).

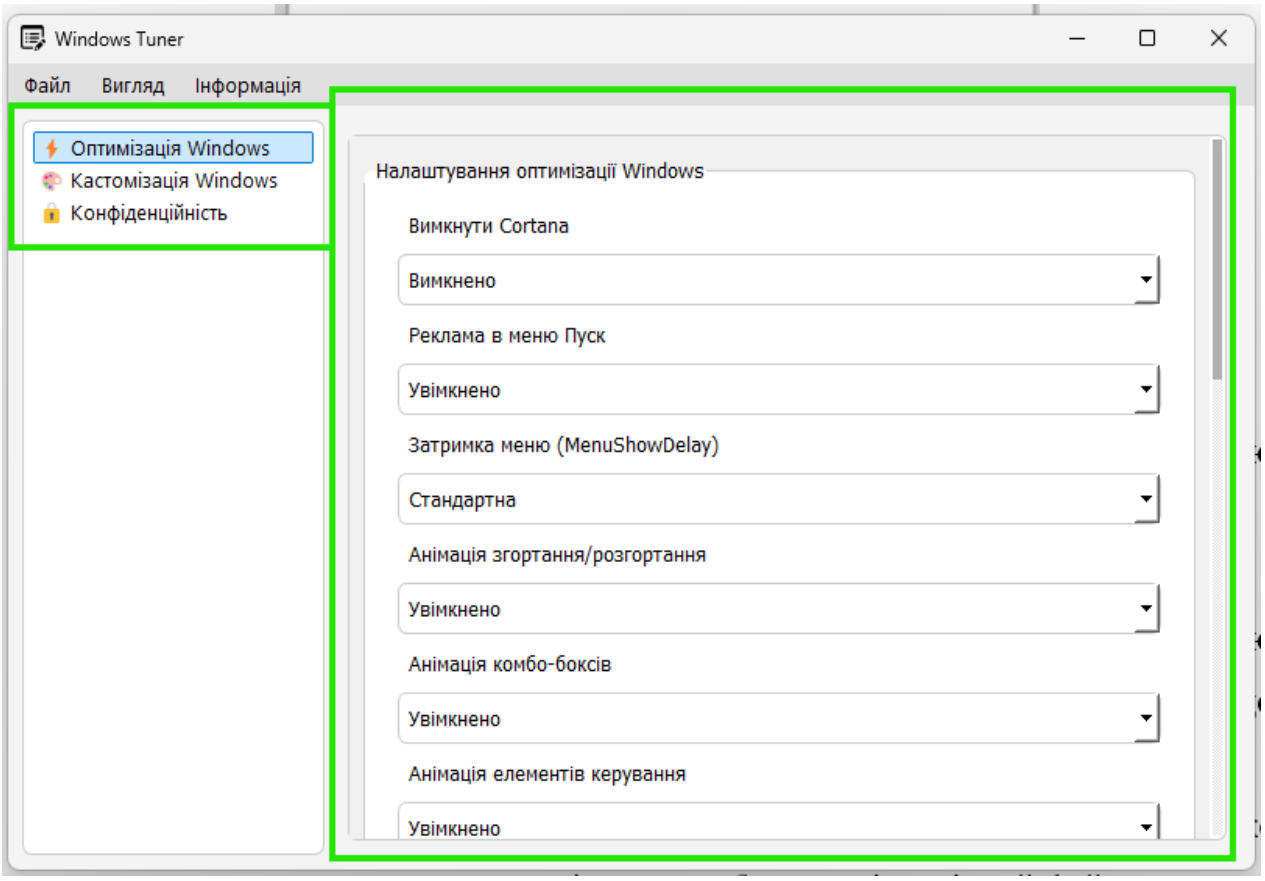

Рисунок 4.8 – Відкритий модуль «Оптимізація»

Щоб змінити параметр реєстру слід натиснути на відповідний дроп-бокс, після чого відобразиться список можливих значень (див. рисунок 4.9).

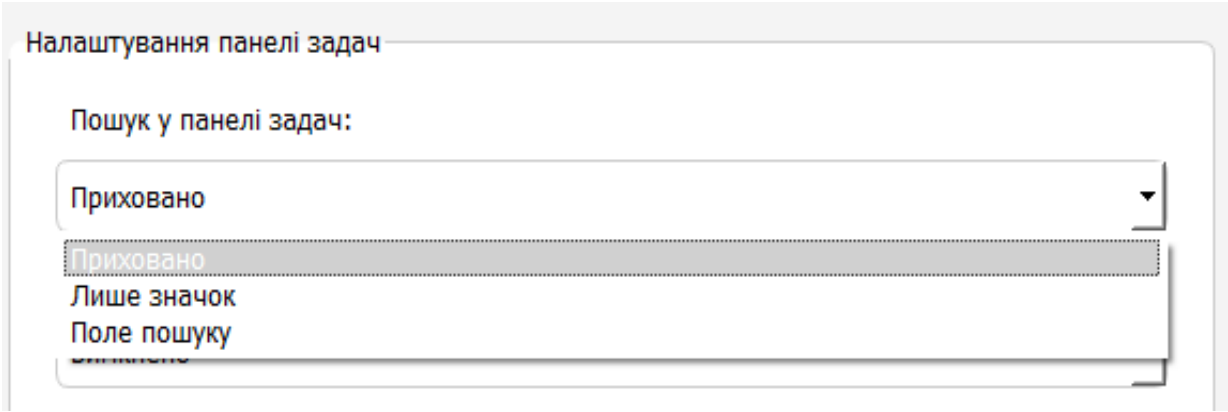

Рисунок 4.9 – Розгорнутий дроп-бокс

Щоб застосувати внесенні зміни до реєстру, слід натиснути кнопку «Застосувати» (див. рисунок 4.10).

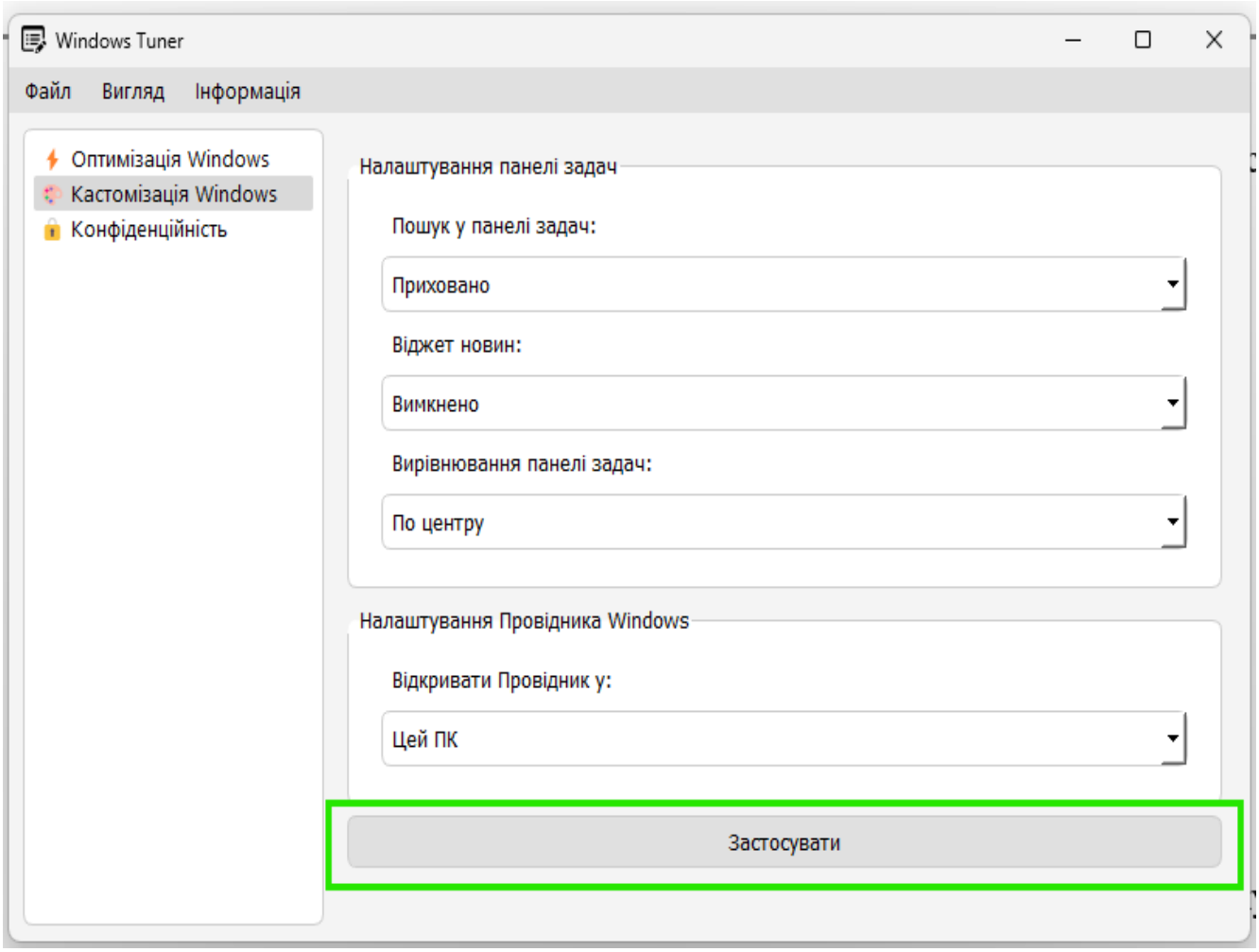

Рисунок 4.10 – Кнопка «Застосувати»

Якщо логування увімкнуто, то всі дії користувача записуються у файл, приклад вмісту файлу наведено на рисунку 4.11.

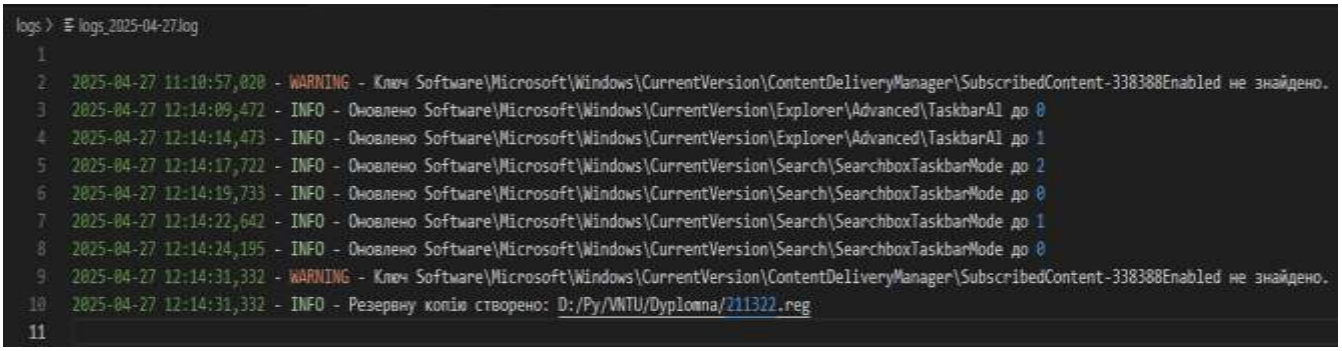

Рисунок 4.11 – Приклад вмісту файлу логів

Для того щоб створити резервну копію слід натиснути на пункт меню «Файл», у відкритому списку обрати пункт «Зберегти резервну копію», де слід обрати, які саме значення налаштувань треба зберегти. Приклад вмісту файл резервної копію продемонстровано на рисунку 4.12.

all.reg

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Search]
"SearchboxTaskbarMode"=dword:00000000

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Feeds]
"ShellFeedsTaskbarViewMode"=dword:00000002

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced]
"TaskbarAl"=dword:00000001

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced]
"LaunchTo"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\DataCollection]
"AllowTelemetry"=dword:00000000

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Privacy]
"TailoredExperiencesWithDiagnosticDataEnabled"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\LocationAndSensors]
"DisableLocation"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\LocationAndSensors]
"DisableWindowsLocationProvider"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\AdvertisingInfo]
"DisabledByGroupPolicy"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System]
"EnableActivityFeed"=dword:00000000

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System]
"PublishUserActivities"=dword:00000000

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System]
"UploadUserActivities"=dword:00000000

[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\Windows Search]
"AllowCortana"=dword:00000000

[HKEY_CURRENT_USER\Control Panel\Desktop]
"MenuShowDelay"="400"

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\AnimateMinMax]
"DefaultApplied"=dword:00000001
```

Рисунок 4.12 – Приклад вмісту файлу резервної копії

Щоб відновити дані з резервної копії потрібно натиснути правою кнопкою миші по файлу, та в контексному меню обрати «Запустити від імені Адміністратора», адже деякі зміни в реєстру потребують права адміністратора.

Інтерфейс застосунку орієнтований на максимальну простоту та інтуїтивність використання, що дозволяє працювати з налаштуваннями Windows навіть користувачам без глибоких технічних знань.

Таким чином, дана інструкція охоплює базові аспекти роботи із застосунком: запуск, навігацію по категоріях налаштувань, застосування змін, створення резервних копій, зміну теми оформлення та перегляд журналу змін. Застосунок створено таким чином, щоб забезпечити користувачу безпечну та зручну зміну системних параметрів без необхідності прямого редагування реєстру вручну.

4.4 Висновки

У четвертому розділі проведено тестування програмних модулів застосунку. Було виконано функціональне тестування, тестування відмов, тестування швидкодії, тестування інтерфейсу користувача та тестування сумісності. Перевірено реакцію застосунку на типові та виняткові ситуації. Результати тестування засвідчили правильність реалізації основного функціоналу, стабільність роботи та високу швидкодію системи. Також було розроблено детальну інструкцію для користувача.

ВИСНОВКИ

У рамках кваліфікаційної роботи розроблено програмний застосунок, призначений для зміни налаштувань операційної системи Windows через редагування параметрів системного реєстру. Для розробки використовувалося середовище програмування VSCode, а також мова програмування Python з використанням бібліотеки PyQt5 для створення графічного інтерфейсу користувача та бібліотеки winreg для доступу до системного реєстру.

Реалізація кваліфікаційної роботи відповідає методичним вказівкам [26].

Було проведено аналіз наявних аналогів програм для налаштування Windows, вивчено їхні недоліки та сформульовано основні вимоги до власного розробленого продукту.

Під час аналізу технологій розробки обґрунтовано вибір мови Python для розробки застосунку, а також вибір бібліотек для роботи з системним реєстром і побудови графічного інтерфейсу.

У процесі виконання бакалаврської кваліфікаційної роботи було здійснено:

– розробку окремих програмних модулів для взаємодії з реєстром та логування подій;

– створення зручного, адаптивного графічного інтерфейсу користувача з підтримкою зміни тем;

– проведення повного тестування програмного продукту відповідно до поставлених задач.

Тестування програмного забезпечення підтвердило його працездатність, високу швидкодію, стабільність функціонування, зручність інтерфейсу та відповідність технічному завданню. Також було розроблено детальну інструкцію для користувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сучасний розвиток технологій для зміни реєстру Windows. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry (дата звернення: 28.04.2025)

2. Russinovich, M., Solomon, D., Ionescu, A. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more. 7th Edition. Microsoft Press, 2021. 864 p.

3. В. П. Майданюк, В.І. Ткачук. Налаштування Windows з використанням системного реєстру «IV Міжнародна науково-практична конференція «Інформаційні технології в освіті та науці» (2025) . [Електронний ресурс]. URL: https://mdpu.org.ua/nauka/konferentsiyi-ta-seminary/.

4. Winaero Tweaker. [Електронний ресурс]. URL: https://winaero.com/winaero-tweaker/ (дата звернення: 29.04.2025)

5. O&O ShutUp10++. [Електронний ресурс]. URL: https://www.oosoftware.com/en/shutup10 (дата звернення: 29.04.2025)

6. Registry Editor. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/regedit (дата звернення: 28.04.2025)

7. Glary Utilities. [Електронний ресурс]. URL: https://www.glarysoft.com/glary-utilities/ (дата звернення: 29.04.2025)

8. Advanced SystemCare. [Електронний ресурс]. URL: https://www.iobit.com/en/advancedsystemcarefree.php (дата звернення: 29.04.2025)

9. Process and route data with data flows - Azure IoT Operations. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/azure/iot-operations/connect-to-cloud/overview-dataflow (дата звернення: 30.04.2025).

10. Monitor Windows Registry data - Splunk Documentation. [Електронний ресурс]. URL: https://docs.splunk.com/Documentation/SplunkC loud/latest/Data/Monitor Windowsregistrydata (дата звернення: 02.05.2025).

11. Draw.io. [Електронний ресурс]. URL: https://app.diagrams.net/ (дата звернення: 05.05.2025)

12. Python. [Електронний ресурс]. URL: https://www.python.org/ (дата звернення: 07.05.2025)

13. C++. [Електронний ресурс]. URL: https://isocpp.org/ (дата звернення: 07.05.2025)

14. Java. [Електронний ресурс]. URL: https://www.oracle.com/java/technologies/javase-downloads.html (дата звернення: 07.05.2025)

15. C#. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 28.04.2025)

16. Visual Studio Code. [Електронний ресурс]. URL: https://code.visualstudio.com/ (дата звернення: 10.05.2025)

17. PyCharm. [Електронний ресурс]. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 10.05.2025)

18. Sublime Text. [Електронний ресурс]. URL: https://www.sublimetext.com/ (дата звернення: 10.05.2025)

19. Atom. [Електронний ресурс]. URL: https://github.com/atom/atom (дата звернення: 10.05.2025)

20. Eclipse. [Електронний ресурс]. URL: https://www.eclipse.org/ (дата звернення: 10.05.2025)

21. Tkinter. [Електронний ресурс]. URL: https://docs.python.org/3/library/tk.html (дата звернення: 15.05.2025)

22. PyQt5. [Електронний ресурс]. URL: https://riverbankcomputing.com/software/pyqt/intro (дата звернення: 15.05.2025)

23. winreg. [Електронний ресурс]. URL: https://docs.python.org/3/library/winreg.html (дата звернення: 15.05.2025)

24. pywin32. [Електронний ресурс]. URL: https://github.com/mhammond/pywin32 (дата звернення: 15.05.2025)

25. ctypes. [Електронний ресурс]. URL: https://docs.python.org/3/library/ctypes.html (дата звернення: 15.05.2025)

26. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 121 "Інженерія програмного забезпечення" / Уклад. О. Н. Романюк, Г. О. Черноволик – Вінниця: ВНТУ, 2022. – 52с.

ДОДАТКИ

59

ДОДАТОК А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
« 21 » березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу «Розробка програмного забезпечення для налаштування Windows з використанням системного реєстру»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. Майданюк В.П.
« 21 » березня 2025 року.
Виконав:
студент гр. 1ПІ-21б Ткачук В.І.
« 21 » березня 2025 року.

м. Вінниця – 2025 рік

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: «Розробка програмного забезпечення для налаштування Windows з використанням системного реєстру».

Галузь застосування – налаштування операційної системи Windows.

2. Підстава для розробки.

Підставою для виконання бакалаврської кваліфікаційної роботи (БКР) є індивідуальне завдання на БКР та наказ №97 від 20 березня 2025 р. ректора ВНТУ про закріплення тем БКР.

3. Мета та призначення розробки.

Метою роботи є підвищення зручності та безпеки процесу налаштування операційної системи Windows шляхом розробки програмного забезпечення з інтуїтивним графічним інтерфейсом, що дозволяє змінювати параметри системи через системний реєстр.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР:

1. В. П. Майданюк, В.І. Ткачук. Налаштування Windows з використанням системного реєстру «IV Міжнародна науково-практична конференція «Інформаційні технології в освіті та науці» (2025).

2. Russinovich, M., Solomon, D., Ionescu, A. Windows Internals, Part 1: System architecture, processes, threads, memory management, and more. 7th Edition. Microsoft Press, 2021. 864 p.

3. Сучасний розвиток технологій для зміни реєстру Windows. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry (дата звернення: 28.04.2025)

4. Process and route data with data flows - Azure IoT Operations. [Електронний ресурс]. URL: https://learn.microsoft.com/en-us/azure/iot-operations/connect-to-cloud/overview-dataflow (дата звернення: 30.04.2025).

5. Python. [Електронний ресурс]. URL: https://www.python.org/ (дата звернення: 07.05.2025)

5. Технічні вимоги

– операційна система – Windows;

– середовище розробки – Visual Studio Code;

– мова програмування – Python.

6. Конструктивні вимоги.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

–пояснювальна записка до бакалаврської кваліфікаційної роботи;

–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 бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану розвитку програмних застосунків для редагування реєстру та обгрунтування задач розробки	25.03.25 – 31.03.25
2	Розробка методів, моделі та алгоритму роботи програмного забезпечення	03.04.25 – 14.04.25
3	Варіантний аналіз та вибір мови програмування розробки	17.04.25 – 21.04.25
4	Розробка програмного забезпечення	24.04.25 – 12.05.25
5	Тестування програмного забезпечення	15.05.25 – 19.05.25
6	Оформлення матеріалів до захисту БКР	22.05.25 – 30.05.25

10. Порядок контролю та прийняття.

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

62

ДОДАТОК Б.

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Розробка програмного забезпечення для налаштування Windows з використанням системного реєстру»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, факультет інформаційних технологій та комп'ютерної інженерії, ІПІ-21Б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 4,3 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

■ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Майданюк В.П. к.т.н., доц. каф. ПЗ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Ткачук В.І.
(підпис) (прізвище, ініціали)

ДОДАТОК В. Лістинг програми

main.py

```
import sys
from PyQt5.QtWidgets import QApplication
from windows.MainWindow import MainWindow

if __name__ == "__main__":
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())
```

MainWindow.py

```
import winreg
from PyQt5.QtWidgets import (
    QMainWindow, QWidget, QHBoxLayout, QListWidget, QStackedWidget,
    QMenuBar, QAction, QFileDialog, QMenu
)
from modules.CustomizationModule import CustomizationModule
from modules.OptimizationModule import OptimizationModule
from modules.PrivacyModule import PrivacyModule
from modules.AboutDialog import AboutDialog
from utils.RegistryManager import RegistryManager
from constants.registry_keys import REGISTRY_KEYS
from constants.themes import LIGHT_THEME, DARK_THEME
from utils.Logger import Logger

class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Windows Tuner")
        self.resize(800, 500)
        self.theme = "light"
        self.set_theme(self.detect_system_theme())
```

```
self.logging_enabled = True
Logger.enabled = self.logging_enabled

menubar = self.menuBar()
file_menu = menubar.addMenu("Файл")

# Пункт "Логування"
toggle_logging_action = QAction("Логування", self)
toggle_logging_action.setCheckable(True)
toggle_logging_action.setChecked(self.logging_enabled)
toggle_logging_action.triggered.connect(self.toggle_logging)
file_menu.addAction(toggle_logging_action)

# Підменю "Зберегти резервну копію"
backup_menu = QMenu("Зберегти резервну копію", self)

backup_all = QAction("Усі налаштування", self)
backup_all.triggered.connect(lambda: self.create_backup("all"))
backup_customization = QAction("Тільки кастомізація", self)
backup_customization.triggered.connect(lambda: self.create_backup("customization"))
backup_privacy = QAction("Тільки конфіденційність", self)
backup_privacy.triggered.connect(lambda: self.create_backup("privacy"))
backup_optimization = QAction("Тільки оптимізація", self)
backup_optimization.triggered.connect(lambda: self.create_backup("optimization"))

backup_menu.addAction(backup_all)
backup_menu.addAction(backup_customization)
backup_menu.addAction(backup_privacy)
backup_menu.addAction(backup_optimization)

file_menu.addMenu(backup_menu)

# Вихід
exit_action = QAction("Вихід", self)
```

```
exit_action.triggered.connect(self.close)
file_menu.addAction(exit_action)

# Вигляд
view_menu = menubar.addMenu("Вигляд")
theme_menu = view_menu.addMenu("Змінити тему")
light_theme_action = QAction("Світла", self)
dark_theme_action = QAction("Темна", self)
light_theme_action.triggered.connect(lambda: self.set_theme("light"))
dark_theme_action.triggered.connect(lambda: self.set_theme("dark"))
theme_menu.addAction(light_theme_action)
theme_menu.addAction(dark_theme_action)

# Інформація
info_menu = menubar.addMenu("Інформація")
about_action = QAction("Про застосунок", self)
about_action.triggered.connect(self.show_about_dialog)
info_menu.addAction(about_action)

# Основний інтерфейс
main_widget = QWidget()
self.setCentralWidget(main_widget)
main_layout = QHBoxLayout()
main_widget.setLayout(main_layout)

self.nav_menu = QListWidget()
self.nav_menu.addItem("⚡ Оптимізація Windows")
self.nav_menu.addItem("🎨 Кастомізація Windows")
self.nav_menu.addItem("🔒 Конфіденційність")
self.nav_menu.currentRowChanged.connect(self.display_page)
main_layout.addWidget(self.nav_menu, 1)

self.content_stack = QStackedWidget()
self.opt_page = OptimizationModule()
self.cust_page = CustomizationModule()
```

```
        self.privacy_page = PrivacyModule()

        self.content_stack.addWidget(self.opt_page)
        self.content_stack.addWidget(self.cust_page)
        self.content_stack.addWidget(self.privacy_page)
        main_layout.addWidget(self.content_stack, 3)

        self.set_theme(self.detect_system_theme())

    def display_page(self, index):
        self.content_stack.setCurrentIndex(index)

    def create_backup(self, key_type):
        options = QFileDialog.Options()
        backup_file, _ = QFileDialog.getSaveFileName(
            self,
            "Зберегти резервну копію",
            "",
            "Файли .reg (*.reg)",
            options=options
        )
        if backup_file:
            if key_type == "all":
                keys_to_backup = REGISTRY_KEYS
            else:
                keys_to_backup = {
                    name: info for name, info in REGISTRY_KEYS.items()
                    if info.get("type") == key_type
                }

            RegistryManager.backup_registry_keys_to_reg(keys_to_backup, backup_file)

    def set_theme(self, theme):
        self.theme = theme
        if theme == "light":
```

```
            self.setStyleSheet(LIGHT_THEME)
        elif theme == "dark":
            self.setStyleSheet(DARK_THEME)

    def detect_system_theme(self):
        try:
            with winreg.OpenKey(
                winreg.HKEY_CURRENT_USER,
                r"Software\Microsoft\Windows\CurrentVersion\Themes\Personalize"
            ) as key:
                value, _ = winreg.QueryValueEx(key, "AppsUseLightTheme")
                return "light" if value == 1 else "dark"
        except Exception:
            return "light"

    def toggle_logging(self, checked):
        self.logging_enabled = checked
        Logger.enabled = checked
        print(f"Логування {'увімкнено' if checked else 'вимкнено'}.")

    def show_about_dialog(self):
        dialog = AboutDialog(self)
        dialog.set_theme(self.theme)
        dialog.exec_()
```

PrivacyModule.py

```
from PyQt5.QtWidgets import QWidget, QVBoxLayout, QLabel, QComboBox, QPushButton, QGroupBox
from utils.RegistryManager import RegistryManager
from constants.registry_keys import REGISTRY_KEYS

class PrivacyModule(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.init_ui()
```

```
    def init_ui(self):
        layout = QVBoxLayout()

        # 🔒 Група конфіденційності
        privacy_group = QGroupBox("Налаштування конфіденційності Windows")
        privacy_layout = QVBoxLayout()

        self.telemetry_dropdown = self._create_dropdown("Надсилання діагностичних даних:")
        privacy_layout.addWidget(QLabel("Надсилання діагностичних даних:"))
        privacy_layout.addWidget(self.telemetry_dropdown)

        self.tailored_dropdown = self._create_dropdown("Персональні підказки та поради на основі вашої активності в Windows.")
        privacy_layout.addWidget(QLabel("Персональні підказки та поради на основі вашої активності в Windows."))
        privacy_layout.addWidget(self.tailored_dropdown)

        self.location_dropdown = self._create_dropdown("Відстеження місцезнаходження:")
        privacy_layout.addWidget(QLabel("Відстеження місцезнаходження:"))
        privacy_layout.addWidget(self.location_dropdown)

        self.ads_id_dropdown = self._create_dropdown("Персоналізована реклама:")
        privacy_layout.addWidget(QLabel("Персоналізована реклама:"))
        privacy_layout.addWidget(self.ads_id_dropdown)

        self.activity_feed_dropdown = self._create_dropdown("Журнал активності:")
        privacy_layout.addWidget(QLabel("Журнал активності:"))
        privacy_layout.addWidget(self.activity_feed_dropdown)

        privacy_group.setLayout(privacy_layout)
        layout.addWidget(privacy_group)

        # Кнопка застосування
        self.apply_button = QPushButton("Застосувати")
```

```
        self.apply_button.clicked.connect(self.apply_changes)
        layout.addWidget(self.apply_button)

        layout.addStretch(1)
        self.setLayout(layout)

        self.load_current_settings()

    def _create_dropdown(self, label_text):
        dropdown = QComboBox()
        dropdown.addItems(["Увімкнено", "Вимкнено"])
        return dropdown

    def load_current_settings(self):
        """Завантаження поточних значень з реєстру"""
        self.set_dropdown_value(self.telemetry_dropdown,
REGISTRY_KEYS["ALLOW_TELEMETRY"])
        self.set_dropdown_value(self.tailored_dropdown,
REGISTRY_KEYS["TAILORED_EXPERIENCES"])
        self.set_dropdown_value(self.location_dropdown,
REGISTRY_KEYS["DISABLE_LOCATION"])
        self.set_dropdown_value(self.ads_id_dropdown,
REGISTRY_KEYS["DISABLE_ADVERTISING_ID"])
        self.set_dropdown_value(self.activity_feed_dropdown,
REGISTRY_KEYS["ENABLE_ACTIVITY_FEED"])

    def apply_changes(self):
        """Застосування змін до реєстру"""
        self.set_registry_value(REGISTRY_KEYS["ALLOW_TELEMETRY"],
self.telemetry_dropdown.currentIndex())
        self.set_registry_value(REGISTRY_KEYS["TAILORED_EXPERIENCES"],
self.tailored_dropdown.currentIndex())
        self.set_registry_value(REGISTRY_KEYS["DISABLE_LOCATION"],  0  if
self.location_dropdown.currentIndex() == 1 else 1)
```

```
        self.set_registry_value(REGISTRY_KEYS["DISABLE_ADVERTISING_ID"], 1 if self.ads_id_dropdown.currentIndex() == 1 else 0)
        self.set_registry_value(REGISTRY_KEYS["ENABLE_ACTIVITY_FEED"], self.activity_feed_dropdown.currentIndex())

        print("Конфіденційні параметри оновлено.")

    def set_registry_value(self, key, value):
        current_value = RegistryManager.get_value(key)
        if current_value != value:
            RegistryManager.set_value(key, value)

    def set_dropdown_value(self, dropdown, key):
        value = RegistryManager.get_value(key)
        if value is not None and value in (0, 1):
            dropdown.setCurrentIndex(value)
```

OptimizationModule.py

```
import winreg
from PyQt5.QtWidgets import QWidget, QVBoxLayout, QLabel, QComboBox, QPushButton, QGroupBox, QScrollArea
from utils.RegistryManager import RegistryManager
from constants.registry_keys import REGISTRY_KEYS

# Додаємо кастомний QComboBox з вимкненою прокруткою
class NonWheelComboBox(QComboBox):
    def wheelEvent(self, event):
        event.ignore()  # Ігноруємо подію прокрутки колесиком


class OptimizationModule(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.init_ui()
```

```
def init_ui(self):
    # Обгортка зі скроллом
    scroll_area = QScrollArea()
    scroll_area.setWidgetResizable(True)

    scroll_content = QWidget()
    scroll_layout = QVBoxLayout(scroll_content)
    self.dropdowns = {}

    group = QGroupBox("Налаштування оптимізації Windows")
    group_layout = QVBoxLayout()

    settings = [
        ("Вимкнути Cortana", "ALLOW_CORTANA"),
        ("Реклама в меню Пуск", "START_ADS"),
        ("Затримка меню (MenuShowDelay)", "MENU_SHOW_DELAY"),
        ("Анімація згортання/розгортання", "ANIMATE_MIN_MAX"),
        ("Анімація комбо-боксів", "COMBOBOX_ANIMATION"),
        ("Анімація елементів керування", "CONTROL_ANIMATIONS"),
        ("Тіні курсора", "CURSOR_SHADOW"),
        ("Повний вміст вікна при перетягуванні", "DRAG_FULL_WINDOWS"),
        ("Тіні під текстом піктограм", "DROP_SHADOW"),
        ("Aero Peek", "AERO_PEEK"),
        ("Плавне прокручування списків", "LISTBOX_SMOOTH_SCROLLING"),
        ("Підсвічування вибору (ListView)", "LISTVIEW_ALPHA_SELECT"),
        ("Тіні у списках (ListView)", "LISTVIEW_SHADOW"),
        ("Анімація виділення", "SELECTION_FADE"),
        ("Анімація панелі завдань", "TASKBAR_ANIMATIONS"),
        ("Ескізи замість іконок", "THUMBNAILS_OR_ICON"),
        ("Анімація підказок", "TOOLTIP_ANIMATION")
    ]

    for label, key in settings:
        group_layout.addWidget(QLabel(label))
        cb = NonWheelComboBox()
```

```
            if key == "MENU_SHOW_DELAY":
                cb.addItems(["Стандартна", "Миттєво"])
            else:
                cb.addItems(["Вимкнено", "Увімкнено"])
            self.dropdowns[key] = cb
            group_layout.addWidget(cb)

        group.setLayout(group_layout)
        scroll_layout.addWidget(group)

        self.apply_button = QPushButton("Застосувати")
        self.apply_button.clicked.connect(self.apply_changes)
        scroll_layout.addWidget(self.apply_button)
        scroll_layout.addStretch(1)

        scroll_area.setWidget(scroll_content)

        main_layout = QVBoxLayout(self)
        main_layout.addWidget(scroll_area)

        self.load_current_settings()

    def load_current_settings(self):
        for key, dropdown in self.dropdowns.items():
            value = RegistryManager.get_value(REGISTRY_KEYS[key])
            if key == "MENU_SHOW_DELAY":
                dropdown.setCurrentIndex(1 if value == "0" else 0)
            else:
                dropdown.setCurrentIndex(value if isinstance(value, int) and value in (0, 1) else 1)

    def apply_changes(self):
        for key, dropdown in self.dropdowns.items():
            current_value = RegistryManager.get_value(REGISTRY_KEYS[key])
            if key == "MENU_SHOW_DELAY":
                new_value = "0" if dropdown.currentIndex() == 1 else "200"
```

```
            else:
                new_value = dropdown.currentIndex()
            if current_value != new_value:
                RegistryManager.set_value(REGISTRY_KEYS[key], new_value,
value_type=winreg.REG_SZ if isinstance(new_value, str) else winreg.REG_DWORD)

        print("Налаштування оптимізації оновлено.")
```

CustomizationModule.py

```
from PyQt5.QtWidgets import QWidget, QVBoxLayout, QLabel, QComboBox, QPushButton,
QGroupBox
from utils.RegistryManager import RegistryManager
from constants.registry_keys import REGISTRY_KEYS

class CustomizationModule(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.init_ui()

    def init_ui(self):
        layout = QVBoxLayout()

        # Група налаштувань панелі задач
        taskbar_group = QGroupBox("Налаштування панелі задач")
        taskbar_layout = QVBoxLayout()

        self.search_dropdown = QComboBox()
        self.search_dropdown.addItems(["Приховано", "Лише значок", "Поле пошуку"])
        taskbar_layout.addWidget(QLabel("Пошук у панелі задач:"))
        taskbar_layout.addWidget(self.search_dropdown)
```

```
self.news_dropdown = QComboBox()
self.news_dropdown.addItems(["Увімкнено", "Лише значок", "Вимкнено"])
taskbar_layout.addWidget(QLabel("Віджет новин:"))
taskbar_layout.addWidget(self.news_dropdown)

self.taskbar_alignment_dropdown = QComboBox()
self.taskbar_alignment_dropdown.addItems(["Зліва", "По центру"])
taskbar_layout.addWidget(QLabel("Вирівнювання панелі задач:"))
taskbar_layout.addWidget(self.taskbar_alignment_dropdown)

taskbar_group.setLayout(taskbar_layout)
layout.addWidget(taskbar_group)

# Група налаштувань Провідника Windows
explorer_group = QGroupBox("Налаштування Провідника Windows")
explorer_layout = QVBoxLayout()

self.explorer_open_dropdown = QComboBox()
self.explorer_open_dropdown.addItems(["Швидкий доступ", "Цей ПК"])
explorer_layout.addWidget(QLabel("Відкривати Провідник у:"))
explorer_layout.addWidget(self.explorer_open_dropdown)

explorer_group.setLayout(explorer_layout)
layout.addWidget(explorer_group)

# Кнопка застосування змін
self.apply_button = QPushButton("Застосувати")
self.apply_button.clicked.connect(self.apply_changes)
layout.addWidget(self.apply_button)
layout.addStretch(1)
self.setLayout(layout)

self.load_current_settings()
```

```
    def load_current_settings(self):
        """Завантаження поточних значень з реєстру"""
        self.set_dropdown_value(self.search_dropdown, REGISTRY_KEYS["SEARCH"])
        self.set_dropdown_value(self.news_dropdown, REGISTRY_KEYS["NEWS_WIDGET"])
        self.set_dropdown_value(self.taskbar_alignment_dropdown,
REGISTRY_KEYS["TASKBAR_ALIGNMENT"])
        self.set_dropdown_value(self.explorer_open_dropdown,
REGISTRY_KEYS["EXPLORER_OPEN_THIS_PC"])

    def apply_changes(self):
        """Перевірка змін та застосування їх у реєстрі"""
        self.set_registry_value_if_changed(REGISTRY_KEYS["SEARCH"],
self.search_dropdown.currentIndex())
        self.set_registry_value_if_changed(REGISTRY_KEYS["NEWS_WIDGET"],
self.news_dropdown.currentIndex())
        self.set_registry_value_if_changed(REGISTRY_KEYS["TASKBAR_ALIGNMENT"],
self.taskbar_alignment_dropdown.currentIndex())
        self.set_registry_value_if_changed(REGISTRY_KEYS["EXPLORER_OPEN_THIS_PC"],
self.explorer_open_dropdown.currentIndex())

        print("Зміни застосовано. Може знадобитися перезапуск Explorer.exe")

    def set_registry_value_if_changed(self, key, value):
        """Перевірка змін та застосування їх у реєстрі, якщо значення змінилося"""
        current_value = RegistryManager.get_value(key)
        if isinstance(current_value, bytes):  # Якщо ключ очікує bytes, конвертуємо
            new_value = bytes([value]) + current_value[1:]
        else:
            new_value = value

        if current_value != new_value:
            RegistryManager.set_value(key, new_value)

    def set_dropdown_value(self, dropdown, key):
        """Допоміжний метод для встановлення значення випадаючого списку"""
```

```
        value = RegistryManager.get_value(key)
        if isinstance(value, bytes):  # Декодуємо UserPreferencesMask
            value = list(value)[0]
        if value is not None and isinstance(value, int) and 0 <= value < dropdown.count():
            dropdown.setCurrentIndex(value)
```

AboutDialog.py

```
from PyQt5.QtWidgets import QDialog, QVBoxLayout, QLabel, QPushButton
from PyQt5.QtCore import Qt
from constants.themes import LIGHT_THEME, DARK_THEME

class AboutDialog(QDialog):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.setWindowTitle("Про застосунок")
        self.setFixedSize(400, 250)
        self.setWindowFlags(self.windowFlags() & ~Qt.WindowContextHelpButtonHint)

        self.layout = QVBoxLayout()
        self.setLayout(self.layout)

        self.info_label = QLabel(
            "<h2>Windows Tuner</h2>"
            "<p>Версія: 1.0.0.3</p>"
            "<p>Розробник: Ткачук В.І.</p>"
            "<p>Призначення: Застосунок для налаштування параметрів Windows через реєстр.</p>"
        )
        self.info_label.setWordWrap(True)
        self.layout.addWidget(self.info_label)

        self.close_button = QPushButton("Закрити")
        self.close_button.clicked.connect(self.accept)
```

```
        self.layout.addWidget(self.close_button)

    def set_theme(self, theme):
        if theme == "dark":
            self.setStyleSheet(DARK_THEME)
        else:
            self.setStyleSheet(LIGHT_THEME)
```

Logger.py

```
import os
import logging
from datetime import datetime

class Logger:
    enabled = True

    def __init__(self, log_dir="logs"):
        self.log_dir = log_dir
        if not os.path.exists(self.log_dir):
            os.makedirs(self.log_dir)

        self.log_file = os.path.join(self.log_dir, f"logs_{datetime.now().strftime('%Y-%m-%d')}.log")
        self.logger = logging.getLogger(__name__)
        self.logger.setLevel(logging.INFO)

        file_handler = logging.FileHandler(self.log_file, encoding='utf-8')
        file_handler.setLevel(logging.INFO)

        formatter = logging.Formatter('%(asctime)s - %(levelname)s - %(message)s')
        file_handler.setFormatter(formatter)

        if not self.logger.handlers:
            self.logger.addHandler(file_handler)

    def info(self, message):
```

```
        if Logger.enabled:
            self.logger.info(message)

    def warning(self, message):
        if Logger.enabled:
            self.logger.warning(message)

    def error(self, message):
        if Logger.enabled:
            self.logger.error(message)
```

RegistryManager.py

```
import winreg
import subprocess
from utils.Logger import Logger
from constants.registry_keys import REGISTRY_KEYS

class RegistryManager:
    logger = Logger()

    @staticmethod
    def get_value(key_info):
        """Отримує значення з реєстру Windows за інформацією про ключ."""
        try:
            with winreg.OpenKey(
                key_info["hive"], key_info["path"], 0, winreg.KEY_READ | winreg.KEY_WOW64_64KEY
            ) as key:
                value, _ = winreg.QueryValueEx(key, key_info["name"])
                return value
        except FileNotFoundError:
            RegistryManager.logger.warning(
                f"Ключ {key_info['path']}\\{key_info['name']} не знайдено."
            )
            return None
```

```
@staticmethod
def set_value(key_info, value, value_type=winreg.REG_DWORD):
    """Оновлює значення в реєстрі Windows за інформацією про ключ."""
    try:
        try:
            key = winreg.OpenKey(
                key_info["hive"], key_info["path"], 0, winreg.KEY_SET_VALUE
            )
        except FileNotFoundError:
            key = winreg.CreateKey(key_info["hive"], key_info["path"])

        winreg.SetValueEx(key, key_info["name"], 0, value_type, value)
        winreg.CloseKey(key)

        RegistryManager.logger.info(
            f"Оновлено {key_info['path']}\\{key_info['name']} до {value}"
        )
    except Exception as e:
        RegistryManager.logger.error(
            f"Помилка оновлення {key_info['path']}\\{key_info['name']}: {e}"
        )

@staticmethod
def backup_registry_keys_to_reg(keys, backup_file):
    """Створює резервну копію реєстрових ключів у .reg файл."""
    try:
        hive_map = {
            winreg.HKEY_CLASSES_ROOT: "HKEY_CLASSES_ROOT",
            winreg.HKEY_CURRENT_USER: "HKEY_CURRENT_USER",
            winreg.HKEY_LOCAL_MACHINE: "HKEY_LOCAL_MACHINE",
            winreg.HKEY_USERS: "HKEY_USERS",
            winreg.HKEY_CURRENT_CONFIG: "HKEY_CURRENT_CONFIG",
        }
```

```
            with open(backup_file, "w", encoding="utf-8") as reg_file:
                reg_file.write("Windows Registry Editor Version 5.00\n\n")

                for key_name, key_info in keys.items():
                    hive = key_info["hive"]
                    path = key_info["path"]
                    name = key_info["name"]
                    value = RegistryManager.get_value(key_info)

                    if value is not None:
                        hive_str = hive_map.get(hive)
                        if not hive_str:
                            RegistryManager.logger.warning(f"Пропущено ключ {key_name} через невідомий
hive.")
                            continue

                        reg_path = f"{hive_str}\\{path}"
                        reg_file.write(f"[{reg_path}]\n")
                        if isinstance(value, int):
                            reg_file.write(f'"{name}"=dword:{value:08X}\n\n')
                        elif isinstance(value, str):
                            reg_file.write(f'"{name}"="{value}"\n\n')

                RegistryManager.logger.info(f"Резервну копію створено: {backup_file}")

        except Exception as e:
            RegistryManager.logger.error(f"Помилка при створенні резервної копії: {e}")



registry_keys.py
import winreg

REGISTRY_KEYS = {
    # Кастомізація
    "SEARCH": {
```

```
    "hive": winreg.HKEY_CURRENT_USER,
    "path": r"Software\Microsoft\Windows\CurrentVersion\Search",
    "name": "SearchboxTaskbarMode",
    "type": "customization"
},
"NEWS_WIDGET": {
    "hive": winreg.HKEY_CURRENT_USER,
    "path": r"Software\Microsoft\Windows\CurrentVersion\Feeds",
    "name": "ShellFeedsTaskbarViewMode",
    "type": "customization"
},
"TASKBAR_ALIGNMENT": {
    "hive": winreg.HKEY_CURRENT_USER,
    "path": r"Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced",
    "name": "TaskbarAl",
    "type": "customization"
},
"EXPLORER_OPEN_THIS_PC": {
    "hive": winreg.HKEY_CURRENT_USER,
    "path": r"Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced",
    "name": "LaunchTo",
    "type": "customization"
},

# Конфіденційність
"ALLOW_TELEMETRY": {
    "hive": winreg.HKEY_LOCAL_MACHINE,
    "path": r"SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\DataCollection",
    "name": "AllowTelemetry",
    "type": "privacy"
},
"TAILORED_EXPERIENCES": {
    "hive": winreg.HKEY_CURRENT_USER,
    "path": r"Software\Microsoft\Windows\CurrentVersion\Privacy",
    "name": "TailoredExperiencesWithDiagnosticDataEnabled",
```

```
    "type": "privacy"
},
"DISABLE_LOCATION": {
    "hive": winreg.HKEY_LOCAL_MACHINE,
    "path": r"SOFTWARE\Policies\Microsoft\Windows\LocationAndSensors",
    "name": "DisableLocation",
    "type": "privacy"
},
"DISABLE_WINDOWS_LOCATION_PROVIDER": {
    "hive": winreg.HKEY_LOCAL_MACHINE,
    "path": r"SOFTWARE\Policies\Microsoft\Windows\LocationAndSensors",
    "name": "DisableWindowsLocationProvider",
    "type": "privacy"
},
"DISABLE_ADVERTISING_ID": {
    "hive": winreg.HKEY_LOCAL_MACHINE,
    "path": r"SOFTWARE\Policies\Microsoft\Windows\AdvertisingInfo",
    "name": "DisabledByGroupPolicy",
    "type": "privacy"
},
"ENABLE_ACTIVITY_FEED": {
    "hive": winreg.HKEY_LOCAL_MACHINE,
    "path": r"SOFTWARE\Policies\Microsoft\Windows\System",
    "name": "EnableActivityFeed",
    "type": "privacy"
},
"PUBLISH_USER_ACTIVITIES": {
    "hive": winreg.HKEY_LOCAL_MACHINE,
    "path": r"SOFTWARE\Policies\Microsoft\Windows\System",
    "name": "PublishUserActivities",
    "type": "privacy"
},
"UPLOAD_USER_ACTIVITIES": {
    "hive": winreg.HKEY_LOCAL_MACHINE,
    "path": r"SOFTWARE\Policies\Microsoft\Windows\System",
```

```
        "name": "UploadUserActivities",
        "type": "privacy"
    },

    # Оптимізація
    "ALLOW_CORTANA": {
        "hive": winreg.HKEY_LOCAL_MACHINE,
        "path": r"SOFTWARE\Policies\Microsoft\Windows\Windows Search",
        "name": "AllowCortana",
        "type": "optimization"
    },
    "START_ADS": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path": r"Software\Microsoft\Windows\CurrentVersion\ContentDeliveryManager",
        "name": "SubscribedContent-338388Enabled",
        "type": "optimization"
    },
    "MENU_SHOW_DELAY": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path": r"Control Panel\Desktop",
        "name": "MenuShowDelay",
        "type": "optimization"
    },
    "ANIMATE_MIN_MAX": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\AnimateMinMax",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "COMBOBOX_ANIMATION": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\ComboBoxAnimation",
        "name": "DefaultApplied",
```

```
        "type": "optimization"
    },
    "CONTROL_ANIMATIONS": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\ControlAnimations",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "CURSOR_SHADOW": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path": r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\CursorShadow",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "DRAG_FULL_WINDOWS": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\DragFullWindows",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "DROP_SHADOW": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path": r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\DropShadow",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "AERO_PEEK": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\DWMAeroPeekEnabled",
        "name": "DefaultApplied",
        "type": "optimization"
    },
```

```
    "LISTBOX_SMOOTH_SCROLLING": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\ListBoxSmoothScrolling",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "LISTVIEW_ALPHA_SELECT": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\ListviewAlphaSelect",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "LISTVIEW_SHADOW": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path": r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\ListviewShadow",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "SELECTION_FADE": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path": r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\SelectionFade",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "TASKBAR_ANIMATIONS": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\TaskbarAnimations",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "THUMBNAILS_OR_ICON": {
        "hive": winreg.HKEY_CURRENT_USER,
```

```
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\ThumbnailsOrIcon",
        "name": "DefaultApplied",
        "type": "optimization"
    },
    "TOOLTIP_ANIMATION": {
        "hive": winreg.HKEY_CURRENT_USER,
        "path":
r"Software\Microsoft\Windows\CurrentVersion\Explorer\VisualEffects\TooltipAnimation",
        "name": "DefaultApplied",
        "type": "optimization"
    }
}
```

ДОДАТОК Г. Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАЛАШТУВАННЯ WINDOWS З ВИКОРИСТАННЯМ СИСТЕМНОГО РЕЄСТРУ

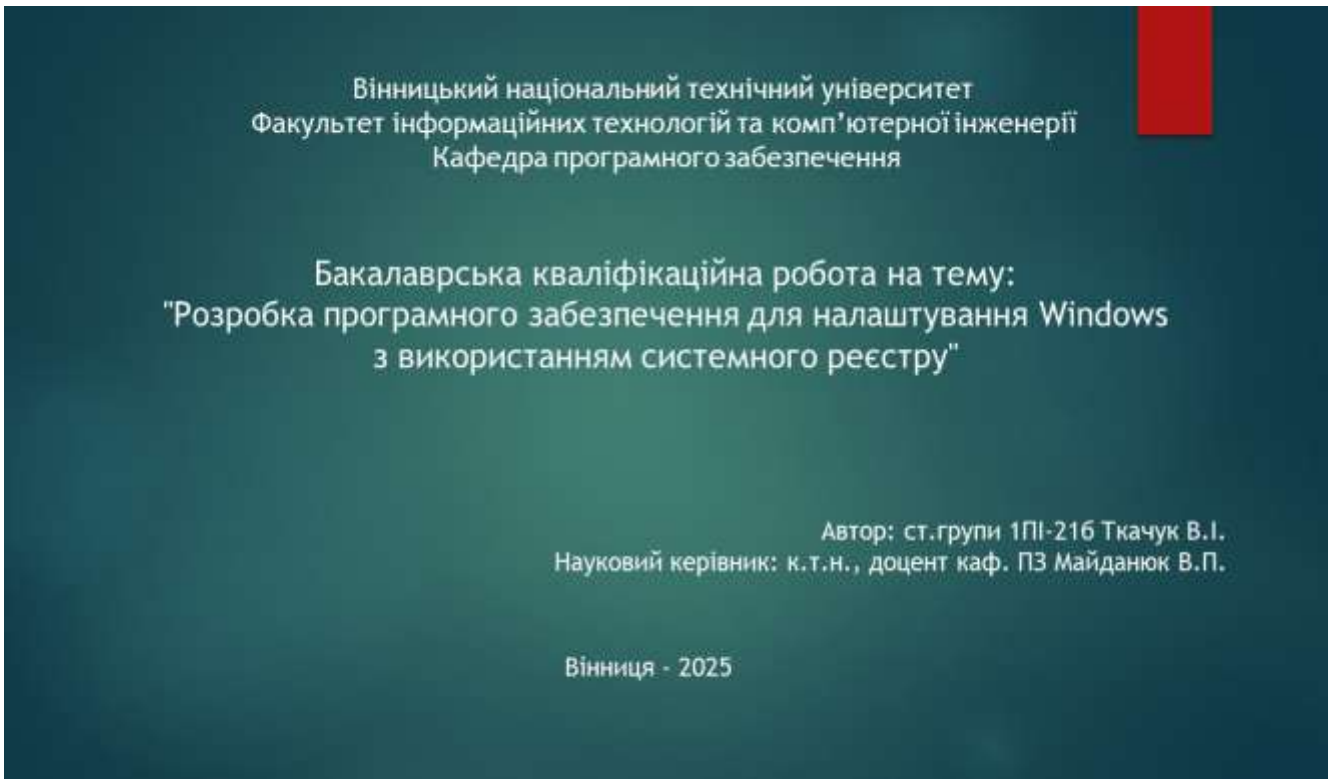

Рисунок Г.1 – Титульний аркуш

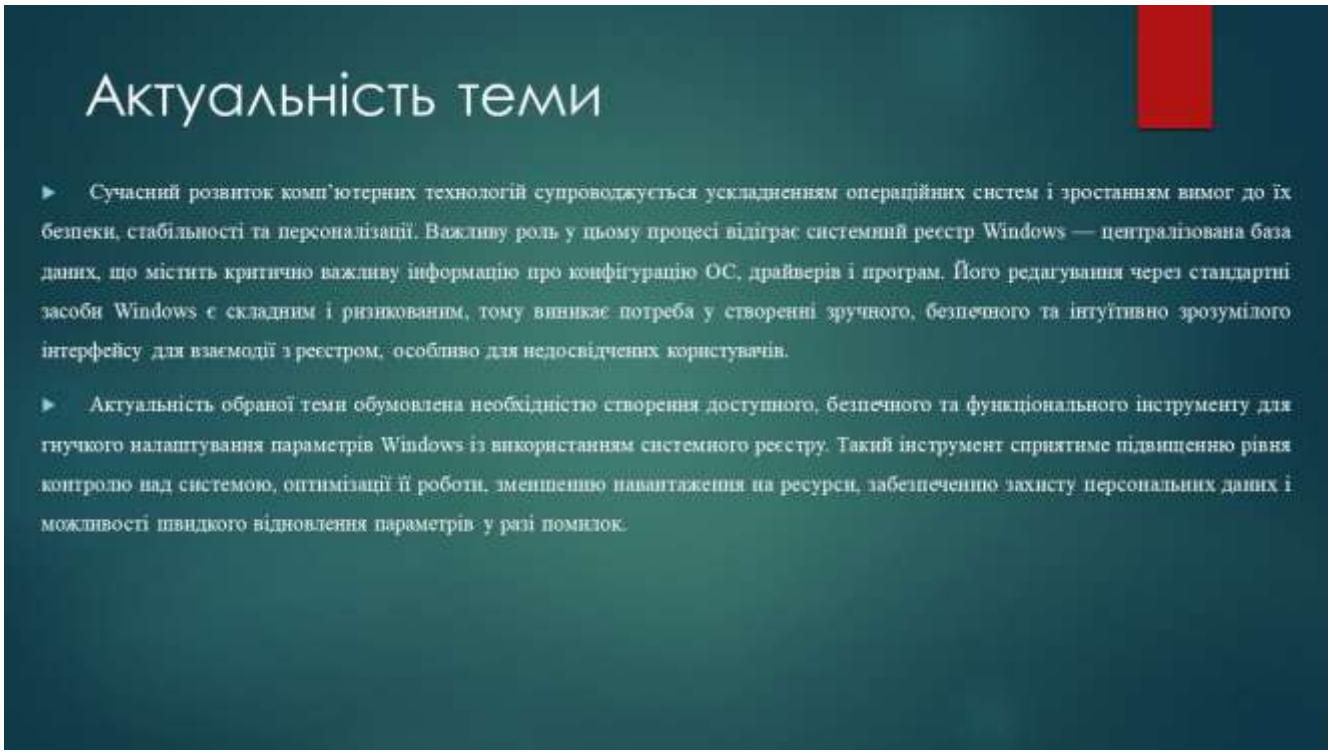

Рисунок Г.2 – Актуальність теми

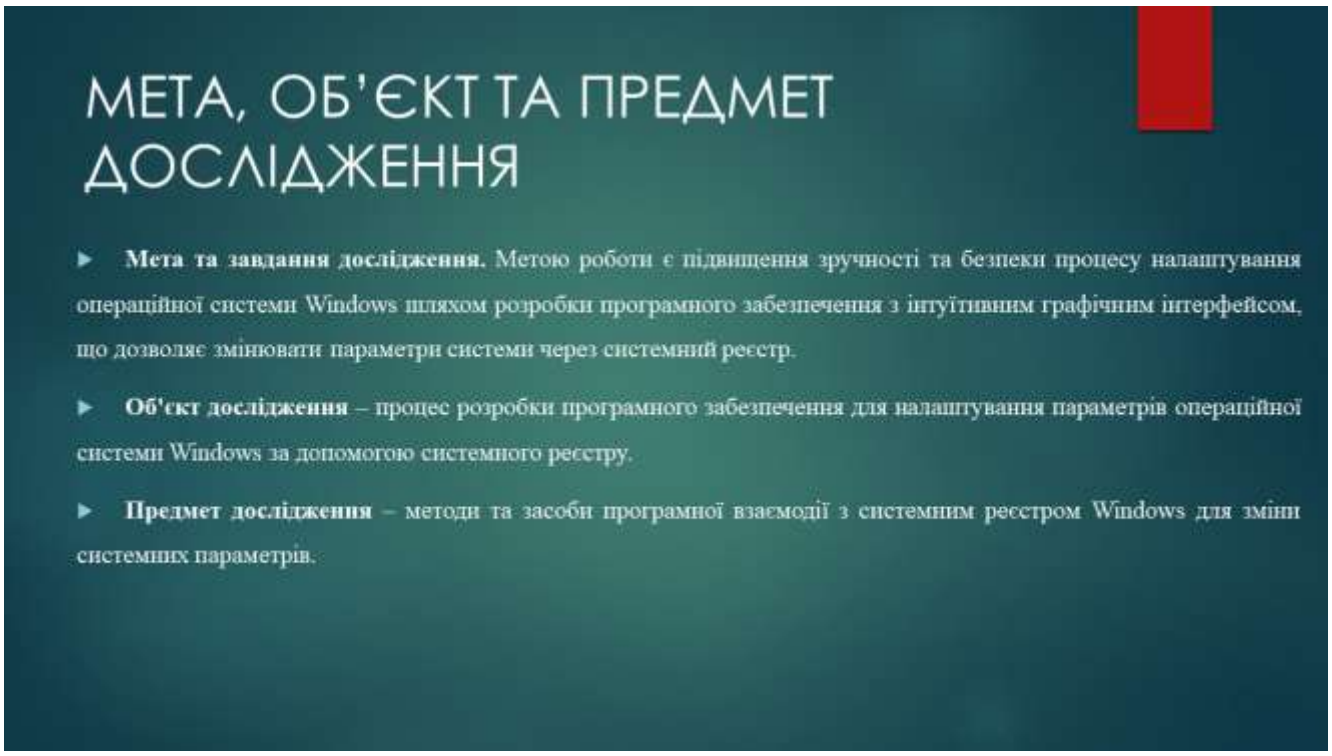

Рисунок Г.3 – Мета, об’єкт та предмет дослідження

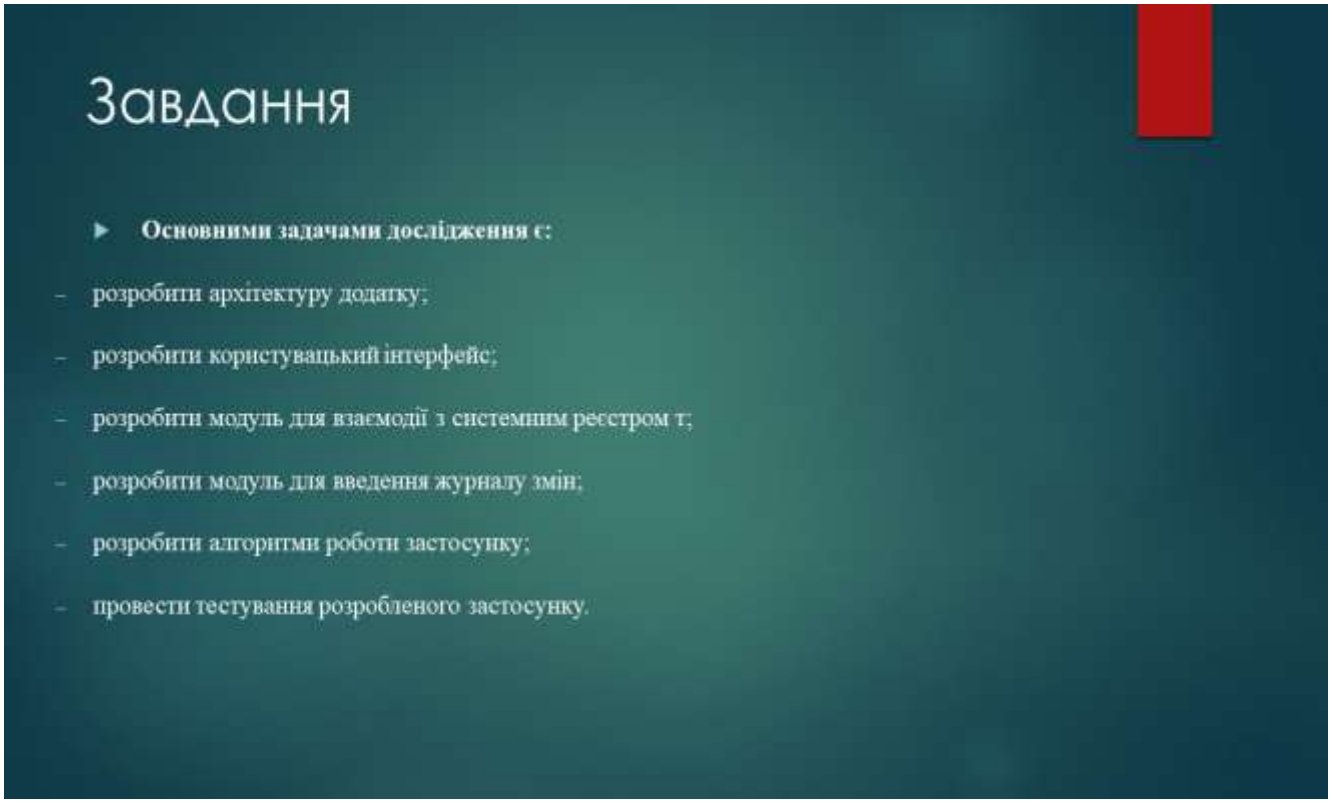

Рисунок Г.4 – Завдання

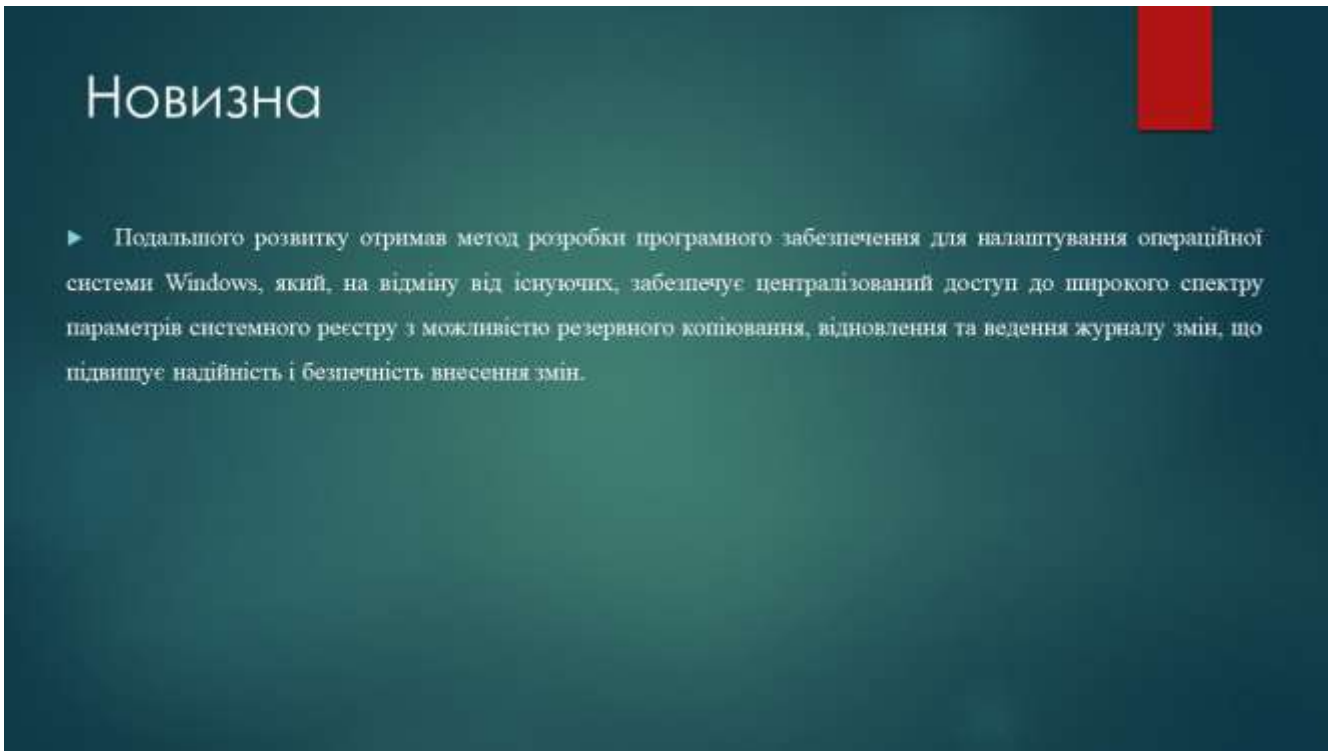

Рисунок Г.5 – Новизна

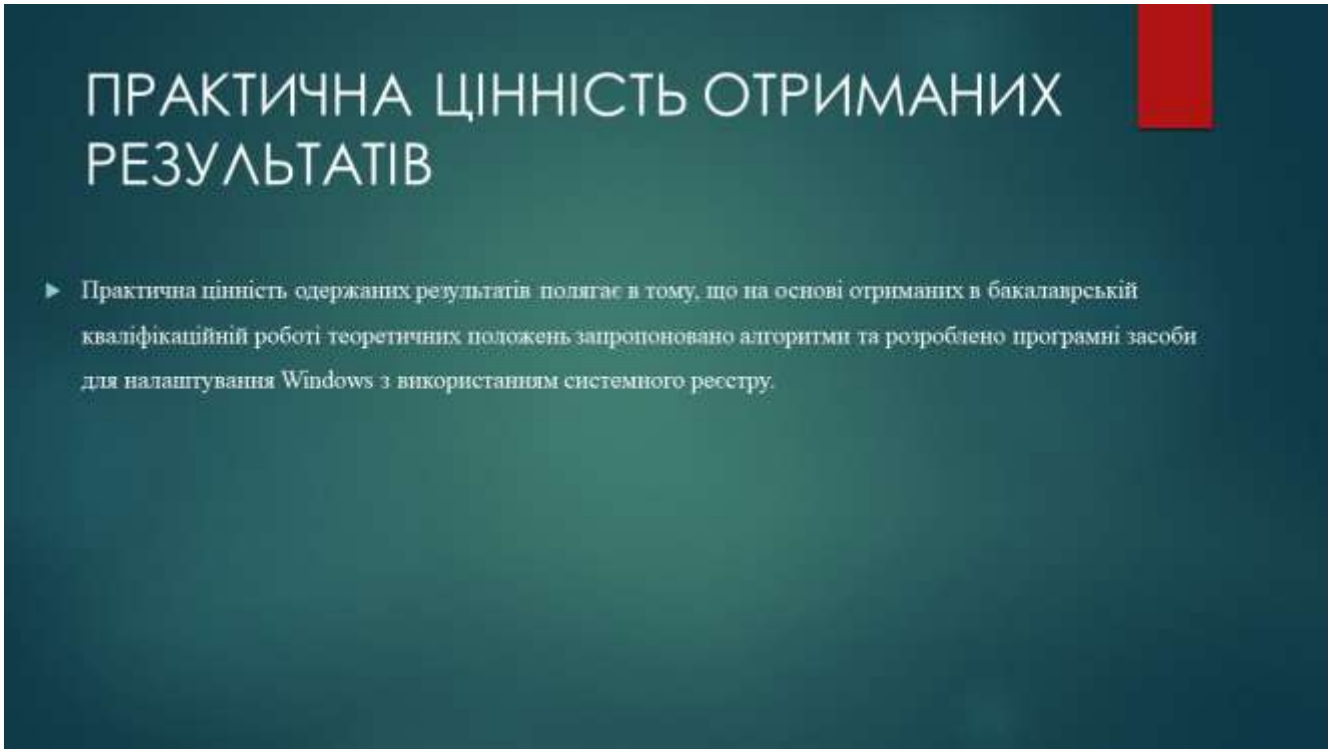

Рисунок Г.6 – Практична цінність отриманих результатів

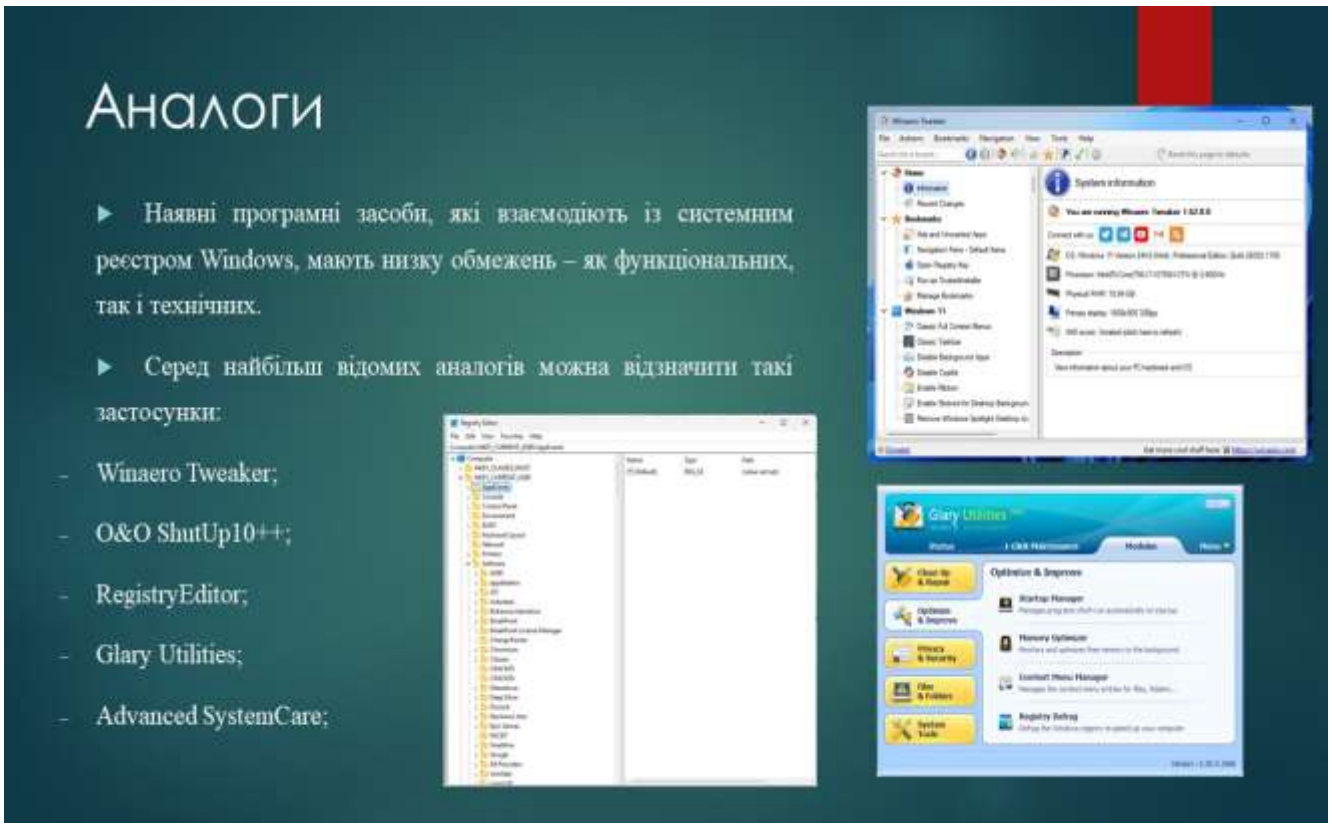

Рисунок Г.7 – Аналоги

Критерій	Winaero Tweaker	O&O ShutUp10++	Registry Editor	Glary Utilities	Advanced SystemCare	Власна розробка
Підтримка резервного копіювання	-	+	-	+	+	+
Логування змін	-	-	-	-	-	+
Пояснення до кожного параметра	+	+	-	+	+	+
Можливість зміни теми (світла, темна)	+	-	+	-	+	+
Створення резервної копії за категорією налаштувань	-	-	-	-	-	+
Загальна оцінка	40%	40%	20%	40%	60%	100%

Рисунок Г.8 – Порівняльний аналіз аналогів

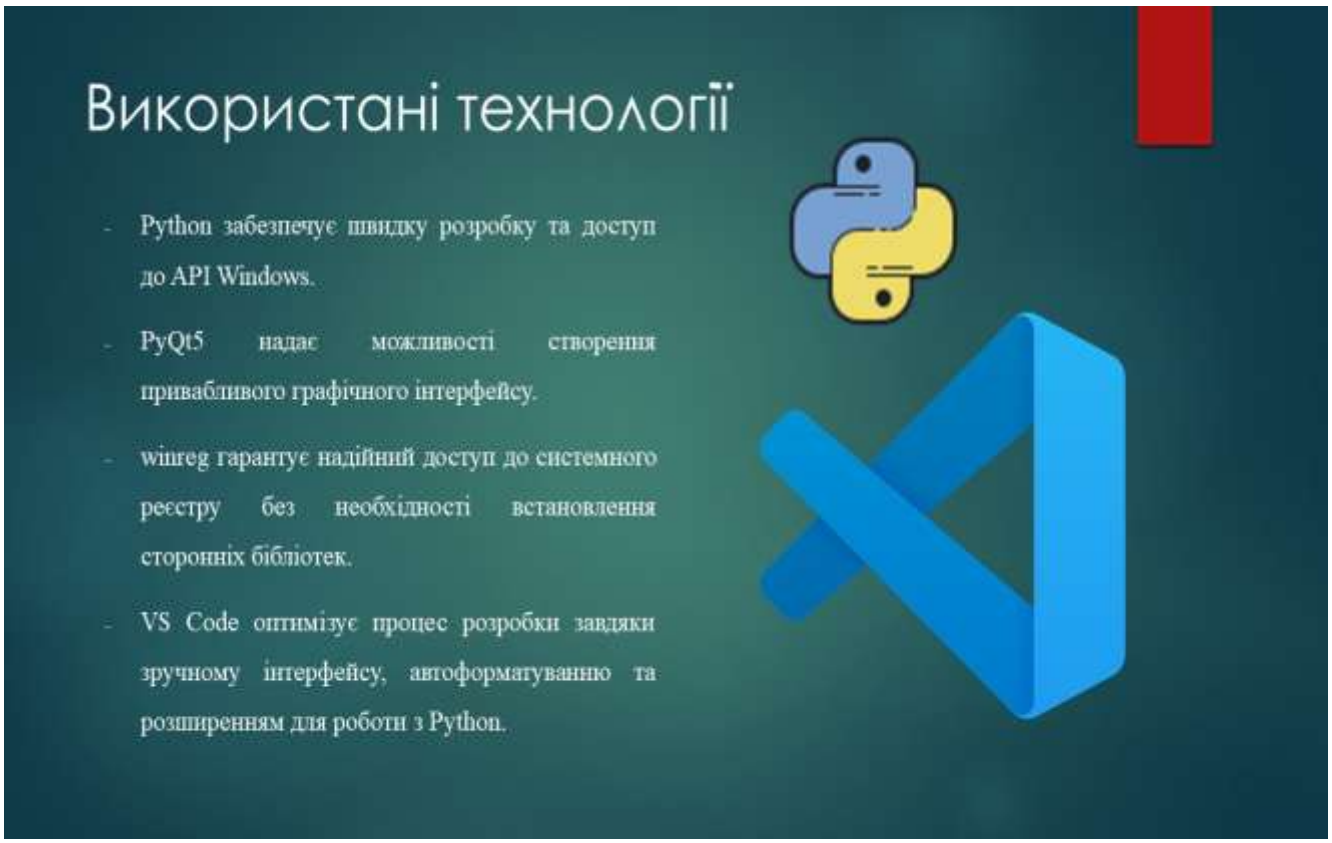

Рисунок Г.9 – Використанні технології

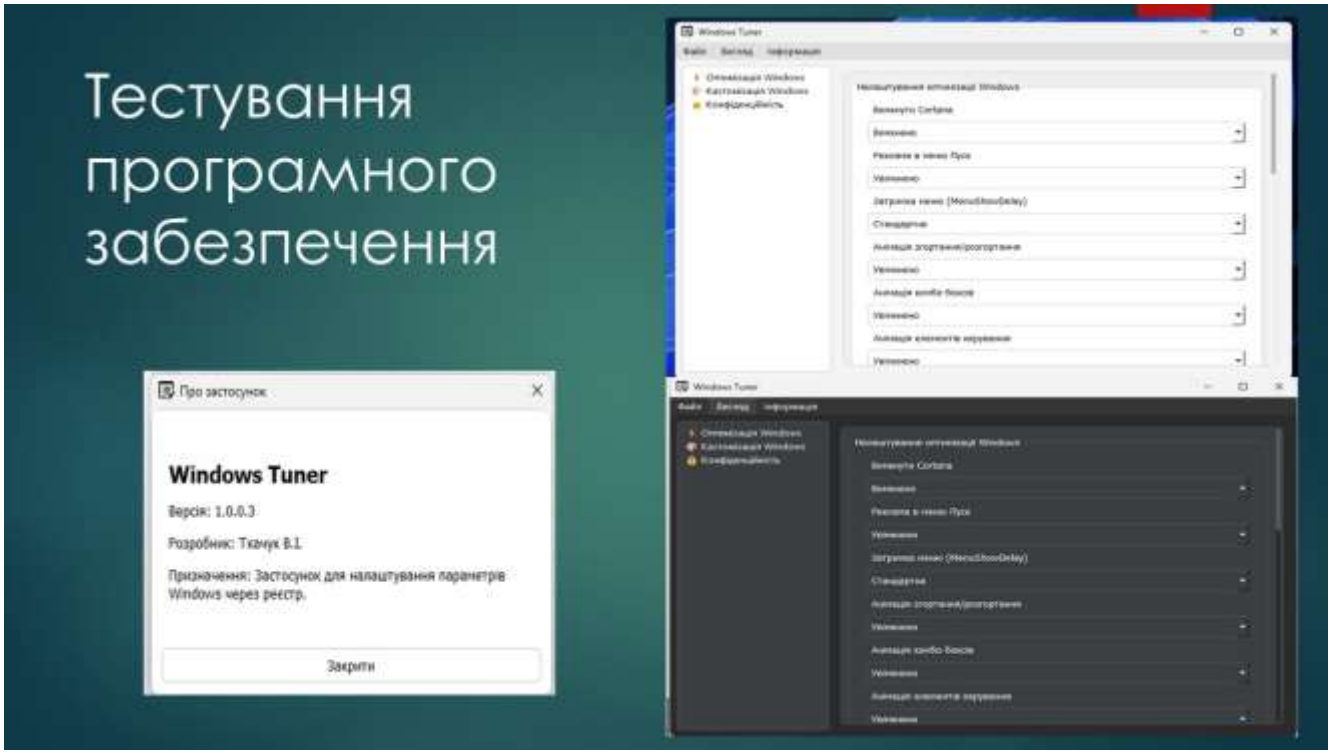

Рисунок Г.10 – Тестування програмного забезпечення

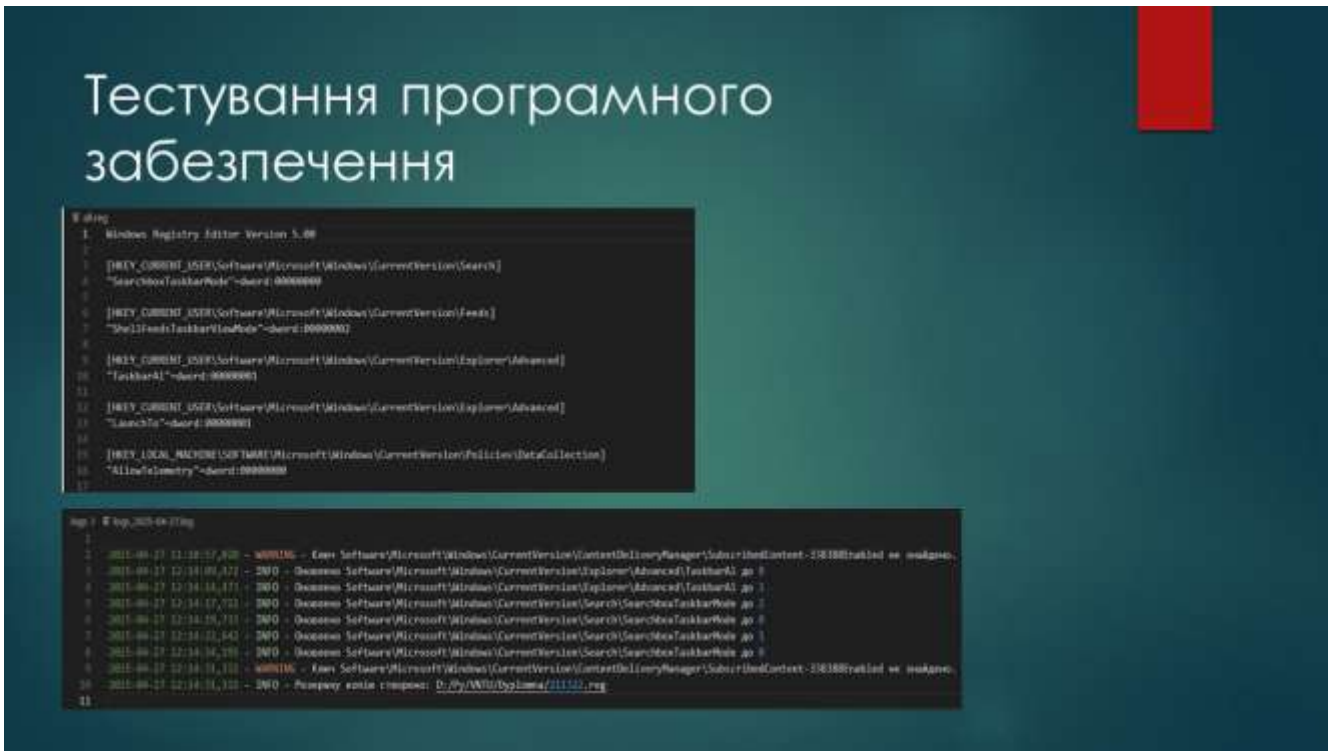

Рисунок Г.11 – Тестування програмного засобу

Рисунок Г.12 – Висновки

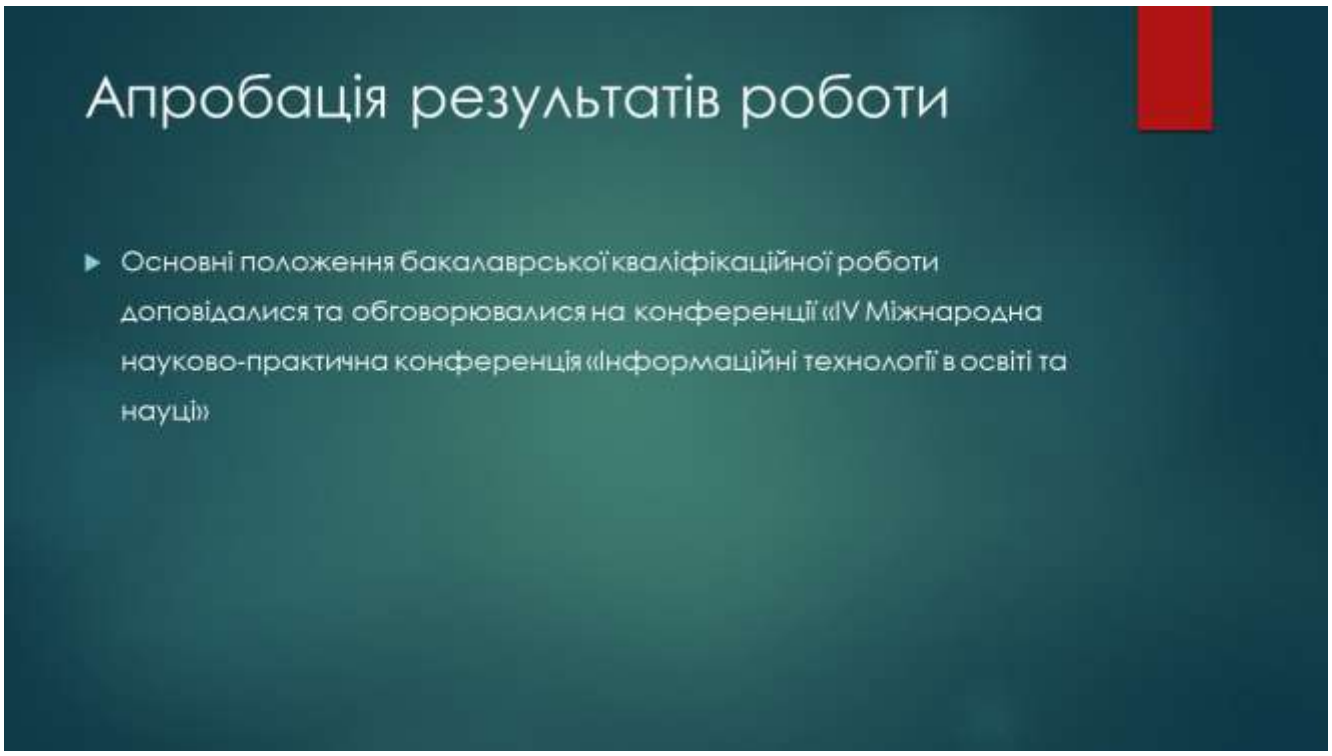

Рисунок Г.13 – Апробація

Рисунок Г.14 - Фінальний слайд