ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка застосунку мовою C# для налаштування операційної системи Windows»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Віталій КИСЮК
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Віталій КИСЮК

Керівник: Ігор ГАМАНЮК
Ст. викладач кафедри ІПЗ

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кисюку Віталію Вікторовичу

1. Тема кваліфікаційної роботи: «Розробка застосунку мовою C# для налаштування операційної системи Windows»
керівник кваліфікаційної роботи ст. викладач кафедри ІПЗ Ігор ГАМАНЮК, затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: актуальність, мова програмування C#, методи аналізу даних, вимоги до розробки застосунку, методи обробки та зберігання даних.

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

1. Аналіз предметної області для налаштування операційної системи Windows.
2. Проектування застосунку для налаштування операційної системи Windows.

3. Програмна реалізація застосунку для налаштування операційної системи Windows.
4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*
1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби та інструменти реалізації.
4. Діаграма прецедентів.
5. Діаграма класів меню.
6. Діаграма станів меню.
7. Діаграма діяльності очищення.
8. Діаграма послідовності очищення.
9. Діаграма послідовності оновлення застосунку: «Застосунок останньої версії»
10. Діаграма послідовності оновлення застосунку: «Існує новіша версія».
11. Файлова структура проєкту.
12. Екранні форми.
13. Демонстрація роботи застосунку.
14. Апробація результатів дослідження.

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд програмних засобів реалізації програмного продукту	14.03-18.03.2024	
4	Проектування застосунку	19.03-24.03.2024	
5	Програмна реалізація застосунку	25.03-25.04.2024	
6	Тестування застосунку	26.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Віталій КИСЮК
(підпис)

Керівник
кваліфікаційної роботи _____ Ігор ГАМАНЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 55 стор., 2 табл., 29 рис., 7 джерел.

Мета роботи – спростити процес налаштування операційної системи Windows шляхом впровадження застосунку для налаштування операційної системи Windows.

Об'єкт дослідження – процес налаштування операційної системи Windows.

Предмет дослідження – програмне забезпечення, яке призначене для налаштування операційної системи Windows.

Короткий зміст роботи: В роботі проаналізовано алгоритми та методи для налаштування операційної системи Windows. Проаналізовано інструментальні засоби для налаштування операційної системи Windows: CCleaner, CleanMyPC, Fortect. Розроблено алгоритм роботи застосунку та програмно реалізовані ключові функціональні можливості, зокрема: пошук та очищення не потрібних та тимчасових файлів, що залишилися після роботи програм, можливість вимкнення функцій системи, якими користувач не користується, алгоритм для поліпшення оперативної пам'яті шляхом очищення кешованного розділу, менеджер по управлінню автозавантаження програм, можливість оновлення застосунку, забезпечення перегляду системної інформації, можливість відстеження рівня навантаженості комплектуючих системи в поточному часі. Проведено функціональне тестування додатку. В роботі використано середовище розробки Visual Studio для створення застосунку, мову програмування C# для написання коду, Windows Forms для розробки графічного інтерфейсу, .NET Framework для розробки, запуску та управління застосунком, GitHub для контролю версій застосунку, Mpress для стиснення застосунку в розмірі.

Сферою використання застосунку є процес налаштування операційної системи Windows.

КЛЮЧОВІ СЛОВА: НАЛАШТУВАННЯ, ЗАСТОСУНОК, WINDOWS, VISUAL STUDIO, C#, .NET FRAMEWORK, WINDOWS FORMS, GITHUB, MPRESS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

ОС – Операційна система

UML – Unified Modeling Language

CLR – Common Language Runtime

IDE – Integrated Development Environment

ВСТУП

У сучасному світі інформаційних технологій продуктивність і стабільність роботи комп'ютерних систем є критично важливими для успішного виконання широкого спектру завдань, від повсякденної роботи до спеціалізованих професійних задач. Операційні системи, що є основою функціонування комп'ютерів, потребують постійного догляду та налаштування для забезпечення оптимальної роботи. З цією метою були розроблені застосунки для налаштування операційної системи, або так звані оптимізатори.

Застосунки для налаштування операційної системи, відомі як оптимізатори, є програмним забезпеченням, призначеним для покращення продуктивності, стабільності та безпеки комп'ютерних систем. Вони надають користувачам інструменти для управління системними ресурсами та налаштування параметрів ОС, що сприяє більш ефективному використанню обчислювальних можливостей пристрою. Такі програми є важливими помічниками для користувачів різного рівня технічної підготовки, оскільки вони автоматизують багато рутинних завдань, що в іншому випадку потребували б значних зусиль і часу.

Застосунок для налаштування операційної системи Windows — це програмне забезпечення, яке дозволяє автоматизувати процеси управління системними ресурсами, поліпшувати та налаштувати параметри операційної системи Windows. Вони допомагають користувачам зберігати високу продуктивність і стабільність роботи комп'ютера, зменшуючи потребу у ручному втручанні та технічних знаннях. Ці програми особливо важливі в умовах постійного збільшення обсягів даних і складності завдань, які виконуються на комп'ютерах.

Популярність застосунків для налаштування операційної системи Windows зумовлена їхньою здатністю значно спрощувати життя користувачів. Вони забезпечують швидке і ефективне очищення системи від непотрібних файлів, оптимізацію використання ресурсів, захист від шкідливого програмного

забезпечення та конфіденційність даних. Крім того, застосунки пропонують зручні інструменти для моніторингу стану системи і автоматичного виконання оновлень, що робить їх незамінними як для звичайних користувачів, так і для професіоналів.

Таким чином, застосунки є важливим компонентом сучасних комп'ютерних систем, сприяючи їх ефективній роботі та тривалому терміну експлуатації. Вони дозволяють користувачам концентруватися на виконанні своїх завдань, не турбуючись про технічні аспекти підтримки операційної системи.

Об'єкт дослідження – процес налаштування операційної системи Windows.

Предмет дослідження – програмне забезпечення, яке призначене для налаштування операційної системи Windows.

Мета роботи – спростити процес налаштування операційної системи Windows шляхом впровадження застосунку для налаштування операційної системи Windows.

Задачі дипломної роботи:

1. Провести аналіз інструментальних засобів для налаштування операційної системи Windows.
2. Здійснити огляд та проаналізувати існуючі застосунки для налаштування операційної системи Windows.
3. Визначити вимоги та спроектувати інтерфейс до застосунку для налаштування операційної системи Windows.
4. Розробити основні та додаткові функції до застосунку для налаштування операційної системи Windows.
5. Реалізувати застосунок для налаштування операційної системи Windows.
6. Провести тестування застосунку для налаштування операційної системи Windows.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДЛЯ НАЛАШТУВАННЯ ОПЕРАЦІЙНОЇ СИСТЕМИ WINDOWS

1.1 Постановка задачі щодо налаштування операційної системи Windows

Зараз, у світі, швидкість комп'ютерних технологій відіграють ключову роль у повсякденному житті. Однією з найпоширеніших операційних систем є Windows, яка використовується як у приватних користувачів, так і в організаціях. Вона використовується на багатьох комп'ютерах завдяки своїй зручності, широким можливостям і підтримці великої кількості програмного забезпечення.

Незважаючи на свої переваги, ОС Windows має досить складну систему налаштувань, що може бути викликом для багатьох користувачів. Від правильного налаштування ОС залежить продуктивність, безпека та загальний комфорт роботи за комп'ютером. Системні адміністратори і досвідчені користувачі зазвичай мають необхідні знання для налаштування ОС, проте більшість звичайних користувачів стикаються з труднощами під час спроби покращити свою систему.

Застосунки для налаштування ОС Windows базуються на ідеї автоматизації процесів підтримки та налаштування операційної системи, що дозволяє користувачам досягати кращої продуктивності та стабільності системи без глибоких технічних знань. Основною метою оптимізаторів є забезпечення максимальної ефективності роботи комп'ютера шляхом очищення системи від непотрібних файлів, поліпшення використання системних ресурсів та забезпечення безпеки.

Застосунки часто інтегрують у собі різноманітні функції, такі як аналіз поточного стану системи, автоматизація рутинних завдань (наприклад, очищення тимчасових файлів), підвищення продуктивності шляхом управління

автозапуском програм, моніторинг системи та оновлення драйверів. Всі ці функції спрямовані на те, щоб забезпечити стабільну та швидку роботу комп'ютера, зменшуючи час на завантаження системи та підвищуючи швидкість її роботи.

Популярність застосунків для налаштування операційної системи зростає з кількох причин:

1. Сучасні комп'ютерні системи використовуються для виконання все більш складних завдань, що потребує високої продуктивності. Ці застосунки допомагають підтримувати систему в оптимальному стані, забезпечуючи швидку та безперебійну роботу.
2. Розраховані на користувачів з різним рівнем технічної підготовки. Вони мають простий інтерфейс, що дозволяє легко налаштовувати систему навіть користувачам без глибоких знань у галузі IT.
3. Автоматизують багато рутинних завдань, що економить час користувачів і забезпечує ефективне управління системними ресурсами. Замість того, щоб вручну виконувати всі ці дії, користувачі можуть скористатися автоматизованими інструментами, що значно спрощує процес налаштування системи.
4. Мають сприяти оптимальному використанню системних ресурсів, забезпечуючи більш швидке завантаження системи, покращення швидкості роботи програм та зменшення затримок під час виконання завдань. Це особливо важливо для старіших пристроїв або систем з обмеженими ресурсами.

Застосунки для налаштування операційної системи відіграють важливу роль у забезпеченні стабільної та ефективної роботи комп'ютерних систем. Вони автоматизують багато рутинних завдань, покращують продуктивність і забезпечують безпеку, що робить їх незамінними для користувачів з різним рівнем технічної підготовки. Популярність таких програм обумовлена їхньою зручністю, ефективністю та багатофункціональністю, що дозволяє

підтримувати оптимальний стан системи без значних зусиль з боку користувача.

Більшість існуючих інструментів для налаштування операційної системи Windows, пропонують широкий спектр функцій для поліпшення і очищення системи. Однак ці програми часто мають складні інтерфейси і включають багато функцій, які можуть бути зайвими або важкими для розуміння непідготовленими користувачами. Це створює необхідність у розробці більш інтуїтивного і простого у використанні додатку, який би дозволив користувачам ефективно налаштовувати і поліпшувати свою операційнусистему Windows.

Мета даної дипломної роботи – спростити процес налаштування операційної системи Windows шляхом впровадження застосунку для налаштування операційної системи Windows, роблячи його доступним і зрозумілим для широкого кола користувачів. Основні завдання роботи включають аналіз існуючих рішень, вибір оптимальних інструментів для розробки, проектування архітектури додатку, розробку користувацького інтерфейсу і реалізацію ключових функцій програми.

У процесі роботи буде використано мову програмування C# та платформу .NET Framework, які забезпечують потужні інструменти для створення сучасних і продуктивних додатків. Завдяки своїм особливостям, C# дозволяє ефективно працювати з системними ресурсами Windows, що є ключовим аспектом при розробці застосунку для налаштування ОС.

Результатом виконання дипломної роботи стане готовий програмний продукт, яким буде легко налаштувати операційну систему Windows та поліпшити її продуктивність. Це дозволить підвищити продуктивність і безпеку комп'ютерів, а також зробить процес налаштування доступним навіть для користувачів з мінімальними технічними знаннями. Він розроблений мовою програмування C# з використанням фреймворку Windows Forms.

1.2 Огляд існуючих рішень для налаштування операційної системи Windows

Серед програмного забезпечення для налаштування операційної системи Windows необхідно розглянути CCleaner, CleanMyPC та Fortect.

1.2.1 CCleaner

CCleaner — це багатофункціональна утиліта, розроблена компанією Piriform, яка дозволяє користувачам очищувати та поліпшувати їхні комп'ютери.

Однією з ключових задач при налаштуванні та обслуговуванні операційної системи Windows є підтримка її чистоти та поліпшенні. Існує багато програмних рішень, які допомагають користувачам виконувати ці завдання. Одним з найбільш популярних і ефективних інструментів є CCleaner.

Переваги CCleaner:

- Інтуїтивно зрозумілий інтерфейс робить CCleaner доступним навіть для некваліфікованих користувачів.
- Програма швидко сканує систему та видаляє непотрібні файли, що значно підвищує продуктивність комп'ютера.
- Безкоштовна версія. CCleaner доступний у безкоштовній версії, що робить його доступним для широкого кола користувачів.
- Окрім базових функцій очищення, програма пропонує інструменти для поліпшення системи, які можуть бути корисними для більш досвідчених користувачів.

Недоліки CCleaner:

- Ризики видалення важливих файлів. При невірному використанні програми можна випадково видалити важливі системні файли, що може призвести до нестабільності системи.
- Реклама та додаткові пропозиції. У безкоштовній версії часто показується реклама, а інсталяційний процес може включати

пропозиції встановити додаткові програми, що може бути небажаним для користувачів.

– Проблеми з конфіденційністю. У минулому CCleaner стикався з проблемами безпеки, включаючи інциденти з шкідливим програмним забезпеченням у версіях для завантаження.

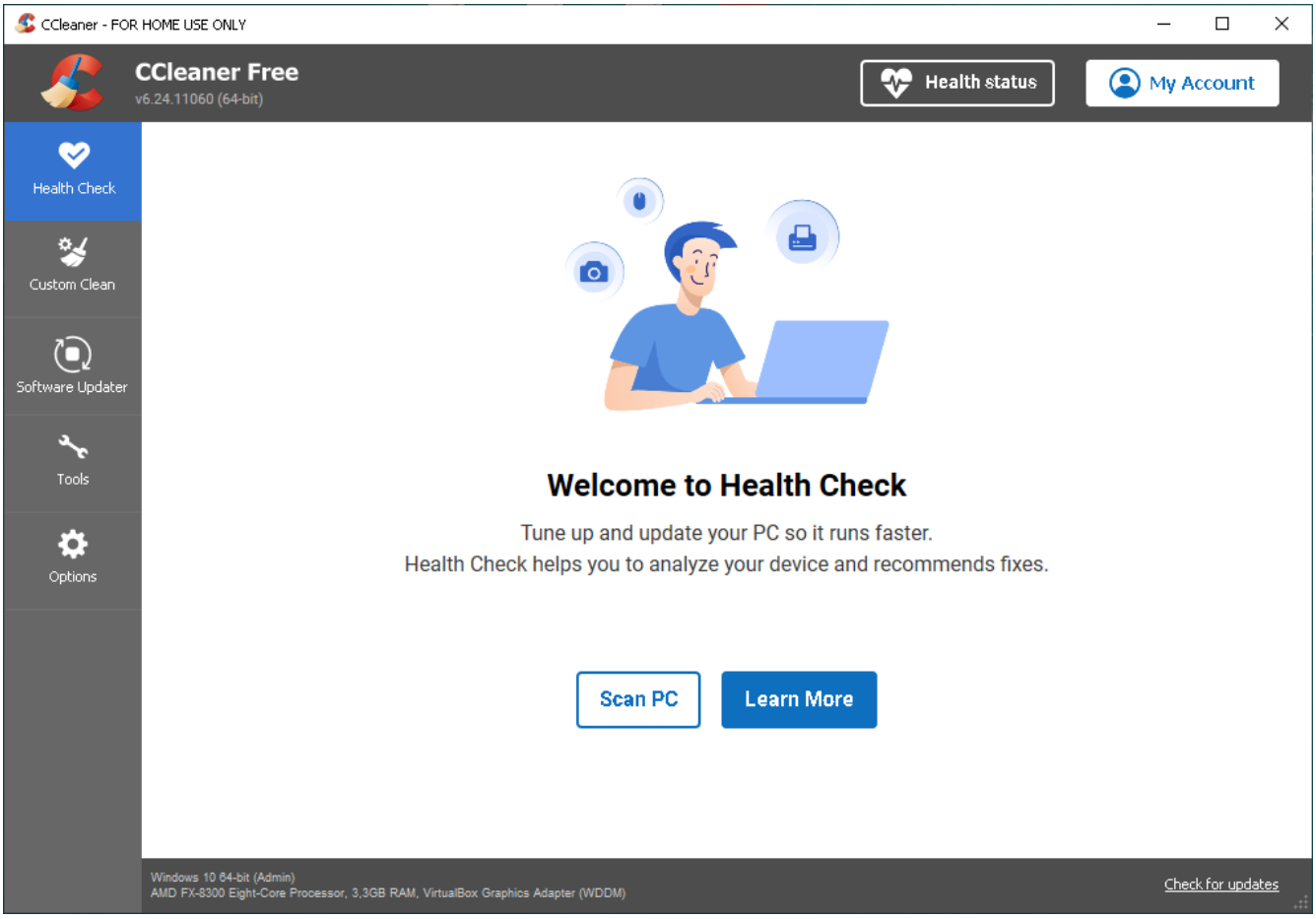


Рис. 1.1 Вікно застосунку CCleaner

1.2.2 CleanMyPC

CleanMyPC — це утиліта, яка розроблена компанією MacPaw, яка надає користувачам засоби для очищення та поліпшення їхніх персональних комп'ютерів. Для підтримки оптимальної продуктивності операційної системи Windows важливо регулярно проводити її очищення та оптимізацію. Серед численних інструментів, що пропонуються на ринку, CleanMyPC є однією з популярних програм, що забезпечує комплексне обслуговування системи.

Переваги CleanMyPC:

- Програма пропонує широкий спектр інструментів для очищення та оптимізації, забезпечуючи всебічний підхід до обслуговування ПК.
- CleanMyPC використовує безпечні алгоритми для очищення системи, що мінімізує ризик видалення важливих файлів.
- Компанія MacPaw надає підтримку користувачам, включаючи регулярні оновлення та технічну допомогу.

Недоліки CleanMyPC:

- На відміну від деяких конкурентів, CleanMyPC є платною програмою, що може бути недоліком для користувачів, які шукають безкоштовні рішення.
- Безкоштовна версія програми має обмежені функції, що може не задовольнити потреби всіх користувачів.
- Як і будь-яка інша утиліта для очищення системи, невірне використання CleanMyPC може призвести до видалення необхідних файлів або налаштувань.

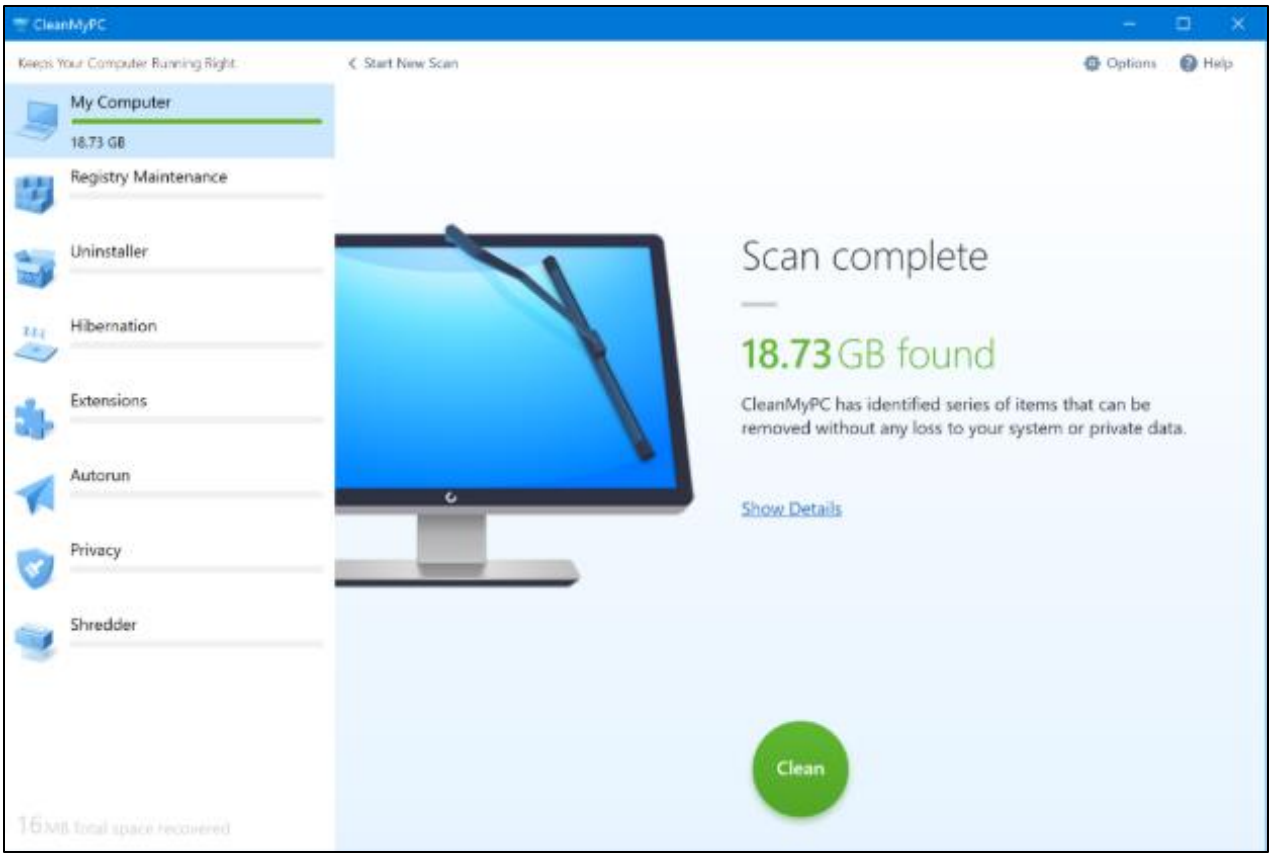


Рис. 1.2 Вікно застосунку Fortect

1.2.3 Fortect

Fortect є програмою для діагностики та ремонту системи Windows, що дозволяє виявляти та усувати різні проблеми, які можуть впливати на стабільність та продуктивність системи. Ця утиліта допомагає відновлювати працездатність системи шляхом діагностики та ремонту різних проблем.

Переваги:

- Fortect проводить ретельне сканування системи, виявляючи навіть приховані проблеми.
- Програма автоматично замінює пошкоджені файли та виправляє помилки, що забезпечує зручність використання.
- Відновлення системних файлів та виправлення помилок сприяє підвищенню стабільності та продуктивності Windows.

Недоліки:

- Вартість. Fortect є платною програмою, що може бути недоліком для користувачів, які шукають безкоштовні рішення.
- Обмеження у безкоштовній версії. Безкоштовна версія програми має обмежену функціональність, що може не задовольнити потреби всіх користувачів.
- Потреба в регулярному оновленні. Для ефективної роботи програми необхідно регулярно оновлювати базу даних та саму програму.

Рис. 1.3 Вікно застосунку Fortect

1.2.4 Таблиця порівнянь існуючих застосунків

Для визначення переваг, недоліків та ключових можливостей проведено порівняння систем CCleaner, CleanMyPC, Fortect та розроблюваного застосунку – Tweaker in 1. Результати порівняння наведено у таблиці 1.1.

Таблиця 1.1

Порівняльна таблиця

Застосунки / Характеристики	CCleaner	CleanMyPC	Fortect	Tweaker in 1
Управління автозапуском програм	+	+	-	+
Звільнення кешу оперативної пам’яті	-	-	-	+
Показ системної інформації	+	-	-	+
Відстеження використання ресурсів системи в поточному часі	-	-	-	+
Видалення кешу телеграма	-	-	-	+
Українська локалізація	+	+	-	+
Однофайловий застосунок	-	-	-	+
Запуск без необхідності файла інсталятора	+	-	-	+
Розмір	50-100 МБ	23.07МБ	281 МБ	0.2 МБ

2 ІНСТРУМЕНТИ ДЛЯ РОЗРОБКИ ЗАСТОСУНКУ

Для розробки було розглянуто інструменти як Visual Studio, .NET Framework, Windows Forms, C#, GitHub та Mpress. При розробці застосунку для налаштування операційної системи Windows використання таких інструментів значно полегшило процес розробки. Вони забезпечують інтуїтивний інтерфейс для написання коду, управління версіями, створення користувацьких інтерфейсів та поліпшенню розміру виконуваних файлів. Кожен з цих інструментів має свої унікальні переваги та надає розробнику потужні можливості для створення ефективного та продуктивного програмного забезпечення.

2.1 Інтегроване середовище розробки Microsoft Visual Studio 2022

Visual Studio – це інтегроване середовище розробки (IDE), яке створене компанією Microsoft, що є одним з найпотужніших інструментів для розробки програмного забезпечення на платформі Windows. Воно є одним із найпотужніших інструментів для розробників програмного забезпечення, який підтримує широкий спектр мов програмування і платформ.

Visual Studio було створено для полегшення розробки програмного забезпечення, надання розробникам зручного інструменту для написання, налагодження та тестування коду. З роками Visual Studio еволюціонував, додаючи нові функції та розширення, що дозволяють розробникам працювати ефективніше. Історія Visual Studio бере свій початок у 1997 році, коли Microsoft випустила першу версію цього IDE.

Visual Studio забезпечує зручний інтерфейс з підсвічуванням синтаксису, автодоповненням коду та потужними засобами налагодження. Є підтримка різноманітних мов програмування Visual Studio підтримує велику кількість мов програмування, включаючи C#, VB.NET, C++, Python, JavaScript, TypeScript, HTML, CSS та багато інших. Це робить його універсальним інструментом для

розробників, які працюють з різними технологіями. Visual Studio забезпечує потужні можливості для налагодження програм. Розробники можуть встановлювати точки зупину, переглядати значення змінних у реальному часі, відстежувати виконання коду та знаходити і виправляти помилки більш ефективно. Також є підтримка безперервної інтеграції та розгортання (CI/CD), а також можливість роботи з Git і GitHub для керування версіями коду. Велика кількість плагінів та розширень, що дозволяють додавати нові функції та можливості. Існує безліч розширень, доступних у Visual Studio Marketplace, які можуть додавати нові інструменти, підтримку додаткових мов програмування, інтерфейси для роботи з хмарними сервісами та багато іншого.

Visual Studio зіграв значну роль у розвитку програмного забезпечення. Його можливості і потужність дозволяють розробникам створювати складні додатки для різних платформ, включаючи десктопні програми, веб-додатки, мобільні додатки та хмарні сервіси. Завдяки інтеграції з іншими продуктами Microsoft і підтримці сучасних технологій, Visual Studio залишається одним з провідних інструментів для розробників по всьому світу.

Успіх Visual Studio також сприяв розвитку спільноти розробників, які діляться знаннями, розширеннями та інструментами, допомагаючи один одному створювати більш якісне програмне забезпечення.

На рис 2.1 проілюстровано вигляд вікна Visual Studio.

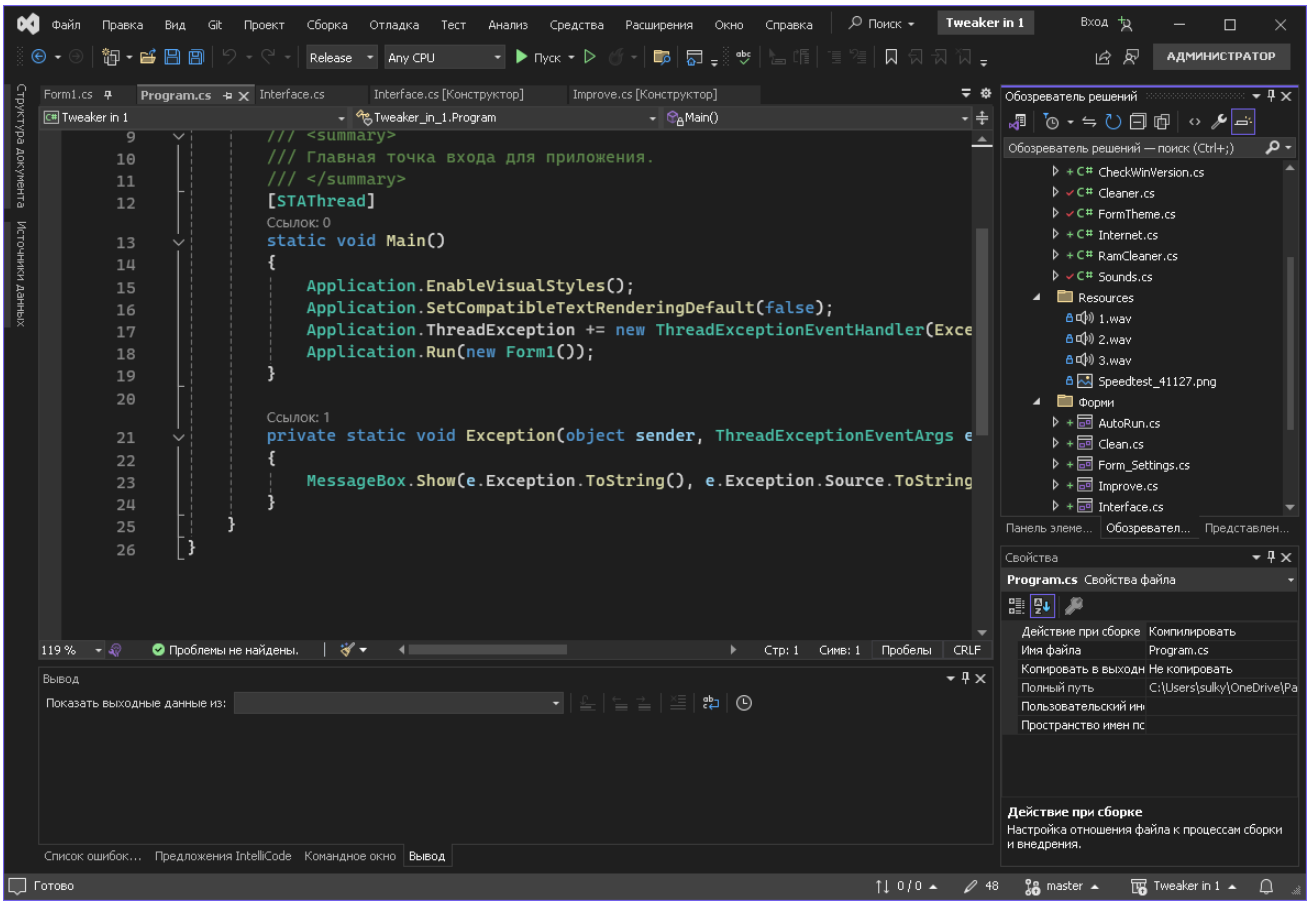


Рис. 2.1 Вікно інтегрованого середовища Visual Studio 2022

Visual Studio є надзвичайно важливим інструментом у світі розробки програмного забезпечення. Його багатофункціональність, підтримка різних мов програмування, розширюваність та інтеграція з хмарними сервісами роблять його незамінним для багатьох розробників. Завдяки Visual Studio та його легкому аналогу Visual Studio Code, розробники мають доступ до потужних інструментів, що дозволяють їм створювати, тестувати і налагоджувати програми більш ефективно та з меншими витратами часу.

2.2 Мова програмування C#

C# є потужною та гнучкою мовою програмування, яка дозволяє розробляти широкий спектр додатків, від веб-сайтів і настільних програм до мобільних додатків і ігор. Його об’єктно-орієнтований підхід, автоматичне управління пам’яттю, підтримка асинхронного програмування та сучасні мовні конструкції роблять його популярним вибором серед розробників. Інтеграція з .NET, потужні інструменти розробки і велика спільнота забезпечують додаткові переваги для програмістів, які використовують C#.

C# (вимовляється як "сі-шарп") — це об’єктно-орієнтована мова програмування, розроблена компанією Microsoft як частина своєї .NET ініціативи. Створена в 2000 році, C# поєднує в собі простоту C, потужність C++ та об’єктно-орієнтовані концепції Java. C# став однією з найпопулярніших мов програмування, використовуючи переваги .NET платформи для розробки різноманітних додатків, включаючи веб-додатки, настільні програми, мобільні додатки та ігри.

Основні особливості C#:

1. C# повністю підтримує принципи об’єктно-орієнтованого програмування, включаючи інкапсуляцію, спадкування та поліморфізм. Це дозволяє створювати модульні, повторно використовувані та масштабовані додатки. Об’єктно-орієнтований підхід полегшує підтримку та розвиток великих програмних проектів.

2. Однією з важливих особливостей C# є автоматичне управління пам’яттю завдяки механізму збирача сміття (garbage collector). Це значно зменшує кількість помилок, пов’язаних з управлінням пам’яттю, таких як витоки пам’яті та порушення доступу до пам’яті.

3. C# є мовою зі строгою типізацією, що означає, що всі змінні та об’єкти повинні мати визначені типи. Це дозволяє виявляти багато помилок на етапі компіляції, покращуючи надійність і безпеку коду.

4. Language Integrated Query (LINQ) — це потужний інструмент, інтегрований у C#, який дозволяє виконувати запити до колекцій даних прямо в коді. LINQ робить роботу з даними більш зручною та читабельною, дозволяючи використовувати SQL-подібні запити до масивів, списків та інших колекцій.

5. C# підтримує асинхронне програмування за допомогою ключових слів async та await. Це дозволяє створювати ефективні та продуктивні додатки, які можуть виконувати тривалі операції, не блокуючи основний потік виконання.

6. C# постійно розвивається, і нові версії мови включають підтримку сучасних мовних конструкцій і синтаксичних особливостей, таких як лямбда-вирази, анонімні типи, кортежі, шаблони матчинг і багато іншого. Це робить мову потужною і зручною для розробників.

Рис.2.2 Логотип мови програмування C#

2.3 .NET Framework

.NET Framework — це програмна платформа, розроблена компанією Microsoft, яка дозволяє створювати, розгортати і виконувати додатки та служби на основі XML Web Services. Вперше випущена в 2002 році, .NET Framework включає в себе велику бібліотеку класів і середовище виконання (CLR), що підтримує різні мови програмування. Ця платформа стала основою для багатьох сучасних додатків, що працюють на операційних системах Windows.

Основні компоненти .NET Framework

- CLR є основним компонентом .NET Framework, що забезпечує виконання коду. Він відповідає за управління пам’яттю, виконання

коду, обробку винятків, безпеку та інші системні послуги. CLR підтримує кілька мов програмування, що дозволяє розробникам писати код на різних мовах і використовувати його разом в одному додатку.

- FCL — це велика бібліотека класів, яка надає базовий набір функціональних можливостей для розробки додатків. Вона включає в себе тисячі класів, що охоплюють різні області, такі як робота з файлами, обробка XML, доступ до баз даних, розробка графічних інтерфейсів користувача, мережеві взаємодії і багато іншого.
- .NET Framework підтримує декілька мов програмування, включаючи C#, VB.NET, F# та C++. Це дозволяє розробникам вибирати мову, яка найбільше відповідає їхнім потребам та вподобанням. Завдяки CLR, код, написаний на різних мовах, може бути легко інтегрований і взаємодіяти в межах одного додатку.
- ASP.NET — це компонент .NET Framework, призначений для розробки веб-додатків і веб-сервісів. Він надає розробникам інструменти і бібліотеки для створення динамічних веб-сторінок, веб-аплікацій і веб-сервісів. ASP.NET підтримує як веб-формати, так і модель MVC (Model-View-Controller), що забезпечує гнучкість при розробці веб-рішень.

.NET Framework є потужною і гнучкою платформою для розробки додатків різних типів. Його багатий набір бібліотек, підтримка кількох мов програмування, висока продуктивність і безпека роблять його популярним вибором серед розробників. Незважаючи на появу нових технологій, таких як .NET Core і .NET 5/6, .NET Framework продовжує використовуватися в багатьох проектах завдяки своїй стабільності і перевіреній часом функціональності.

Для розробки десктопного застосунку мовою C# використовувався фреймворк .NET Framework (рис. 2.3), який надає розширений набір бібліотек та інструментів для створення програм на платформі Windows.

Рис. 2.3 Логотип фреймворку .NET Framework

2.4 Windows Forms

Windows Forms (WinForms) — це технологія для створення графічних інтерфейсів користувача (GUI) на платформі Windows, яка з'явилася на початку 2000-х років як частина .NET Framework від Microsoft. Вона дозволяє розробникам створювати настільні додатки з використанням бібліотек класів, які надають засоби для розробки візуальних елементів, таких як кнопки, текстові поля, списки та інші компоненти. Незважаючи на появу новіших технологій, таких як WPF (Windows Presentation Foundation) та UWP (Universal Windows Platform), Windows Forms залишається популярним вибором для багатьох розробників завдяки своїй простоті, стабільності та широкій підтримці.

Основні можливості Windows Forms:

- Однією з головних переваг Windows Forms є його простота. Інтерфейс розробки базується на візуальному редакторі, де розробники можуть перетягувати елементи управління на форму, налаштовувати їх властивості та події. Це значно спрощує процес створення інтерфейсу користувача і дозволяє зосередитися на логіці додатку.
- Windows Forms надає великий набір стандартних елементів управління, таких як кнопки, текстові поля, мітки, списки, таблиці та інші. Ці елементи можуть бути легко налаштовані і розширені відповідно до потреб додатка. Крім того, існує багато сторонніх

бібліотек, які додають додаткові компоненти і розширюють функціональність Windows Forms.

- Windows Forms є частиною .NET Framework, що дозволяє розробникам використовувати всі можливості цієї платформи, включаючи роботу з базами даних, веб-службами, обробку XML, мережеві можливості та багато іншого. Завдяки цьому розробка на Windows Forms є потужною та гнучкою.
- Windows Forms підтримує подійно-орієнтовану модель програмування, що означає, що кожен елемент управління може генерувати події, на які розробники можуть реагувати, використовуючи відповідні обробники подій. Це робить додатки інтерактивними та реактивними до дій користувача.
- Оскільки Windows Forms існує вже багато років, для цієї технології доступна велика кількість документації, прикладів коду, підручників та форумів, де розробники можуть отримати допомогу та обмінятися досвідом.

Windows Forms є одним з найпоширеніших інструментів для створення графічного інтерфейсу у .NET Framework. За допомогою Windows Forms було створено графічний інтерфейс застосунку для налаштування операційної системи Windows.

Рис. 2.4 Логотип фреймворку Windows Forms

2.5 GitHub

GitHub є потужною платформою для спільної розробки програмного забезпечення, яка забезпечує ефективне управління кодом, автоматизацію робочих процесів і співпрацю з іншими розробниками. Завдяки своїм численним функціям та інструментам, GitHub став незамінним для багатьох розробників та організацій по всьому світу. Його здатність підтримувати проекти з відкритим кодом, корпоративні рішення та освітні ініціативи робить його універсальним інструментом для будь-якого розробника.

GitHub — це платформа для спільної розробки програмного забезпечення та хостингу проектів, заснована на системі контролю версій Git. Запущена у 2008 році, яка швидко стала популярною серед розробників завдяки своїм потужним можливостям для спільної роботи, багатому набору інструментів та широкій спільноті. GitHub надає можливість зберігати та керувати репозиторіями коду, відслідковувати зміни, співпрацювати над проектами, автоматизувати робочі процеси і багато іншого.

Основні особливості GitHub:

- Репозиторії на GitHub є центральним місцем для зберігання та управління кодом. Вони містять всі файли проекту, включаючи історію змін, що дозволяє розробникам відслідковувати та відновлювати попередні версії коду. Кожен репозиторій може бути публічним або приватним, що дозволяє контролювати доступ до проекту.
- Форк (fork) — це копія репозиторію, яку користувач може створити для внесення змін або додавання нових функцій. Після внесення змін користувач може надіслати пул реквест (pull request), щоб запропонувати свої зміни основному проекту. Це дозволяє розробникам легко співпрацювати та об’єднувати свої зусилля для покращення коду.

- GitHub надає інструменти для управління завданнями і відслідковування проблем. Issues (проблеми) дозволяють розробникам повідомляти про баги, пропонувати нові функції та обговорювати зміни. Проекти (projects) — це інструмент для управління завданнями, який дозволяє створювати дошки з картками для відслідковування прогресу і організації роботи над проектом.
- GitHub Actions — це інструмент для автоматизації робочих процесів, який дозволяє створювати скрипти для автоматичного виконання завдань, таких як тестування коду, розгортання додатків або оновлення документації. Це робить розробку більш ефективною і надійною.
- GitHub Pages — це сервіс для хостингу статичних веб-сайтів прямо з репозиторію. Це дозволяє розробникам легко публікувати документацію, портфоліо, блоги та інші веб-сайти, використовуючи інструменти GitHub для управління контентом.
- GitHub підтримує активну спільноту розробників, де користувачі можуть обмінюватися досвідом, обговорювати проекти та співпрацювати над відкритим кодом. Соціальні функції, такі як обговорення, коментарі та стеження за репозиторіями, роблять платформу зручною для взаємодії та співпраці.

За допомогою API GitHub наш застосунок може періодично перевіряти наявність оновлень та сповіщати користувачів про їх наявність, тобто застосунок буде парсити на веб-сервісі GitHub версію застосунку для порівняння існуючої версії, щоб застосунок розумів чи існує новіша версія для оновлення.

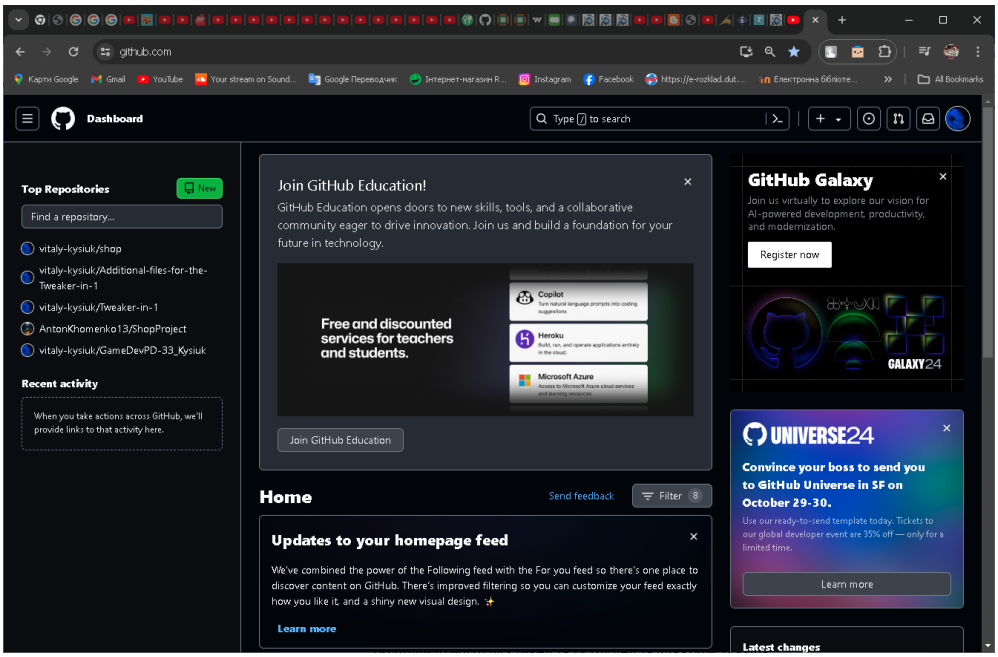


Рис. 2.5 Вікно веб-сервісу GitHub

2.6 Mpress

MPRESS є потужним і ефективним інструментом для стиснення виконуваних файлів, який забезпечує зменшення розміру додатків без втрати їх функціональності. Завдяки підтримці різних форматів файлів, простоті використання і додатковому рівню безпеки, MPRESS стає незамінним інструментом для розробників і організацій. Він допомагає оптимізувати процеси дистрибуції, зберігання і завантаження програмного забезпечення, що робить його важливим компонентом у сучасному програмному забезпеченні.

MPRESS — це інструмент для стиснення виконуваних файлів, розроблений для зменшення розміру додатків без втрати функціональності. Він використовується для оптимізації розмірів програм, що особливо важливо для дистрибуції, зберігання та завантаження файлів. MPRESS підтримує стискання файлів у форматах PE (Portable Executable), включаючи EXE, DLL, OCX, а також у форматі ELF (Executable and Linkable Format) для Unix-подібних систем.

Основні особливості MPRESS:

- MPRESS використовує складні алгоритми стиснення, які дозволяють значно зменшити розмір виконуваних файлів. Це допомагає знизити вимоги до дискового простору і швидше завантажувати програми, особливо у випадках, коли доступ до мережі обмежений.
- MPRESS підтримує стискання файлів у форматах PE та ELF, що робить його універсальним інструментом для розробників, які працюють з різними операційними системами. Це забезпечує зручність та гнучкість у використанні.
- Одна з ключових переваг MPRESS — це те, що стиснуті файли не втрачають своєї продуктивності. Інструмент забезпечує збереження вихідної функціональності та продуктивності додатків, що робить його ідеальним для використання у виробничих середовищах.
- MPRESS має простий інтерфейс і не вимагає спеціальних знань для використання. Це дозволяє швидко інтегрувати інструмент у процес розробки і стиснути виконувані файли без значних зусиль.
- MPRESS забезпечує додатковий рівень захисту для виконуваних файлів, роблячи їх важчими для аналізу та реверс-інжинірингу. Це особливо важливо для комерційних додатків, де захист інтелектуальної власності є пріоритетом.
- MPRESS підтримує стискання виконуваних файлів для різних платформ, що робить його універсальним інструментом для розробників, які працюють з Windows і Unix-подібними системами.

Для стиснення застосунку в розмірі, було застосовано інструмент Mpress.

Рис.2.6 Вікно інструмента mpress

3. ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ ЗАСТОСУНКУ

3.1 Алгоритм роботи застосунку для налаштування операційної системи Windows

Алгоритм роботи застосунку для налаштування операційної системи Windows включає кілька основних компонентів, які забезпечують ефективну взаємодію між користувачем та налаштування системою.

Користувач взаємодіє з застосунком для налаштування операційної системи Windows через графічний інтерфейс. Після запуску застосунку користувач бачить головне вікно в якому відображено такі вкладки, як:

- Вкладка "Очищення": головне меню, в якому користувачу відображаються певні пункти з вибору очищення, які він може обрати та застосувати. Вибір очищення включає в себе: пусті папки, тимчасові файли, файли які залишились після центру оновлення Windows, журнал подій, кошик, кеш телеграм, кеш браузеру, кеш оперативної пам’яті.
- Вкладка "Поліпшення": Тут користувач може активувати різні опції для покращення продуктивності та ефективності роботи системи. Ці опції можуть включати відключення телеметрії системи, віджетів, класичних програм в фоні та збільшення кешу файлової системи. Ці пункти спрямовані на поліпшення ресурсів системи.
- Вкладка "Інтерфейс": На цій вкладці користувач може налаштувати параметри візуального вигляду та інтерфейсу системи відповідно до своїх вподобань. Це може включати зміну теми, розмитість шрифту, контекстне меню та класичний переглядач фотографій.
- Вкладка "Автозапуск": Тут користувач може переглянути список програм, які автоматично запускаються при завантаженні системи, а також вимкнути або включити автозапуск для окремих програм за потреби.

- Вкладка "Налаштування": На цій вкладці користувач може налаштовувати різні параметри та налаштування програми, такі як сповіщення, звуки, автоматичне оновлення застосунку при запуску застосунку, анімації та перевірення оновлень застосунку.
- Вкладка "Системна інформація": Тут користувач може переглянути докладну інформацію про свою систему, таку як версія операційної системи, характеристики апаратного забезпечення (процесор, пам'ять, диск тощо), а також інформацію про використання ресурсів в поточному часі.

Цей процес взаємодії дозволяє користувачеві легко керувати своєю системою, поліпшувати та підтримувати її в ефективному стані.

3.2 Моделювання прецедентів застосунку

Діаграма варіантів використання є одним з основних інструментів, що використовуються в моделюванні систем за допомогою мови UML. Вона описує функціональні вимоги до системи, показуючи взаємодію користувачів (акторів) із системою через різні варіанти використання.

У контексті застосунку для налаштування операційної системи Windows, Ця діаграма описує функціональні можливості застосунку для налаштування операційної системи Windows. На ній показано як користувач може взаємодіяти з системою для виконання різних завдань з налаштувань. Прецеденти включають в себе функції очищення, вимкнення, ввімкнення, управління, перегляд та збільшення кешу файлової системи.

Акторами діаграми є користувач та API GitHub. Користувач є головним актором, який може взаємодіяти з системою для виконання різних завдань з налаштувань. API GitHub виступає в ролі веб-сервера для контрою версій застосунку, де зберігаються застосунки новішої версії. Коли користувач перевіряє оновлення застосунку, API GitHub обробляє запит та надає інформацію користувачу про те чи існує новіша версія застосунку.

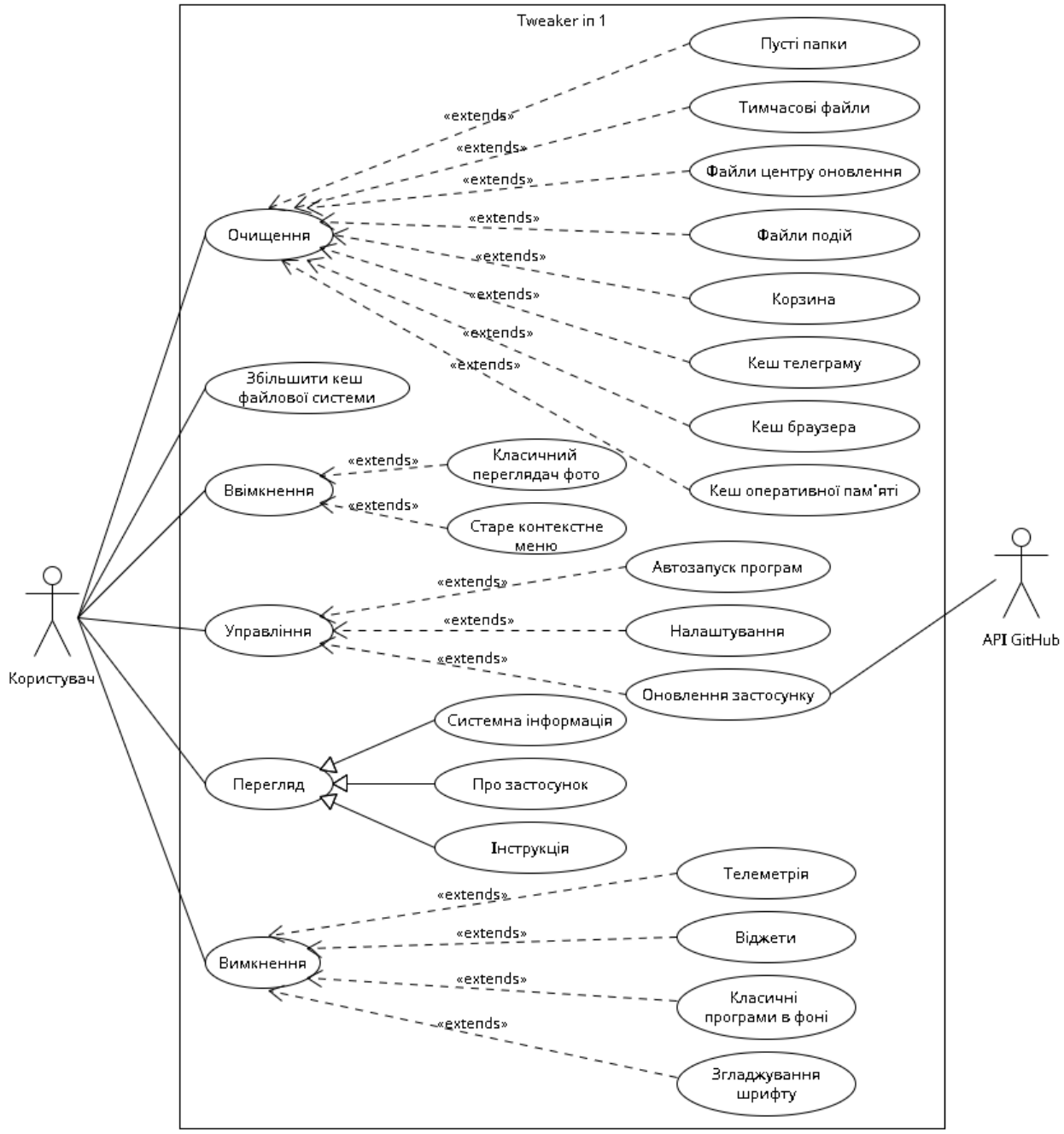


Рис. 3.1 Діаграма прецедентів

3.3 Моделювання класів меню

На етапі проєктування діаграма класів використовується для розробки структури системи. Вона допомагає визначити, які класи будуть необхідні, які атрибути та методи вони матимуть, а також які зв'язки будуть між ними. Це

важлива частина процесу розробки, оскільки вона дозволяє побачити загальну картину системи.

На діаграмі класів меню загальний вигляд всіх класів меню що присутні в застосунку та показано що об'єкт MainForm має посилання на об'єкти відповідних класів і може викликати їх методи, а саме:

- «MainForm» - основний клас меню, що включає всі вкладки: Очищення, Поліпшення, Інтерфейс, Автозапуск, Налаштування, Системна інформація.
- «Clean» - клас, що представляє вкладку "Очищення". Він містить методи для видалення різних типів файлів та кешу. «Improve» - клас, що представляє вкладку "Поліпшення". Містить методи для які пов’язані з поліпшенням системи.
- «Interface» - клас, що представляє вкладку "Інтерфейс". Містить методи для налаштування теми інтерфейсу, встановлення класичного переглядача фотографій, старого контекстного меню та згладжування шрифтів.
- «AutoRun» - клас, що представляє вкладку "Автозапуск". Містить методи для додавання та видалення програм з автозапуску.
- «Settings» - клас, що представляє вкладку "Налаштування". Містить методи для перевірки оновлень, завантаження нової версії, налаштування сповіщень, звуків, автоматичного оновлення та анімацій.
- «SystemInfo» - клас, що представляє вкладку "Системна інформація". Містить метод для відображення системної інформації.

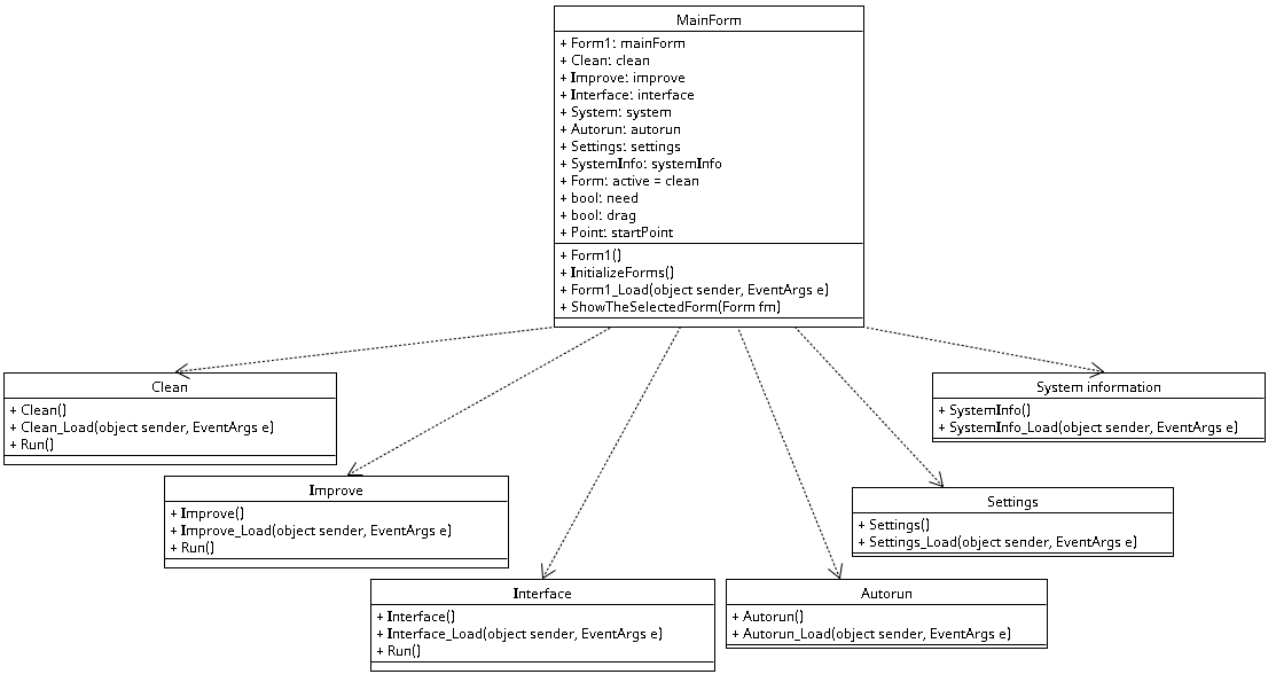


Рис. 3.2 Діаграма класів меню

3.4 Моделювання станів меню

Діаграма станів є важливим інструментом в моделюванні поведінки системи, відображаючи різні стани, в яких може перебувати система, та переходи між цими станами на основі подій або дій користувача. В контексті меню програми, діаграма станів описує можливі стани головного меню та його вкладок, а також взаємодію користувача з цими елементами.

На рис. 3.3 відображено поведінку головного меню та його вкладок у програмі. Початковий стан представляє головне меню, з якого користувач може перейти до будь-якої з вкладок: Очищення, Поліпшення, Інтерфейс, Автозапуск, Налаштування або Системна інформація. Користувач має можливість змінювати налаштування системи і застосовувати зміни. З будь-якої вкладки можна вийти з програми, що переводить систему у кінцевий стан.

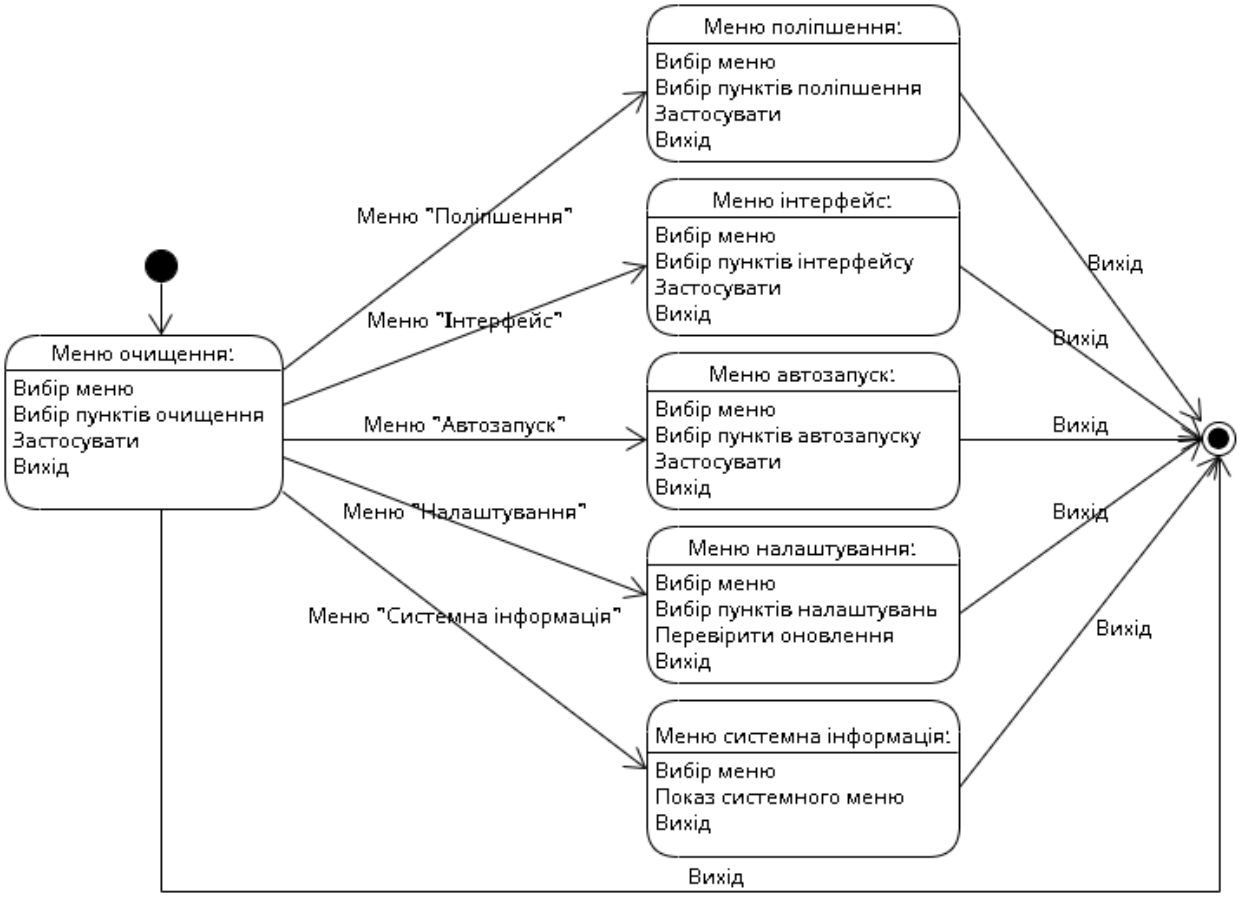


Рис. 3.3 Діаграма станів меню

3.5 Проєктування діяльності очищення

Діаграма діяльності відображає послідовність дій або робочий процес певної системи чи процесу.

На рис 3.4 представлено процес взаємодії користувача із застосунком у вкладці "Очищення", коли користувачу надається вибір певних пунктів очищення, які він може обрати та застосувати.

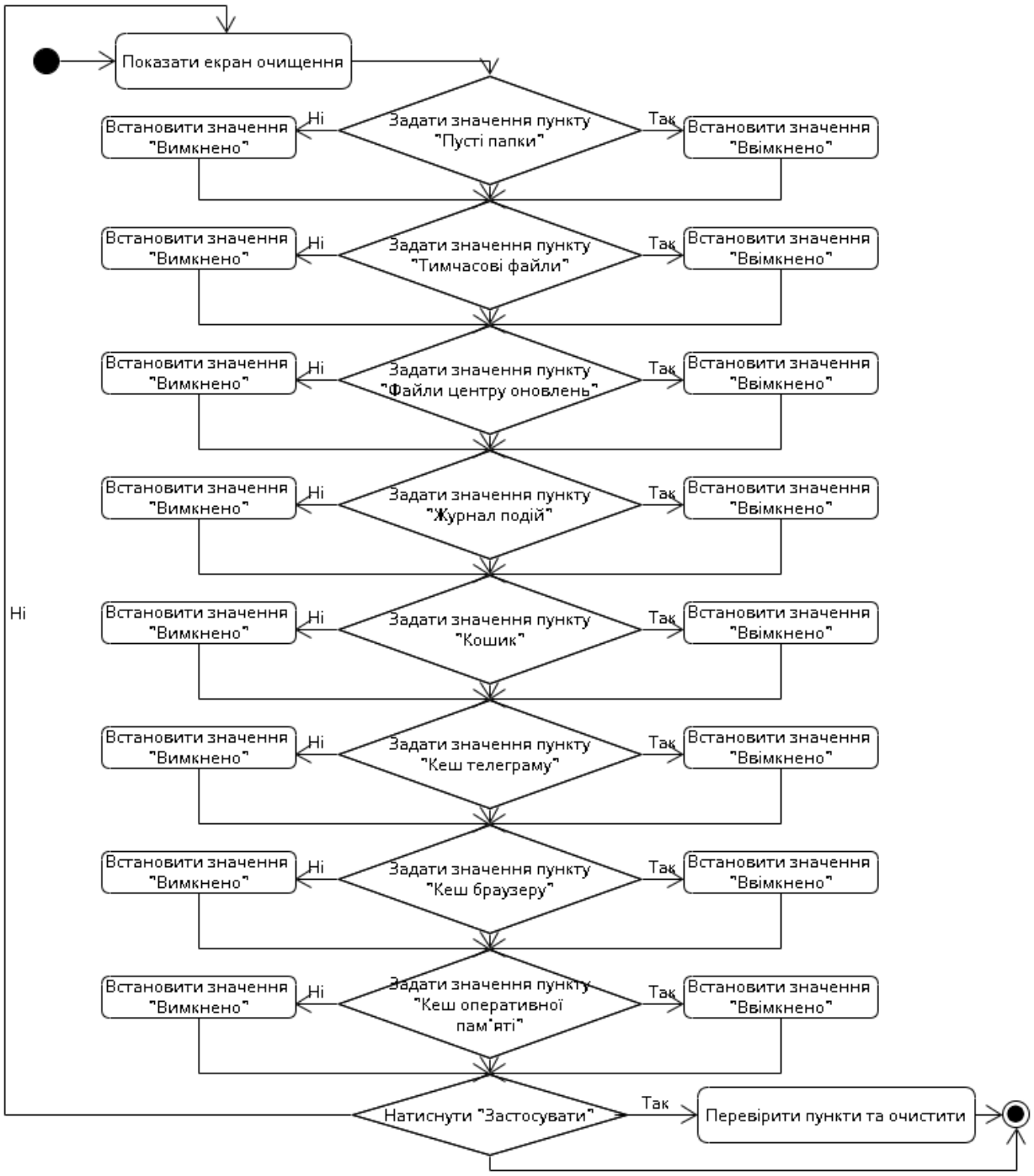


Рис. 3.4 Діаграма діяльності очищення

3.6 Проєктування взаємодії об’єктів очищення

Діаграма послідовності є одним з основних інструментів візуалізації, що використовується для моделювання взаємодій між об'єктами в системі. Вона демонструє, як об'єкти взаємодіють між собою, вказуючи на порядок і

черговість повідомлень або викликів методів. Такі діаграми часто використовуються в процесі розробки програмного забезпечення для аналізу та проектування систем, забезпечуючи чітке уявлення про поведінку системи під час виконання певного сценарію.

На діаграмі 3.5 описується що при виклиці кнопки "Застосування", викликається метод Run(), в якому перевіряється кожен елемент checkbox, якщо він True, то виконується певне очищення файлів і повертається значення size ,яке виводить повідомлення про те, скільки було очищено файлів в системі. Очищення файлів включає в себе пункти вибору з видалення пустих папок, тимчасових файлів, файлів які залишились від центру оновлень Windows, файлів журналу подій, кошику, кешу телеграма, браузера та оперативної пам’яті.

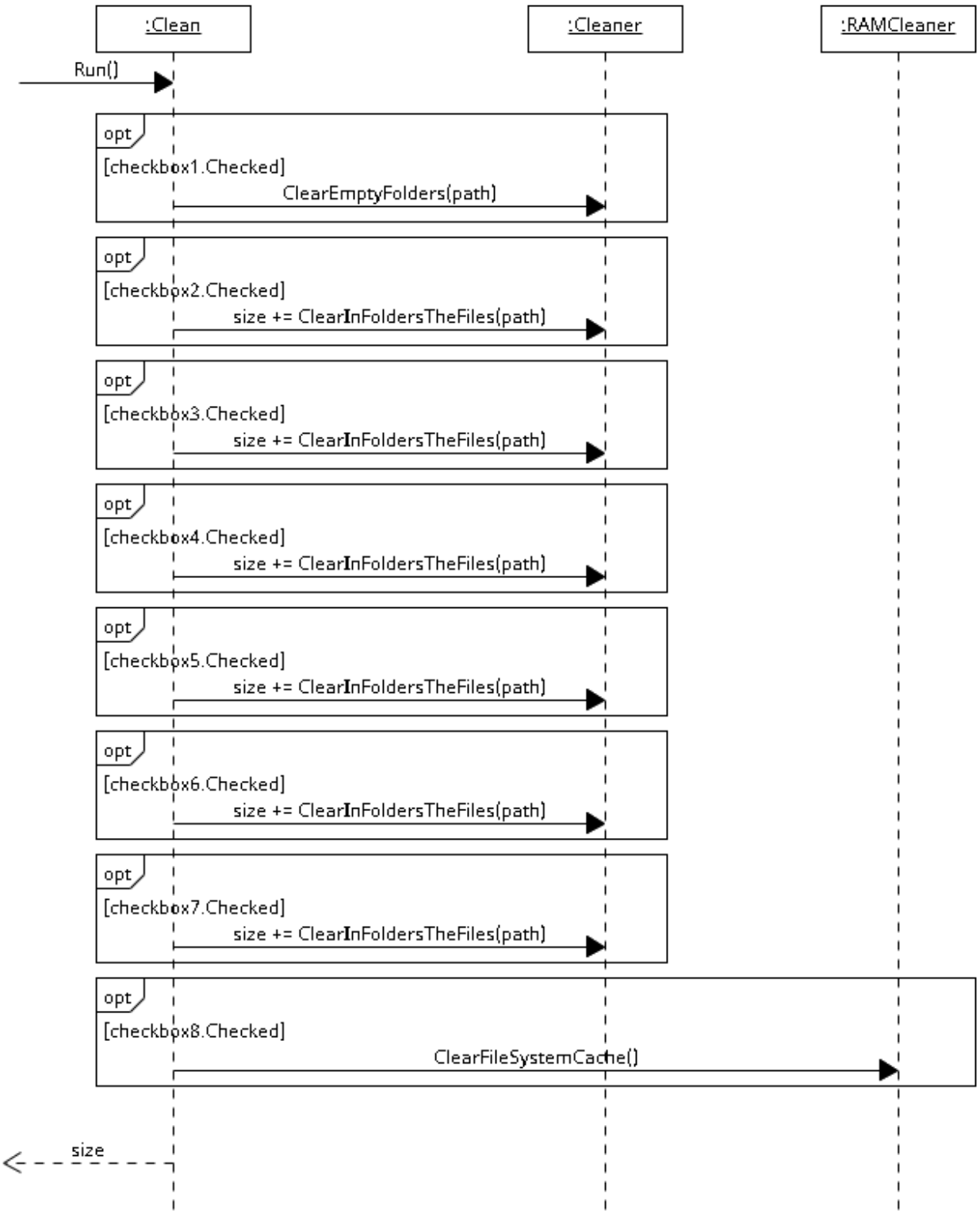


Рис. 3.5 Діаграма послідовності очищення

3.7 Проєктування взаємодії користувача і об’єктів для оновлення застосунку

Діаграма послідовності є чудовим інструментом для візуалізації процесів взаємодії між об'єктами в системі. Вона показує порядок викликів методів і обмін повідомленнями між об'єктами.

У сучасному програмному забезпеченні важливо забезпечити своєчасне оновлення застосунків для підвищення їх функціональності, безпеки та продуктивності. Ці дві діаграми описують процес перевірки наявності оновлень для застосунку, використовуючи GitHub API, та подальші дії залежно від результатів перевірки. Основну увагу приділено двом сценаріям, коли доступна новіша версія застосунку та коли застосунок вже є актуальним, для більш кращого розуміння взаємодії користувача з застосунком та хмарним сховищем GitHub.

На першій діаграмі описується процес, за яким користувач перевіряє наявність оновлень, і виявляється, що доступна новіша версія застосунку. Застосунок завантажує нову версію та повідомляє користувача про успішне завантаження. На початку користувач відкриває вкладку "Налаштування" та натискає кнопку "Перевірити оновлення". Застосунок надсилає запит до GitHub API для отримання інформації про останню доступну версію та відповідає з даними про останню версію. Він порівнює отриману версію з поточною встановленою версією. Якщо доступна новіша версія, застосунок надсилає запит до GitHub API для завантаження нової версії. Застосунок отримує нову версію, завантажує її та повідомляє користувача про успішне завантаження нової версії (рис. 3.6).

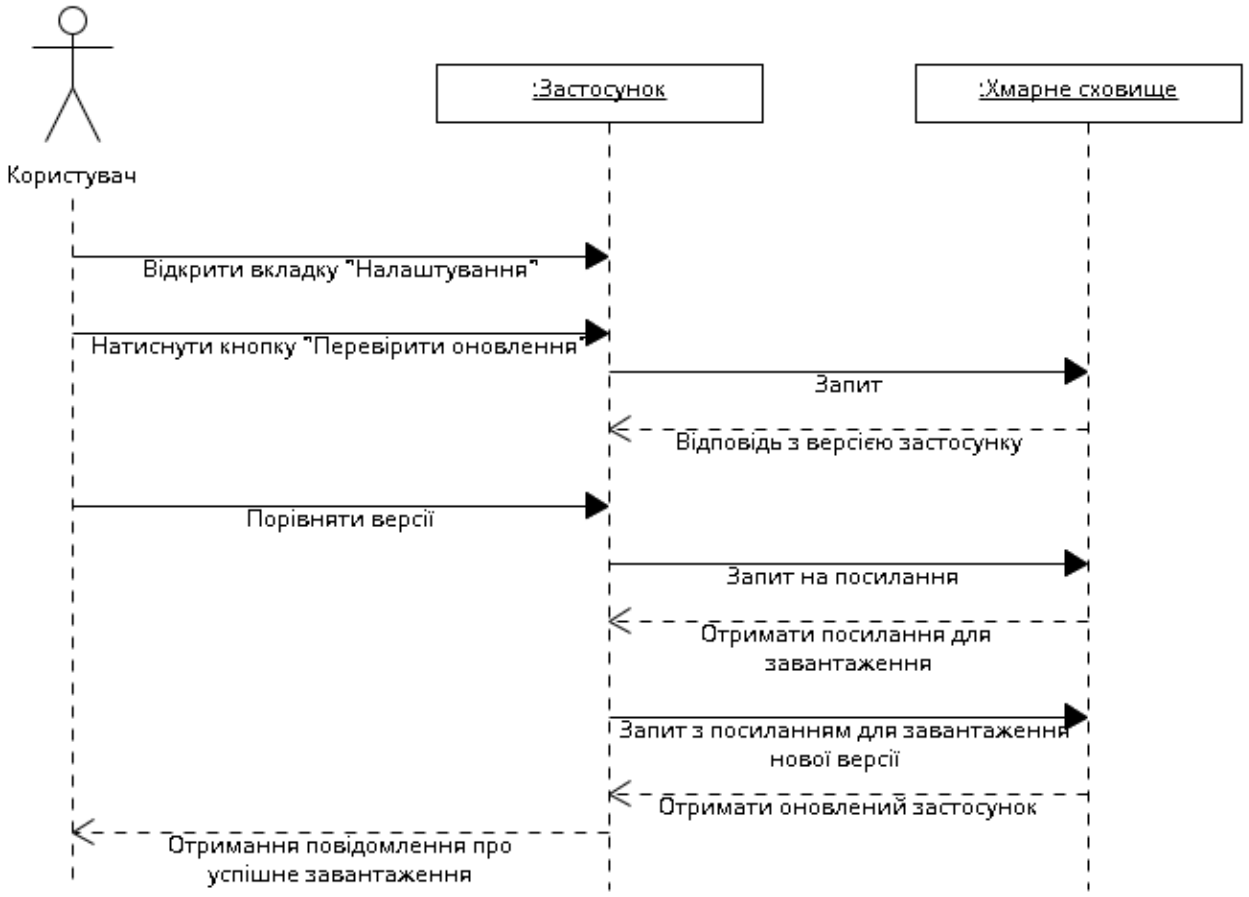


Рис. 3.6 Діаграма послідовності оновлення застосунку

На другій діаграмі описується перевірка оновлення застосунку, коли він актуальний. Ця діаграма ілюструє процес, за яким користувач взаємодіє із застосунком для перевірки наявності оновлень. У випадку, коли встановлена версія є найновішою, застосунок повідомляє користувача про відсутність необхідності оновлення. На початку користувач відкриває вкладку "Налаштування" та натискає кнопку "Перевірити оновлення". Застосунок надсилає запит до GitHub API для отримання інформації про останню доступну версію та відповідає з даними про останню версію. Застосунок порівнює отриману версію з поточною встановленою версією. Якщо версії співпадають, застосунок виводить повідомлення, що застосунок є актуальним (рис.3.7).

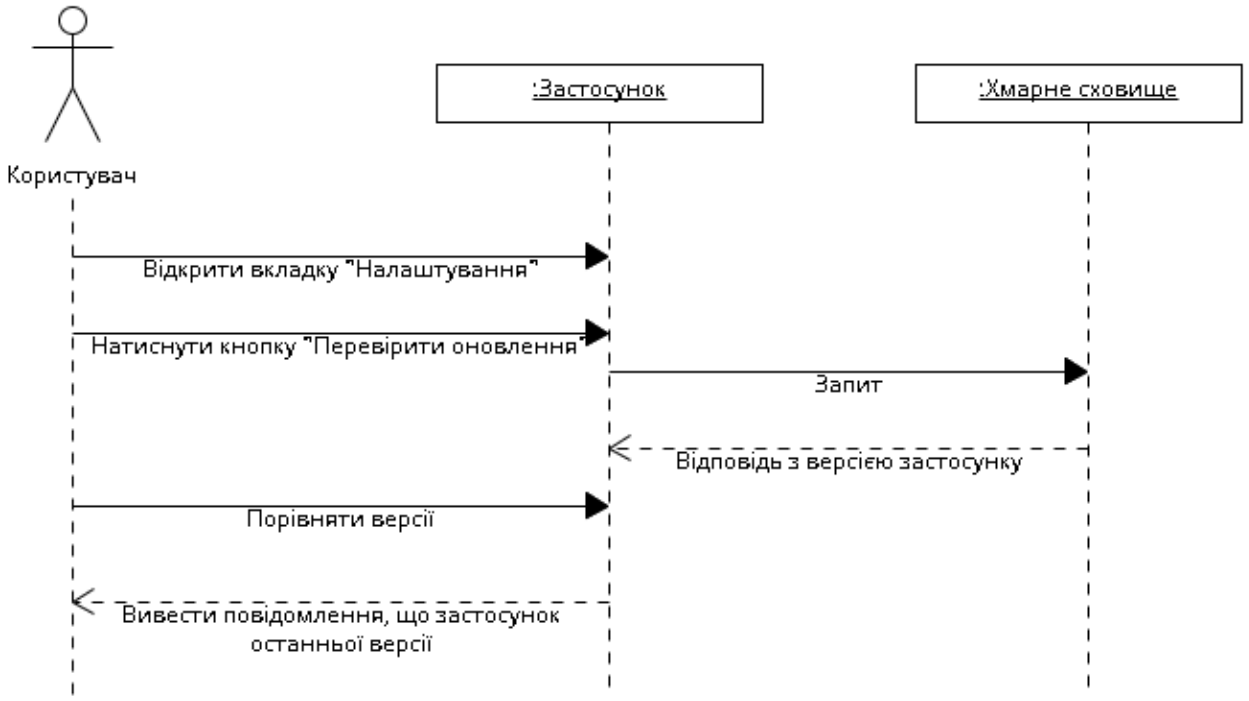


Рис. 3.7 Діаграма послідовності оновлення застосунку

3.8 Організація та управління проєктом застосунку

Файлова структура проєкту в Visual Studio більше відноситься до організації та управління проєктом. Вона допомагає розробникам впорядковувати вихідні файли, ресурси, бібліотеки та інші компоненти проєкту.

Організація та управління проєктом включає впорядкування файлів і ресурсів для зручності розробки, тестування та розгортання. Це дозволяє ефективніше працювати з кодом та ресурсами проєкту.

Файлова структура в Visual Studio спрямована на те, щоб полегшити навігацію, спільну роботу та підтримку проєкту, тому її можна розглядати як частину загального процесу організації проєкту.

На рис 3.8 зображена файлова структура проєкту, яка відображає ієрархію файлів і папок проєкту, включаючи вихідний код, класи, ресурси, бібліотеки, та конфігураційні файли.

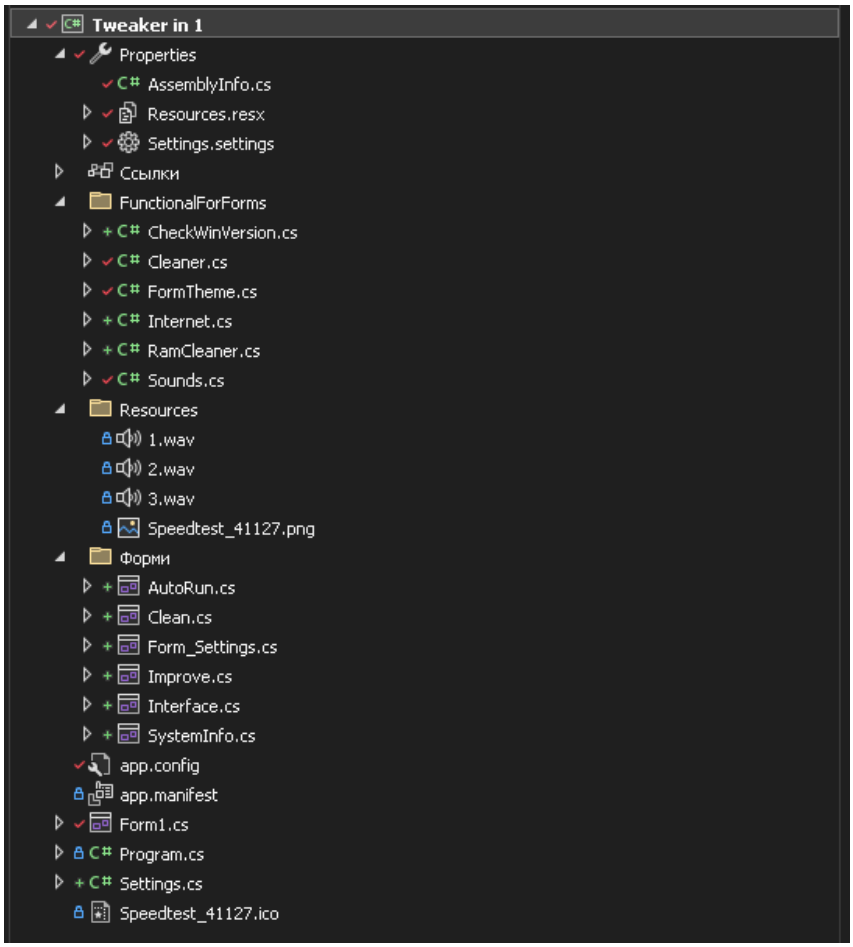


Рис. 3.8 Файлова структура проєкту

4 РОЗРОБКА ЗАСТОСУНКУ

У даному розділі описано програмну реалізацію застосунку, розробленого на мові програмування C# з використанням технології Windows Forms (WinForms). Застосунок призначений для покращення роботи операційної системи Windows, надаючи користувачеві різні функціональні можливості через інтерфейс з кількома вкладками. Основні вкладки включають "Очищення", "Поліпшення", "Інтерфейс", "Автозапуск", "Налаштування" та "Системна інформація". У вкладці "Налаштування" реалізовано функцію перевірки оновлення застосунку за допомогою GitHub API.

4.1 Архітектура застосунку

Застосунок складається головного вікна (MainForm), яка містить вкладки для навігації між різними функціональними можливостями застосунку, а саме:

- Очищення (Clean): забезпечує функції для очищення системи від непотрібних файлів.
- Поліпшення (Improve): надає інструменти для поліпшення продуктивності системи.
- Інтерфейс (Interface): дозволяє налаштовувати вигляд і поведінку інтерфейсу системи.
- Автозапуск (AutoRun): дозволяє керувати програмами, що запускаються при старті системи.
- Налаштування (SettingsForm): містить параметри конфігурації застосунку та функцію перевірки оновлень.
- Системна інформація (SystemInfo): надає інформацію про апаратне та програмне забезпечення системи та можливість відстеження використання ресурсів системи у поточному часі.

4.2 Реалізація функціональних можливостей

Вкладка «Очищення» містить пункти вибору для видалення тимчасових файлів, очищення кошика, видалення кешу браузера та інших непотрібних файлів, що накопичуються в системі. На рис 4.1-5 наведено код реалізації логіки очищення системи.

```
Ссылок: 1
internal async void Run()
{
    #region Перевірка чекбоксів
    bool yes = false;
    foreach (var item in Controls.OfType<CheckBox>())
    {
        if (item.Checked)
        {
            yes = true;
            break;
        }
    }
    if (!yes)
    {
        MessageBox.Show("Не обрано жодного пункту", Application.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    #endregion
    double size = 0;
    string path;
    Form1.mainForm.need = true;
    Form1.mainForm.Print();
    await Task.Delay(1000);
    if (checkBox1.Checked)
    {
        await Task.Delay(200);
        //Cleaner.CleanerFolders();
        //path = Environment.GetLogicalDrives()[0] + "lalala";
        //if (Directory.Exists(path))
        //    size += Cleaner.CleanerInFoldersTheFiles(path);
        checkBox1.Checked = false;
        checkBox1.AutoCheck = false;
        if (Settings.Default.DarkTheme)
            checkBox1.ForeColor = Color.FromName("ControlDarkDark");
```

Рис. 4.1 Код реалізації логіки очищення системи

```
    else
        checkBox1.ForeColor = Color.FromName("Control");
}
if (checkBox2.Checked)
{
    await Task.Delay(1000);
    path = Environment.GetFolderPath(Environment.SpecialFolder.Windows) + @"\Temp";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;

    path = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + @"\Temp";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;

    checkBox2.Checked = false;
    checkBox2.AutoCheck = false;
    if (Settings.Default.DarkTheme)
        checkBox2.ForeColor = Color.FromName("ControlDarkDark");
    else
        checkBox2.ForeColor = Color.FromName("Control");
}
if (checkBox3.Checked)
{
    path = Environment.GetFolderPath(Environment.SpecialFolder.Windows) + @"\SoftwareDistribution\DataStore";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;
    checkBox3.Checked = false;
    checkBox3.AutoCheck = false;
    if (Settings.Default.DarkTheme)
        checkBox3.ForeColor = Color.FromName("ControlDarkDark");
    else
        checkBox3.ForeColor = Color.FromName("Control");
}
```

Рис. 4.2 Код реалізації логіки очищення системи

```
if (checkBox4.Checked)
{
    path = Environment.GetFolderPath(Environment.SpecialFolder.Windows) + @"\SoftwareDistribution\Download";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;
    checkBox4.Checked = false;
    checkBox4.AutoCheck = false;
    if (Settings.Default.DarkTheme)
        checkBox4.ForeColor = Color.FromName("ControlDarkDark");
    else
        checkBox4.ForeColor = Color.FromName("Control");
    await Task.Delay(500);
}
if (checkBox5.Checked)
{
    string[] Drives = Environment.GetLogicalDrives();
    foreach (string s in Drives)
    {
        path = $@"{s}$RECYCLE.BIN";
        if (Directory.Exists(path) && Cleaner.FolderSize(path) > 129)
            size += Cleaner.CleanerInFoldersTheFiles(path);
        Cleaner.realsize = 0;
        Cleaner.size = 0;
    }
    checkBox5.Checked = false;
    checkBox5.AutoCheck = false;
    if (Settings.Default.DarkTheme)
        checkBox5.ForeColor = Color.FromName("Control");
    else
        checkBox5.ForeColor = Color.FromName("ControlDarkDark");
    await Task.Delay(500);
    Form1.Cmd("taskkill /F /IM explorer.exe & start explorer.exe");
}
```

Рис. 4.3 Код реалізації логіки очищення системи

```
if (checkBox6.Checked)
{
    path = Environment.GetFolderPath(Environment.SpecialFolder.UserProfile) + @"\Downloads\Telegram Desktop";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;

    path = Environment.GetFolderPath(Environment.SpecialFolder.UserProfile) + @"\Загрузки\Telegram Desktop";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;

    if (Registry.CurrentUser.OpenSubKey(@"SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache") != null)
    {
        foreach (var key in Registry.CurrentUser.OpenSubKey(@"SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\Shell\MuiCache")?.GetValueNames())
        {
            if (key.ToString().Contains("Telegram.exe"))
            {
                path = key.Substring(0, key.LastIndexOf(@"Telegram\") + 8).ToString();
                break;
            }
        }
    }

    if (Directory.Exists(path + "\\tdata"))
    {
        path += "\\tdata";
        DirectoryInfo directoryInfo = new DirectoryInfo(path);

        foreach (var item in directoryInfo.EnumerateDirectories())
        {
            if (item.ToString().Contains("temp_data"))
            {
                size += Cleaner.CleanerInFoldersTheFiles(path + $"\\{item}");
            }
            Cleaner.realsize = 0;
            Cleaner.size = 0;
            if (item.ToString().Contains("user_data"))
            {
                size += Cleaner.CleanerInFoldersTheFiles(path + $"\\{item}");
            }
            Cleaner.realsize = 0;
            Cleaner.size = 0;
        }
    }


    checkBox6.Checked = false;
    checkBox6.AutoCheck = false;
    if (Settings.Default.DarkTheme)
        checkBox6.ForeColor = Color.FromName("ControlDarkDark");
    else
        checkBox6.ForeColor = Color.FromName("Control");
    await Task.Delay(300);
}
```

Рис. 4.4 Код реалізації логіки очищення системи

```
if (checkBox7.Checked)
{
    Cent Browser

    #region Chrome
    path = $@"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData)}\Google\Chrome\User Data\Default\Cache";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;
    #endregion

    #region Opera Gx
    path = $@"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData)}\Opera Software\Opera GX Stable\Cache\Cache_Data";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    path = $@"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData)}\Opera Software\Opera GX Stable\System Cache\Cache_Data";
    if (Directory.Exists(path))
        size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;
    #endregion

    checkBox7.Checked = false;
    checkBox7.AutoCheck = false;
    if (Settings.Default.DarkTheme)
        checkBox7.ForeColor = Color.FromName("ControlDarkDark");
    else
        checkBox7.ForeColor = Color.FromName("Control");
}
if (checkBox8.Checked)
{
    RamCleaner.ClearFileSystemCache();
    checkBox8.Checked = false;
    checkBox8.AutoCheck = false;
    if (Settings.Default.DarkTheme)
        checkBox8.ForeColor = Color.FromName("ControlDarkDark");
    else
        checkBox8.ForeColor = Color.FromName("Control");
}
size = Convert.ToDouble(string.Format("{0:f1}", size / 1024 / 1024));
await Task.Delay(1000);
Form1.mainForm.need = false;
MessageBox.Show($"Було очищено: {size} MB", Application.ProductName, MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
await Task.Delay(1000);
```

Рис. 4.5 Код реалізації логіки очищення системи

Вкладка «Поліпшення» надає функції для поліпшення роботи системи, вимикаючи телеметрію, віджети, припинення класичних програм в фоні та збільшення кешу файлової системи. На рис 4.6 наведено код реалізації логіки поліпшенні системи.

```
internal void Run()
{
    Перевірка чекбоксів
    //if (checkBox1.Checked)
    //if (checkBox2.Checked)
    if (checkBox3.Checked)
    {
        Registry.LocalMachine.CreateSubKey("SOFTWARE\\Policies\\Microsoft\\Dsh").SetValue("AllowNewsAndInterests", 0, RegistryValueKind.DWord);
        checkBox3.Checked = false;
        checkBox3.AutoCheck = false;
        button4.Visible = true;
        if (Settings.Default.DarkTheme)
            checkBox3.ForeColor = Color.FromName("ControlDarkDark");
        else
            checkBox3.ForeColor = Color.FromName("Control");
    }
    if (checkBox4.Checked)
    {
        Registry.CurrentUser.CreateSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\BackgroundAccessApplications").SetValue("GlobalUserDisabled", 1, RegistryValueKind.DWord);
        Registry.CurrentUser.CreateSubKey("Software\\Microsoft\\Windows\\CurrentVersion\\Search").SetValue("BackgroundAppGlobalToggle", 0, RegistryValueKind.DWord);
        checkBox4.Checked = false;
        checkBox4.AutoCheck = false;
        button5.Visible = true;
        if (Settings.Default.DarkTheme)
            checkBox4.ForeColor = Color.FromName("ControlDarkDark");
        else
            checkBox4.ForeColor = Color.FromName("Control");
        //              [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\BackgroundAccessApplications]
        //              "GlobalUserDisabled" = dword:00000001

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

    }
    if (checkBox5.Checked)
    {
        Registry.LocalMachine.OpenSubKey("HKEY_LOCAL_MACHINE\\SYSTEM\\CurrentControlSet\\Control\\Session Manager\\Memory Management", true).SetValue("LargeSystemCache", 1);
```

Рис. 4.6 Код реалізації логіки очищення системи

Вкладка «Інтерфейс» дозволяє користувачам налаштовувати вигляд робочого столу, змінювати тему, налаштовувати іконки та інші елементи інтерфейсу. На рис 4.7 наведено код реалізації логіки поліпшенні інтерфейсу системи.

```
internal void Run()
{
    if (checkBox3.Checked)
    {
        try
        {
            // Відкриваємо ключ реєстру, де зберігаються налаштування ClearType
            RegistryKey key = Registry.CurrentUser.OpenSubKey(@"Control Panel\Desktop", true);

            if (key != null)
            {
                // Встановлюємо значення параметра "FontSmoothing" на "0" для відключення ClearType
                key.SetValue("FontSmoothing", "0", RegistryValueKind.String);


                key.Close();

                if (Settings.Default.DarkTheme)
                    checkBox3.ForeColor = Color.FromName("ControlDarkDark");
                else
                    checkBox3.ForeColor = Color.FromName("Control");
                checkBox1.Checked = false;
            }
            else
            {
                MessageBox.Show("Не вдалося відкрити ключ реєстру.");
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show("Виникла помилка: " + ex.Message);
        }
    }
```

Рис. 4.7 Код реалізації логіки налаштування інтерфейсу

Вкладка «Автозапуск» надає можливість керувати програмами, які запускаються разом із запуском операційної системи, додаючи або вимикаючи їх з автозапуску. На рис 4.8-9 наведено код реалізації логіки керування автозапуском програм.

```
private void Автозапуск_Load(object sender, EventArgs e)
{
    try
    {
        foreach (var key in Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved\\Run").GetValueNames())
        {
            RegistryKey k = Registry.CurrentUser.OpenSubKey($"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved\\Run", true);
            byte[] value = (byte[])Registry.CurrentUser.OpenSubKey($"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved\\Run", true).GetValue(key);

            CheckBox checkbox = new CheckBox();
            panel1.Controls.Add(checkbox);

            checkbox.Location = new Point(12, y);
            checkbox.Text = key.ToString();
            checkbox.ForeColor = Color.White;
            checkbox.Size = new Size(panel1.Height, 17);

            if (value[0] == 0x02)
                checkbox.Checked = true;
            else if (value[0] == 0x03)
                checkbox.Checked = false;
            else
                checkbox.Enabled = false;
            checkbox.Click += (s, a) => {
                if (Settings.Default.ProgramSounds)
                    Sounds.PlaySound3();
                if (checkbox.Checked)
                {
                    value[0] = 0x02;
                    k.SetValue(key, value, RegistryValueKind.Binary);
                }
                else
                {
                    value[0] = 0x03;
                    k.SetValue(key, value, RegistryValueKind.Binary);
                }
            };
            y += 20;
        }
    }
    catch (Exception) { }
```

Рис. 4.8 Реалізація автозапуску програм для поточного користувача системи

```
foreach(var key in Registry.LocalMachine.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved\\Run").GetValueNames()) {
    try
    {
        RegistryKey k = Registry.LocalMachine.OpenSubKey($"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved\\Run", true);
        byte[] value = (byte[])Registry.LocalMachine.OpenSubKey($"SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Explorer\\StartupApproved\\Run", true).GetValue(key);

        CheckBox checkbox = new CheckBox();
        panel1.Controls.Add(checkbox);

        checkbox.Location = new Point(12, y);
        checkbox.Text = key.ToString();
        checkbox.ForeColor = Color.White;
        checkbox.Size = new Size(panel1.Height, 17);

        if (value[0] == 0x02)
            checkbox.Checked = true;
        else if (value[0] == 0x03)
            checkbox.Checked = false;
        else
        {
            checkbox.AutoCheck = false;
            checkbox.ForeColor = Color.FromName("ControlDarkDark");
        }

        checkbox.Click += (s, a) => {
            if (Settings.Default.ProgramSounds)
                Sounds.PlaySound3();
            if (checkbox.Checked)
            {
                value[0] = 0x02;
                k.SetValue(key, value, RegistryValueKind.Binary);
            }
            else
            {
                value[0] = 0x03;
                k.SetValue(key, value, RegistryValueKind.Binary);
            }
        };
        y += 20;
    }
    catch (Exception) { }
}
```

Рис. 4.9 Реалізація автозапуску програм для всіх користувачів системи

Вкладка «Налаштування» містить різні параметри конфігурації застосунку, а також функцію перевірки наявності оновлень. Користувач може натиснути кнопку для перевірки оновлення, і застосунок звертається до GitHub

API для перевірки останньої доступної версії. На рис 4.10 наведено код реалізації логіки параметри конфігурації застосунку.

```
private void checkBox1_Click(object sender, EventArgs e)
{
    Sounds.PlaySound3();
    if (checkBox1.Checked)
        Properties.Settings.Default.SystemNotification = true;
    else
        Properties.Settings.Default.SystemNotification = false;
    Properties.Settings.Default.Save();
}

Ссылок: 1
private void checkBox2_Click(object sender, EventArgs e)
{
    if (checkBox2.Checked)
        Properties.Settings.Default.ProgramSounds = true;
    else
        Properties.Settings.Default.ProgramSounds = false;
    Properties.Settings.Default.Save();
}

Ссылок: 1
private void checkBox3_Click(object sender, EventArgs e)
{
    if (checkBox3.Checked)
    {
        Properties.Settings.Default.AutoUpdate = true;
    }
    else
    {
        Properties.Settings.Default.AutoUpdate = false;
    }
    Properties.Settings.Default.Save();
}

Ссылок: 1
private void checkBox4_Click(object sender, EventArgs e)
{
    if (checkBox4.Checked)
        Properties.Settings.Default.Animation = true;
    else
        Properties.Settings.Default.Animation = false;
    Properties.Settings.Default.Save();
}

Ссылок: 1
async internal void button1_Click(object sender, EventArgs e)
{
    await new Internet().UpdateChecker();
}
```

Рис. 4.10 Код реалізації логіки налаштування інтерфейсу

Вкладка «Системна інформація» надає користувачам інформацію про їхню систему, включаючи дані про процесор, оперативну пам'ять, дисковий

простір, операційну систему та інші важливі характеристики. На рис 4.11-12 наведено код реалізації логіки відображення системної інформації.

```
internal async void Run()
{
    await Task.Run(async () =>
    {
        if (label1.Text == "")
        {
            try
            {
                string processor = "", coreCounts = "", friquencyProc = "", videoAdapters = "", motherboard = "";
                ulong memory = 0;

                #region Проц
                foreach (var item in new ManagementObjectSearcher("root\\cimv2", "select * from win32_processor").Get())
                {
                    processor = item.GetPropertyValue("Name").ToString();
                    coreCounts = item.GetPropertyValue("NumberOfCores").ToString();
                    friquencyProc = item.GetPropertyValue("MaxClockSpeed").ToString();
                }
                #endregion

                #region Материнська плата
                foreach (var item in new ManagementObjectSearcher("root\\cimv2", "select * from win32_baseboard").Get())
                    motherboard += item.GetPropertyValue("Manufacturer").ToString() + " " + item.GetPropertyValue("Product").ToString();
                #endregion

                #region ОЗУ
                foreach (var item in new ManagementObjectSearcher("root\\cimv2", "select * from win32_physicalmemory").Get())
                    memory += Convert.ToUInt64(item.GetPropertyValue("Capacity")) / 1024 / 1024;
                #endregion

                #region Відеокарта
                foreach (var item in new ManagementObjectSearcher("root\\cimv2", "select * from win32_videocontroller").Get())
                    if (videoAdapters.Length == 0 && item.GetPropertyValue("Name").ToString().Remove(5) == "Intel")
                        videoAdapters += item.GetPropertyValue("Name").ToString();

                foreach (var item in new ManagementObjectSearcher("root\\cimv2", "select * from win32_videocontroller").Get())
                    if (item.GetPropertyValue("Name").ToString().Remove(5) != "Intel")
                        if (videoAdapters.Length == 0)
                            videoAdapters += item.GetPropertyValue("Name").ToString();
                        else
                            videoAdapters += "\n" + item.GetPropertyValue("Name").ToString();

                #endregion
```

Рис. 4.11 Код реалізації логіки налаштування інтерфейсу

```
                string OSBitDepth = "";
                if (Environment.Is64BitOperatingSystem)
                    OSBitDepth = "64-х розрядна";
                else
                    OSBitDepth = "32-х розрядна";
                label1.Text = $"ОС: Windows {Settings.Default.WinVersion} " +
                              //"ОС: " + Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Microsoft\Windows NT\CurrentVersion").GetValue("ProductName").ToString() + " " +
                              Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Microsoft\Windows NT\CurrentVersion").GetValue("DisplayVersion").ToString() + " " + OSBitDepth + Environment.NewLine +
                              "Ім'я комп'ютера: " + Environment.MachineName + Environment.NewLine +
                              $"Процесор: {processor}" + Environment.NewLine +
                              "Частота процессора: " + Convert.ToInt32(friquencyProc) / 1000 + ".00GHz" + Environment.NewLine +
                              "Кількість ядер процесору: " + coreCounts + Environment.NewLine +
                              "Кількість потоків процесора: " + Environment.ProcessorCount + Environment.NewLine +
                              $"Материнська плата: {motherboard}" + Environment.NewLine +
                              "ОЗУ: " + memory + " MB" + Environment.NewLine +
                              $"Відеокарта: {videoAdapters}";
            }
            catch (Exception)
            {
                MessageBox.Show("Інструмент WMI відсутній на ПК");
            }
        }
        await Task.Delay(500);
        while (Form1.mainForm.active.Text == "SystemInfo")
        {
            foreach (var item in new ManagementObjectSearcher("root\\cimv2", "select * from win32_processor").Get())
            {
                label2.Text = $"Поточна тактова частота процесора: {item.GetPropertyValue("CurrentClockSpeed")} MHz";
            }
            foreach (var item in new ManagementObjectSearcher("root\\cimv2", "select * from win32_operatingSystem").Get())
            {
                label3.Text = $"Вільне місце ОЗУ: {Convert.ToUInt64(item.GetPropertyValue("FreePhysicalMemory")) / 1024} Мб";
            }
            await Task.Delay(1000);
        }
});
```

Рис. 4.12 Код реалізації логіки налаштування інтерфейсу

4.3 Тестування застосунку

Для забезпечення якості та стабільності роботи застосунку було проведено ретельне тестування всіх його функцій. Тестування спрямоване на виявлення помилок, перевірку коректності виконання функцій та відповідність роботи застосунку встановленим вимогам. Тестування охоплює всі основні розділи та функції застосунку, включаючи очищення системи, оптимізацію, налаштування інтерфейсу, керування автозапуском програм, перевірку наявності оновлень та відображення системної інформації.

У таблиці 4.1. представлено результати тестування, що включають опис протестованих функцій та очікувані результати їх виконання. Кожна функція була перевірена на предмет її коректного виконання та відповідності очікуванням. Це дозволяє забезпечити впевненість у тому, що застосунок працює стабільно і відповідає потребам користувачів.

Таблиця 4.1

Тестування функціональності застосунку

Пункт меню, який тестувався	Протестована функція	Очікуваний результат	Результат тесту
Запуск застосунку	Ініціація застосунку	Після натискання на файл застосунку, застосунок відкривається	Успіх
Головне вікно	Перемикання між вкладками	Користувач може без проблем перемикатися між вкладками	Успіх
Вкладка «Очищення»	Видалення пустих папок	Порожні папки видалені, відображається повідомлення про успішне завершення	Успіх
Вкладка «Очищення»	Видалення тимчасових файлів	Тимчасові файли видалено, відображається повідомлення про успішне завершення	Успіх

Продовження таблиці 4.1

Пункт меню, який тестувався	Протестована функція	Очікуваний результат	Результат тесту
Вкладка «Очищення»	Видалення файлів центру оновлень	Файли центру оновлень видалені, відображається повідомлення про успішне завершення	Успіх
Вкладка «Очищення»	Очищення журналу подій	Журнал подій очищено, відображається повідомлення про успішне завершення	Успіх
Вкладка «Очищення»	Очищення кошика	Кошик очищено, відображається повідомлення про успішне завершення	Успіх
Вкладка «Очищення»	Очищення кешу браузера	Кеш браузера очищено, відображається повідомлення про успішне завершення	Успіх
Вкладка «Очищення»	Очищення кешу Telegram	Кеш Telegram очищено, відображається повідомлення про успішне завершення	Успіх
Вкладка «Очищення»	Очищення кешу оперативної пам'яті	Кеш оперативної пам'яті очищено, відображається повідомлення про успішне завершення	Успіх
Вкладка «Поліпшення»	Вимкнення телеметрії	Телеметрію вимкнено, відображається повідомлення про успішне виконання	Успіх

Продовження таблиці 4.1

Пункт меню, який тестувався	Протестована функція	Очікуваний результат	Результат тесту
Вкладка «Поліпшення»	Вимкнення віджетів	Віджети вимкнені, відображається повідомлення про успішне виконання	Успіх
Вкладка «Поліпшення»	Вимкнення класичних програм у фоновому режимі	Класичні програми вимкнені у фоновому режимі, відображається повідомлення про успішне виконання	Успіх
Вкладка «Поліпшення»	Збільшення кешу файлової системи	Кеш файлової системи збільшено, відображається повідомлення про успішне виконання	Успіх
Вкладка «Інтерфейс»	Зміна теми інтерфейсу	Тема інтерфейсу змінюється, новий вигляд застосовується негайно	Успіх
Вкладка «Інтерфейс»	Встановлення класичного переглядача фотографій	Класичний переглядач фотографій встановлено, відображається повідомлення про успішне виконання	Успіх
Вкладка «Інтерфейс»	Встановлення старого контекстного меню	Старе контекстне меню встановлено, відображається повідомлення про успішне виконання	Успіх

Продовження таблиці 4.1

Пункт меню, який тестувався	Протестована функція	Очікуваний результат	Результат тесту
Вкладка «Інтерфейс»	Включення згладжування шрифту	Згладжування шрифту включено, відображається повідомлення про успішне виконання	Успіх
Вкладка «Автозапуск»	Ввімкнення програми в автозапуск	Програма ввімкнена в автозапуску, відображається у списку автозапуску як в активному стані	Успіх
Вкладка «Автозапуск»	Вимкнення програми з автозапуску	Програма вимкнена з автозапуску, відображається у списку автозапуску як в не активному стані	Успіх
Вкладка «Налаштування»	Ввімкнути сповіщення	Застосунок виводить жодних сповіщень при певних діях налаштувань	Успіх
Вкладка «Налаштування»	Вимкнути сповіщення	Застосунок не виводить жодних сповіщень при певних діях налаштувань	Успіх
Вкладка «Налаштування»	Ввімкнути звуки застосунку	Застосунок відтворює звуки при натисканні на вкладки та пункти вибору налаштувань	Успіх
Вкладка «Налаштування»	Вимкнути звуки застосунку	Застосунок не відтворює жодних звуків при натисканні на вкладки та пункти вибору налаштувань	Успіх

Продовження таблиці 4.1

Пункт меню, який тестувався	Протестована функція	Очікуваний результат	Результат тесту
Вкладка «Налаштування»	Ввімкнення автоматичного оновлення застосунку	Застосунок автоматично перевіряє оновлення при запуску застосунку та при наявності новішої версії, виводить повідомлення та пропонує оновитися	Успіх
Вкладка «Налаштування»	Вимкнення автоматичного оновлення застосунку	Застосунок автоматично не перевіряє оновлення при запуску застосунку	Успіх
Вкладка «Налаштування»	Ввімкнення анімацій застосунку	При взаємодії з інтерфейсом застосунку, відтворюються плавні переходи між вкладками, згортання та закриття застосунку	Успіх
Вкладка «Налаштування»	Вимкнення анімацій застосунку	При взаємодії з інтерфейсом застосунку, не відтворюються плавні переходи між вкладками, згортання та закриття застосунку	Успіх
Вкладка «Налаштування»	Перевірення оновлення	Виводиться повідомлення про наявність або відсутність нової версії.	Успіх

Продовження таблиці 4.1

Пункт меню, який тестувався	Протестована функція	Очікуваний результат	Результат тесту
Вкладка «Налаштування»	Завантаження нової версії	Нова версія завантажується та застосунок автоматично запускається новішої версії.	Успіх
Вкладка «Системна інформація»	Відображення інформації про систему	Відображається інформація про апаратне і програмне забезпечення системи.	Успіх
Вкладка «Системна інформація»	Відображення використання ресурсів системи	Відображається інформація про використання апаратного забезпечення системи в поточному часі	Успіх

ВИСНОВКИ

У дипломній роботі було розроблено застосунок для налаштування операційної системи Windows. Метою даної роботи спростити процес налаштування операційної системи Windows шляхом впровадження застосунку для налаштування операційної системи Windows, за допомогою якого можна легко налаштувати систему та поліпшити її продуктивність, відповідно до потреб користувача.

Проведено аналіз вимог користувачів, що дозволило визначити основні функції застосунку та створити інтуїтивно зрозумілий інтерфейс. Розроблена архітектура забезпечує модульність та масштабованість, що є важливим для подальшого розвитку та оновлення програмного забезпечення.

У ході реалізації проекту було впроваджено основні функціональні модулі, які дозволяють налаштовувати системні параметри, поліпшувати продуктивність, а також керувати програмами автозапуску. Особлива увага приділена стабільності та надійності роботи застосунку, що підтверджено результатами проведених тестувань.

Розроблено план підтримки та оновлення програмного забезпечення, який включає можливість інтеграції додаткових функцій у майбутньому, таких як відстеження системних ресурсів у реальному часі та автоматичне оновлення налаштувань.

У результаті роботи проект успішно досягнув поставленої мети, надавши користувачам зручний інструмент для налаштування Windows. Застосунок має високу практичну цінність, оскільки дозволяє економити час і зусилля при виконанні рутинних завдань з налаштування системи. У перспективі можна розглянути можливість подальшого розширення функціоналу для ще більшої зручності та ефективності використання.

Загалом, розроблений застосунок є ефективним інструментом, що значно полегшує процес управління системними параметрами, підвищує

продуктивність роботи та забезпечує комфортне використання операційної системи.

1. Проведено аналіз інструментальних засобів та існуючих аналогів застосунків для налаштування операційної системи Windows.
2. Визначено вимоги та спроектовано інтерфейс до застосунку для налаштування операційної системи Windows. Виконано проектування вимог за допомогою діаграми прецедентів.
3. Виконано проектування загального вигляду всіх файлів та класів меню за допомогою діаграм класів меню та файловою структурою проєкту.
4. Спроектовано процес взаємодії користувача з очищенням за допомогою діаграми діяльності очищення, взаємодії користувача та об’єктів для оновлення застосунку за допомогою діаграм послідовностей.
5. Розроблено основні та додаткові функції до застосунку для налаштування операційної системи Windows.
6. Додано підтримку оновлення застосунку для налаштування операційної системи Windows. Для проведення даної операції було використано API GitHub.
7. Реалізовано застосунок для налаштування операційної системи Windows. Для реалізації проєкту було використано мову програмування C# і такі інструменти як Visual Studio, GitHub, .NET Framework.
8. Стиснено застосунок в розмірі для налаштування операційної системи Windows. Для проведення даної операції було використано інструмент Mpress.
9. Проведено тестування застосунку для налаштування операційної системи Windows.

ПЕРЕЛІК ПОСИЛАНЬ

1. Mpress [Електронний ресурс] – Режим доступу до ресурсу: https://www.autohotkey.com/mpress/mpress_web.htm.

2. API GitHub [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots/api

3. UMLetino [Електронний ресурс] – Режим доступу до ресурсу: https://www.umletino.com/

4. CCleaner [Електронний ресурс] – Режим доступу до ресурсу: https://ccleaner.org.ua/uk/

5. CleanMyPC [Електронний ресурс] – Режим доступу до ресурсу: https://macpaw.com/cleanmypc

6. Fortect [Електронний ресурс] – Режим доступу до ресурсу: https://gridinsoft.ua/online-virus-scanner/url/fortect-com.

7. An Intro about Packer. URL: https://medium.com/@muthamil1001/an-intro-about-packer-malware-ffa5a4d788d9.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка застосунку мовою C# для налаштування операційної системи Windows

Виконав студент 4 курсу
Групи ПД-43
Кисюк Віталій Вікторович
Керівник роботи
Старший викладач кафедри ІПЗ Гаманюк Ігор Михайлович

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** – спростити процес налаштування операційної системи Windows шляхом впровадження застосунку для налаштування операційної системи Windows.
- **Об'єкт дослідження** – процес налаштування операційної системи Windows.
- **Предмет дослідження** – програмне забезпечення, яке призначене для налаштування операційної системи Windows.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз інструментальних засобів для налаштування операційної системи Windows.
2. Здійснити огляд та проаналізувати існуючі застосунки для налаштування операційної системи Windows.
3. Визначити вимоги та спроектувати інтерфейс до застосунку для налаштування операційної системи Windows.
4. Розробити основні та додаткові функції до застосунку для налаштування операційної системи Windows.
5. Реалізувати застосунок для налаштування операційної системи Windows.
6. Провести тестування застосунку для налаштування операційної системи Windows.

3

АНАЛІЗ АНАЛОГІВ

Застосунки / Характеристики	CCleaner	CleanMyPC	Fortect	Tweaker in 1
Управління автозапуском програм	+	+	-	+
Звільнення кешу оперативної пам'яті	-	-	-	+
Показ системної інформації	+	-	-	+
Відстеження використання ресурсів системи в поточному часі	-	-	-	+
Видалення кешу телеграма	-	-	-	+
Українська локалізація	+	+	-	+
Однофайловий застосунок	-	-	-	+
Запуск без необхідності файла інсталятора	+	-	-	+
Розмір	50-100 Мб	50,2 Мб	281 Мб	0.2 Мб

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги:

1. Пошук та очищення не потрібних та тимчасових файлів, що залишилися після роботи програм.
2. Можливість вимкнення функцій системи, якими користувач не користується.
3. Алгоритм для поліпшення оперативної пам’яті шляхом очищення кешованного розділу.
4. Менеджер по управлінню автозавантаженням програм.
5. Можливість оновлення застосунку.
6. Забезпечення перегляду системної інформації.
7. Можливість відстеження рівня навантаженості комплектуючих системи в поточному часі.

Нефункціональні вимоги:

1. Швидкість реакції застосунку на запити користувача в межах двух секунд.
2. Низький рівень використання системних ресурсів, не більше ніж 5% оперативної пам'яті та 15% ресурсів процесору.
3. Сумісність з Windows 8/8.1/10/11.
4. Українська локалізація.
5. Застосунок не повинен бути розміром більше ніж 20 Мбайт.

5

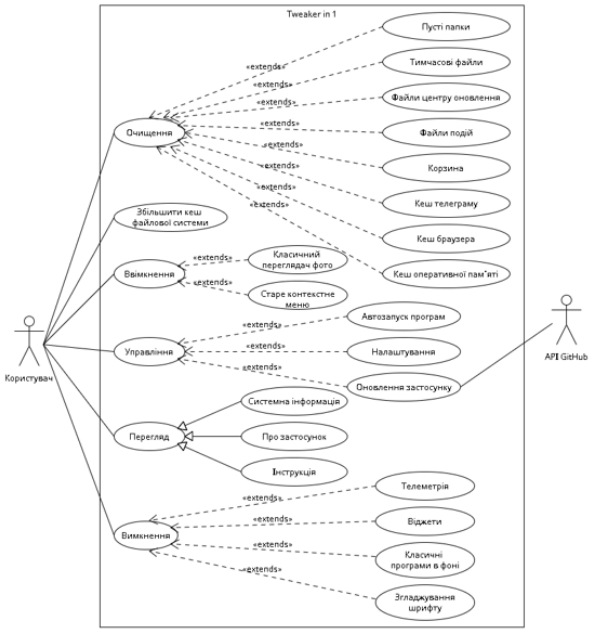
Програмні засоби та інструменти реалізації

Visual Studio 2022

GitHub

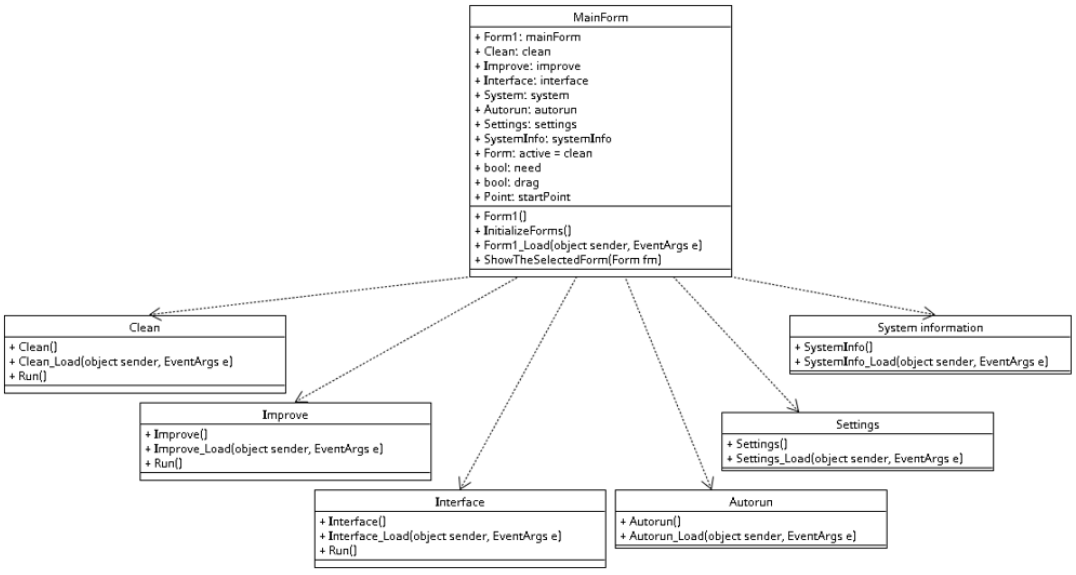
6

Діаграма прецедентів



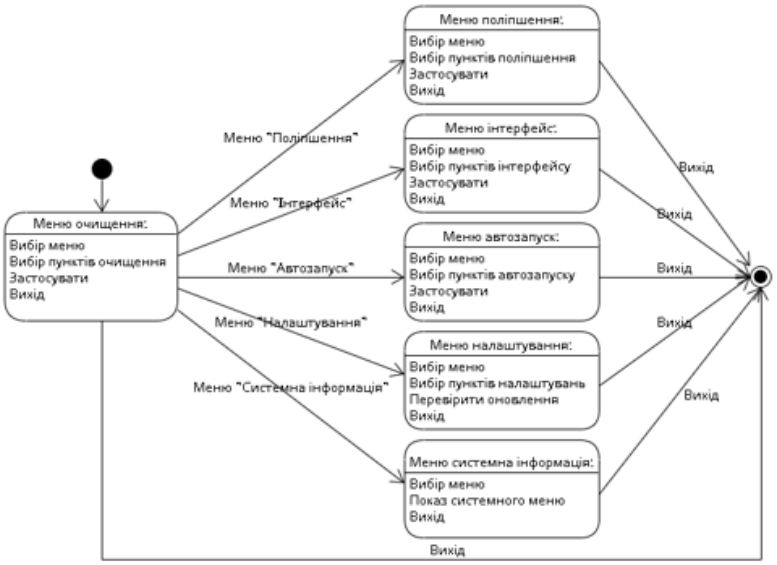
7

Діаграма класів меню



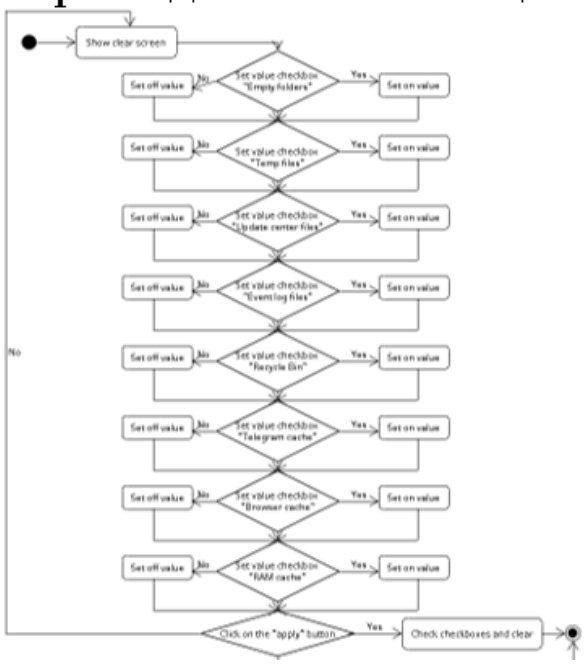
8

Діаграма станів меню



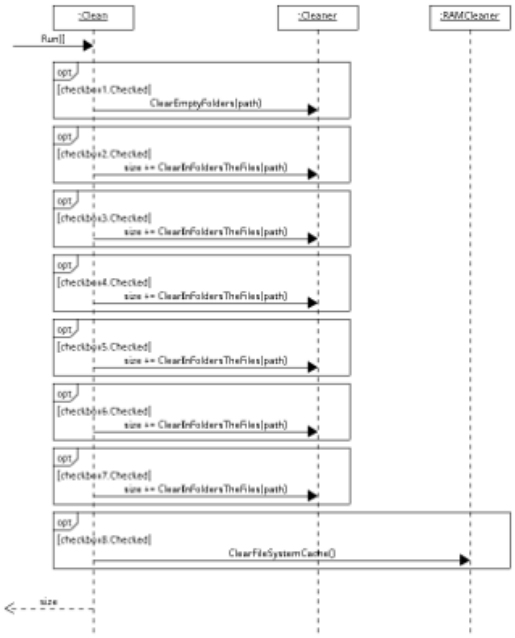
9

Діаграма діяльності очищення



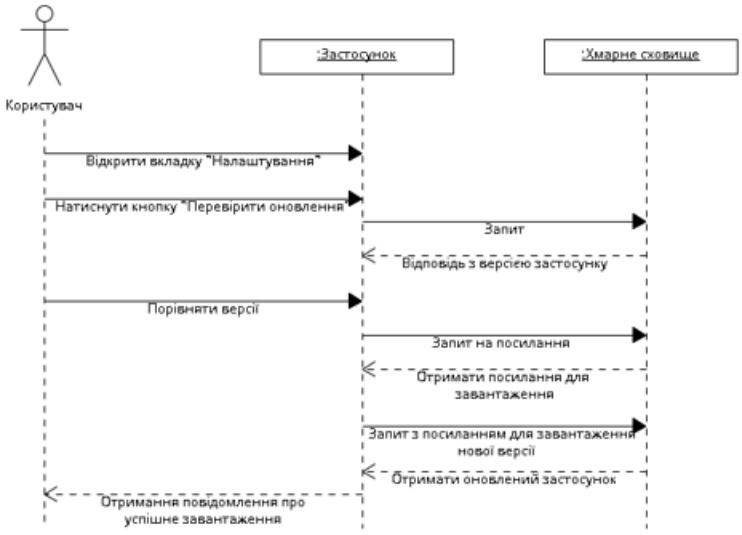
10

Діаграма послідовності очищення



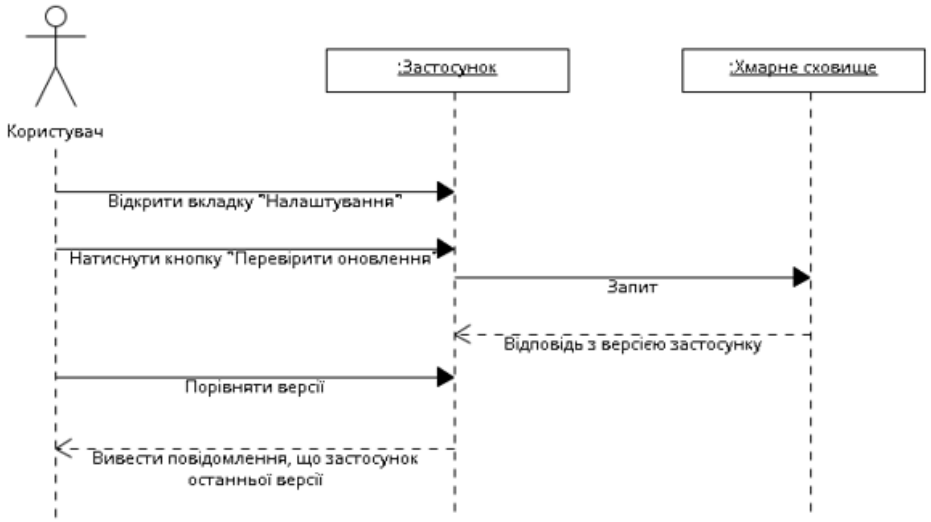
11

Діаграма послідовності оновлення: «Існує новіше версія»



12

Діаграма послідовності оновлення: «Застосунок останньої версії»



13

Файлова структура проєкту

14

ЕКРАННІ ФОРМИ

Головна вкладка застосунку «Очищення»

Виведення повідомлення про загальний розмір очищення файлів

15

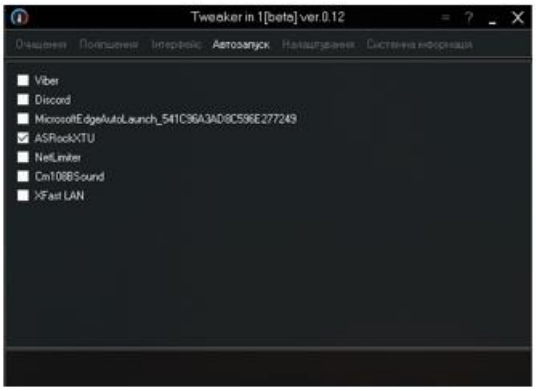
ЕКРАННІ ФОРМИ

Вкладка «Поліпшення»

Вкладка «Інтерфейс»

16

ЕКРАННІ ФОРМИ



Вкладка «Автозапусю»

Вкладка «Налаштування»

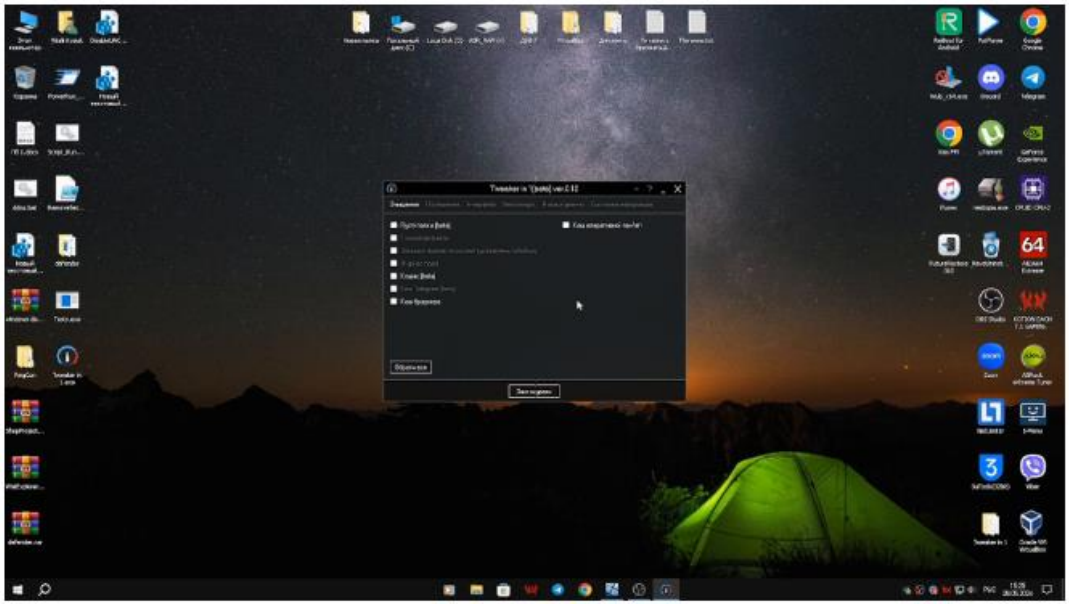
17

ЕКРАННІ ФОРМИ

Вкладка «Системна інформація»

18

ДЕМОНСТРАЦІЯ РОБОТИ ЗАСТОСУНКУ



19

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Кисюк В.В., Гаманюк І.М. Використання Windows Forms для розробки програмного забезпечення щодо налаштування операційної системи Windows: Матеріали IV Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. 15.05.2024, ДУІКТ, м.Київ. К.:ДУІКТ – с.224-227
2. Кисюк В.В., Гаманюк І.М. Використання WinAPI для розробки програмного забезпечення щодо налаштування операційної системи Windows: Матеріали IV Всеукраїнської науково-практичної конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. 15.05.2024, ДУІКТ, м.Київ. К.:ДУІКТ – ст.297-298

20

ВИСНОВКИ

1. Проведено аналіз інструментальних засобів та існуючих аналогів застосунків для налаштування операційної системи Windows.
2. Визначено вимоги та спроектовано інтерфейс до застосунку для налаштування операційної системи Windows. Виконано проектування вимог за допомогою діаграми прецедентів.
3. Виконано проектування загального вигляду всіх файлів та класів меню за допомогою діаграм класів меню та файловою структурою проєкту.
4. Спроектовано процес взаємодії користувача з очищенням за допомогою діаграми діяльності очищення, взаємодії користувача та об'єктів для оновлення застосунку за допомогою діаграм послідовностей.
5. Розроблено основні та додаткові функції до застосунку для налаштування операційної системи Windows.
6. Додано підтримку оновлення застосунку для налаштування операційної системи Windows. Для проведення даної операції було використано API GitHub.
7. Реалізовано застосунок для налаштування операційної системи Windows. Для реалізації проєкту було використано мову програмування C# і такі інструменти як Visual Studio, GitHub, .NET Framework.
8. Стиснено застосунок в розмірі для налаштування операційної системи Windows. Для проведення даної операції було використано інструмент Mpress.
9. Проведено тестування застосунку для налаштування операційної системи Windows.

ДОДАТОК Б. ЛІСТИНГ ПРОГРАМНИХ МОДУЛІВ

MainForm.cs

```
using Microsoft.Win32;
using System;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using Tweaker_in_1.FunctionalForForms;
using Tweaker_in_1.Properties;
using Tweaker_in_1.Форми;

namespace Tweaker_in_1 {
  public partial class Form1: Form {
    public static Form1 mainForm;
    public static Clean очищення;
    public static Improve оптимізація;
    public static Interface інтерфейс;
    public static Interface система;
    internal static AutoRun автозапуск;
    public static Form_Settings form_Settings;
    public static SystemInfo systemInfo;
    internal Form active = очищення;
    internal bool need;
    private bool drag;
    private Point startPoint;

    public Form1() {
      InitializeComponent();
      mainForm = this;
      InitializeForms();
    }

    internal void InitializeForms() {
      очищення = new Clean();
      очищення.TopLevel = false;
      panel3.Controls.Add(очищення);

      оптимізація = new Improve();
      оптимізація.TopLevel = false;
      panel3.Controls.Add(оптимізація);

      інтерфейс = new Interface();
      інтерфейс.TopLevel = false;
      panel3.Controls.Add(інтерфейс);

      система = new Interface();
      система.TopLevel = false;
      panel3.Controls.Add(система);

      автозапуск = new AutoRun();
      автозапуск.TopLevel = false;
      panel3.Controls.Add(автозапуск);

      form_Settings = new Form_Settings();
      form_Settings.TopLevel = false;
      panel3.Controls.Add(form_Settings);

      systemInfo = new SystemInfo();
      systemInfo.TopLevel = false;
      panel3.Controls.Add(systemInfo);
    }

    private async void Form1_Load(object sender, EventArgs e) {
      #region Перевірка віндовс
      CheckWinVersion.GetWindowsVersionName();
      #endregion

      if (Settings.Default.AutoUpdate)
        await new Internet().UpdateChecker();

      #region Перевірка Settings
      if (Settings.Default.SystemNotification)
        form_Settings.checkBox1.Checked = true;
      else
        form_Settings.checkBox1.Checked = false;

      if (Settings.Default.ProgramSounds)
        form_Settings.checkBox2.Checked = true;
      else
        form_Settings.checkBox2.Checked = false;
      if (Settings.Default.AutoUpdate)
        form_Settings.checkBox3.Checked = true;
      else
        form_Settings.checkBox3.Checked = false;
      if (Settings.Default.Animation)
        form_Settings.checkBox4.Checked = true;
      else
        form_Settings.checkBox4.Checked = false;

      if (Settings.Default.DarkTheme) {
        FormTheme.DarkTheme();
        form_Settings.checkBox4.Checked = true;
        button1.ForeColor = Color.FromArgb(255, 255, 255);
      } else {
        FormTheme.LightTheme();
        form_Settings.checkBox4.Checked = false;
        button1.ForeColor = Color.FromArgb(0, 0, 0);
      }
      #endregion

      #region Перевірка Очищення
      if
(Directory.Exists(Environment.GetFolderPath(Environment.SpecialF
older.Windows) + @ "\Temp") &&
File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.
LocalApplicationData) + @ "\Temp")) {
        if
(Cleaner.FolderSize(Environment.GetFolderPath(Environment.Speci
alFolder.Windows) + @ "\Temp") >= 1 &&
Cleaner.FolderSize(Environment.GetFolderPath(Environment.Specia
lFolder.LocalApplicationData) + @ "\Temp") >= 50000000) {
          очищення.checkBox2.AutoCheck = false;
          if (Settings.Default.DarkTheme)
            очищення.checkBox2.ForeColor =
Color.FromName("Control");
          else
            очищення.checkBox2.ForeColor =
Color.FromName("ControlDarkDark");
        }
      }
      if
(Directory.Exists(Environment.GetFolderPath(Environment.SpecialF
older.Windows) + @ "\SoftwareDistribution\DataStore")) {
        if
(Cleaner.FolderSize(Environment.GetFolderPath(Environment.Speci
alFolder.Windows) + @ "\SoftwareDistribution\DataStore") <= 1) {
          очищення.checkBox3.AutoCheck = false;
          if (Settings.Default.DarkTheme)
            очищення.checkBox3.ForeColor =
Color.FromName("ControlDarkDark");
          else
            очищення.checkBox3.ForeColor =
Color.FromName("Control");
        }
      }

      if
(Directory.Exists(Environment.GetFolderPath(Environment.SpecialF
older.Windows) + @ "\SoftwareDistribution\Download")) {
```

```
        if
(Cleaner.FolderSize(Environment.GetFolderPath(Environment.Speci
alFolder.Windows) + @ "\SoftwareDistribution\Download") <= 1) {
          очищення.checkBox4.AutoCheck = false;
          if (Settings.Default.DarkTheme)
            очищення.checkBox4.ForeColor =
Color.FromName("ControlDarkDark");
          else
            очищення.checkBox4.ForeColor =
Color.FromName("Control");
        }
      }

      string[] Drives = Environment.GetLogicalDrives();
      string path = "";
      foreach(string s in Drives) {
        path = $ @ "{s}$RECYCLE.BIN";
        try {
          if (Directory.Exists(path) && Cleaner.FolderSize(path) > 129)
{
            очищення.checkBox5.AutoCheck = true;
            if (Settings.Default.DarkTheme)
              очищення.checkBox5.ForeColor =
Color.FromName("Control");
            else
              очищення.checkBox5.ForeColor =
Color.FromName("ControlDarkDark");
            break;
          }
        } catch (Exception) {}
      }

      if
(Directory.Exists(Environment.GetFolderPath(Environment.SpecialF
older.UserProfile) + @ "\Downloads\Telegram Desktop") ||
Directory.Exists(Environment.GetFolderPath(Environment.SpecialF
older.UserProfile) + @ "\Загрузки\Telegram Desktop")) {
        if
(Cleaner.FolderSize(Environment.GetFolderPath(Environment.Speci
alFolder.UserProfile) + @ "\Downloads\Telegram Desktop") > 1 ||
Cleaner.FolderSize(Environment.GetFolderPath(Environment.Specia
lFolder.UserProfile) + @ "\Загрузки\Telegram Desktop") > 1) {
          очищення.checkBox6.AutoCheck = true;
          if (Settings.Default.DarkTheme)
            очищення.checkBox6.ForeColor =
Color.FromName("Control");
          else
            очищення.checkBox6.ForeColor =
Color.FromName("ControlDarkDark");
        }
      }

      if (Directory.Exists($ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\CentBrowser\User Data\Default\Cache") ||
Directory.Exists($ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Google\Chrome\User Data\Default\Cache") ||
Directory.Exists($ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Opera Software\Opera GX Stable\Cache") ||
Directory.Exists($ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Opera Software\Opera GX Stable\System
Cache\Cache_Data")) {
        if (Cleaner.FolderSize($ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\CentBrowser\User Data\Default\Cache") > 1 ||
Cleaner.FolderSize($ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Google\Chrome\User Data\Default\Cache") > 1 ||
Cleaner.FolderSize($ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Opera Software\Opera GX Stable\Cache") > 1 ||
Cleaner.FolderSize($ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Opera Software\Opera GX Stable\System
Cache\Cache_Data") > 1) {
          очищення.checkBox7.AutoCheck = true;
          if (Settings.Default.DarkTheme)
            очищення.checkBox7.ForeColor =
Color.FromName("Control");
          else
            очищення.checkBox7.ForeColor =
Color.FromName("ControlDarkDark");
        }
      }
      очищення.checkBox8.AutoCheck = true;
      if (Settings.Default.DarkTheme)
        очищення.checkBox8.ForeColor =
Color.FromName("Control");
      else
        очищення.checkBox8.ForeColor =
Color.FromName("ControlDarkDark");
      #endregion

      ShowTheSelectedForm(очищення);
      await Task.Delay(500);
      mainForm.Location = new Point(mainForm.Location.X,
mainForm.Location.Y + 490);

      await Task.Delay(500);
      AnimationFormUo();
    }

    private async void ShowTheSelectedForm(Form fm) {
      await Task.Delay(300);
      if (fm != null && fm != active) {
        if (active != null)
          active.Visible = false;
        active = fm;
        fm.Visible = true;
      }
    }

    private async void AnimationFormUo() {
      while (mainForm.Opacity < 0.98) {
        mainForm.Opacity += 0.02;
        mainForm.Location = new Point(mainForm.Location.X,
mainForm.Location.Y - 10);
        await Task.Delay(1);
      }
    }

    private async void AnimationFormDown() {
      while (mainForm.Opacity > 0) {
        mainForm.Opacity -= 0.02;
        mainForm.Location = new Point(mainForm.Location.X,
mainForm.Location.Y + 10);
        await Task.Delay(1);
      }
    }

    internal static void Cmd(string line) {
      Process.Start(new ProcessStartInfo {
        FileName = "cmd.exe", Arguments = $ @ "/c {line}", Verb =
"runas", WindowStyle = ProcessWindowStyle.Hidden
      }).WaitForExit();
    }

    internal async void Print() {
      while (need) {
        button10.Text = "Застосовую";
        await Task.Delay(200);
        button10.Text += ".";
        await Task.Delay(200);
        button10.Text += ".";
        await Task.Delay(200);
        button10.Text += ".";
        await Task.Delay(500);
      }
      button10.Text = "Застосувати";
    }

    private void button1_Click(object sender, EventArgs e) {
      Sounds.PlaySound3();
```

```
    if (button1.ForeColor != Color.FromArgb(255, 255, 255)) {
      if (Settings.Default.DarkTheme) {
        foreach(Button item in panel2.Controls.OfType < Button > ())
        item.ForeColor = Color.FromName("ControlDarkDark");
        button1.ForeColor = Color.FromArgb(255, 255, 255);
      } else {
        foreach(Button item in panel2.Controls.OfType < Button > ())
        item.ForeColor = Color.FromName("ControlLightLight");
        button1.ForeColor = Color.FromArgb(0, 0, 0);
      }
      ShowTheSelectedForm(очищення);
      if (!button10.Visible)
        button10.Visible = true;
    }
  }

  private void button2_Click(object sender, EventArgs e) {
    Sounds.PlaySound3();
    if (button2.ForeColor != Color.FromArgb(255, 255, 255)) {
      if (Settings.Default.DarkTheme) {
        foreach(Button item in panel2.Controls.OfType < Button > ())
        item.ForeColor = Color.FromName("ControlDarkDark");
        button2.ForeColor = Color.FromArgb(255, 255, 255);
      } else {
        foreach(Button item in panel2.Controls.OfType < Button > ())
        item.ForeColor = Color.FromName("ControlLightLight");
        button2.ForeColor = Color.FromArgb(0, 0, 0);
      }
      ShowTheSelectedForm(оптимізація);
      if (!button10.Visible)
        button10.Visible = true;
    }
  }

  private void button3_Click(object sender, EventArgs e) {
    Sounds.PlaySound3();
    if (button3.ForeColor != Color.FromArgb(255, 255, 255)) {
      if (Settings.Default.DarkTheme) {
        foreach(Button item in panel2.Controls.OfType < Button > ())
        item.ForeColor = Color.FromName("ControlDarkDark");
        button3.ForeColor = Color.FromArgb(255, 255, 255);
      } else {
        foreach(Button item in panel2.Controls.OfType < Button > ())
        item.ForeColor = Color.FromName("ControlLightLight");
        button3.ForeColor = Color.FromArgb(0, 0, 0);
      }
      ShowTheSelectedForm(інтерфейс);
      if (!button10.Visible)
        button10.Visible = true;
    }
  }

  private void button4_Click(object sender, EventArgs e) {
    Sounds.PlaySound3();
    if (button4.ForeColor != Color.FromArgb(255, 255, 255))
      if (Settings.Default.DarkTheme)
        foreach(Button item in panel2.Controls.OfType < Button > ())
{
          item.ForeColor = Color.FromName("ControlDarkDark");
          button4.ForeColor = Color.FromArgb(255, 255, 255);
        }
      else {
        button4.ForeColor = Color.FromArgb(255, 255, 255);
        foreach(Button item in panel2.Controls.OfType < Button > ()) {
          item.ForeColor = Color.FromName("ControlLightLight");
          button4.ForeColor = Color.FromArgb(0, 0, 0);
        }
      }
    ShowTheSelectedForm(form_Settings);
    if (button10.Visible)
      button10.Visible = false;
  }

  private void button5_Click(object sender, EventArgs e) {
    Sounds.PlaySound3();
    if (button5.ForeColor != Color.FromArgb(255, 255, 255))
      if (Settings.Default.DarkTheme)
        foreach(Button item in panel2.Controls.OfType < Button > ())
{
          item.ForeColor = Color.FromName("ControlDarkDark");
          button5.ForeColor = Color.FromArgb(255, 255, 255);
        }
      else
        foreach(Button item in panel2.Controls.OfType < Button > ()) {
          item.ForeColor = Color.FromName("ControlLightLight");
          button5.ForeColor = Color.FromArgb(0, 0, 0);
        }
    ShowTheSelectedForm(systemInfo);
    systemInfo.Run();
  }

  private void button6_Click(object sender, EventArgs e) {
    MessageBox.Show(
      "0.12[beta]" + Environment.NewLine +
      "Підтримка Windows 11" + Environment.NewLine +
      "+ Пункт \"Повне вимкнення та видалення Microsoft
Defender та SmartScreen\"[beta];" + Environment.NewLine +
      "+ Пункт \"Збільшити кеш файлової системи\";" +
Environment.NewLine +
      "Правильне відображення версії Windows в \"Системна
інформація\";" + Environment.NewLine +
      "Виправлено не правильне відображення тексту в
\"Автозапуск\";" + Environment.NewLine +
      "Пофіксено баги програми;" + Environment.NewLine +
      Environment.NewLine +
      Environment.NewLine +
      "0.11[beta]" + Environment.NewLine +
      "+ Пункт \"Очищення кешу оперативної пам'яті\";" +
Environment.NewLine +
      "Пофіксено баги програми;" + Environment.NewLine +
      "Додано анімації програми;" + Environment.NewLine +
      Environment.NewLine +
      Environment.NewLine +
      "0.1 [beta]" + Environment.NewLine +
      "Перша розробка програми;", "Список версій програми");
  }

  private void button7_Click(object sender, EventArgs e) {
    MessageBox.Show("В розробці...", "Справка");
  }

  private async void button8_Click(object sender, EventArgs e) {
    Sounds.PlaySound3();
    AnimationFormDown();
    await Task.Delay(1000);
    notifyIcon1.Visible = true;
    if (form_Settings.checkBox1.Checked)
      notifyIcon1.ShowBalloonTip(1000);
    Hide();
  }

  private async void button9_Click(object sender, EventArgs e) {
    Sounds.PlaySound3();
    AnimationFormDown();
    await Task.Delay(1000);
    Application.Exit();
  }

  private async void відкритиToolStripMenuItem_Click(object
sender, EventArgs e) {
    Show();
    AnimationFormUo();
    await Task.Delay(1000);
    notifyIcon1.Visible = false;
  }

  private async void закритиToolStripMenuItem_Click(object
sender, EventArgs e) {
    AnimationFormDown();
    await Task.Delay(1000);
    Application.Exit();
  }

  private void notifyIcon1_MouseDoubleClick(object sender,
MouseEventArgs e) {
```

```
    Show();
    AnimationFormUo();
    notifyIcon1.Visible = false;
   }

   private void button10_Click(object sender, EventArgs e) {
    if (active.Text == "Очищення")
     очищення.Run();
    else if (active.Text == "Поліпшення")
     оптимізація.Run();
    else if (active.Text == "Інтерфейс")
     інтерфейс.Run();
   }

   private void button11_Click(object sender, EventArgs e) {
    Sounds.PlaySound3();
    if (button11.ForeColor != Color.FromArgb(255, 255, 255))
     if (Settings.Default.DarkTheme)
      foreach(Button item in panel2.Controls.OfType < Button > ())
{
       item.ForeColor = Color.FromName("ControlDarkDark");
       button11.ForeColor = Color.FromArgb(255, 255, 255);
      }
    else {
     button11.ForeColor = Color.FromArgb(255, 255, 255);
     foreach(Button item in panel2.Controls.OfType < Button > ()) {
      item.ForeColor = Color.FromName("ControlLightLight");
      button11.ForeColor = Color.FromArgb(0, 0, 0);
     }
    }
    ShowTheSelectedForm(автозапуск);
    if (button10.Visible)
     button10.Visible = false;
   }

   private void button12_Click(object sender, EventArgs e) {
    Sounds.PlaySound3();
    if (button12.ForeColor != Color.FromArgb(255, 255, 255))
     if (Settings.Default.DarkTheme)
      foreach(Button item in panel2.Controls.OfType < Button > ())
{
       item.ForeColor = Color.FromName("ControlDarkDark");
       button12.ForeColor = Color.FromArgb(255, 255, 255);
      }
    else {
     button12.ForeColor = Color.FromArgb(255, 255, 255);
     foreach(Button item in panel2.Controls.OfType < Button > ()) {
      item.ForeColor = Color.FromName("ControlLightLight");
      button12.ForeColor = Color.FromArgb(0, 0, 0);
     }
    }
    ShowTheSelectedForm(система);
    if (!button10.Visible)
     button10.Visible = true;
   }

   private void pictureBox1_Click(object sender, EventArgs e) {
    if (Application.OpenForms["Сповіщення"] == null) {
     MessageBox.Show("Версія програми: " +
Application.ProductVersion + " [beta]" + Environment.NewLine +
      "Розробник: sulky" + Environment.NewLine +
      "Підтримка Windows 8/8.1/10/11" + Environment.NewLine +
      "Програма розроблена на .NET Framework з використання
WinAPI та підтримується розробником",
Application.ProductName, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);
    }
   }

   private void panel1_MouseDown(object sender, MouseEventArgs
e) {
    drag = true;
    startPoint = new Point(e.X, e.Y);
   }

   private void panel1_MouseMove(object sender, MouseEventArgs
e) {
    if (drag) {
     Point p = PointToScreen(e.Location);
     Location = new Point(p.X - startPoint.X, p.Y - startPoint.Y);
    }
   }

   private void panel1_MouseUp(object sender, MouseEventArgs e)
{
    drag = false;
   }

   private void label1_MouseDown(object sender, MouseEventArgs
e) {
    panel1_MouseDown(sender, e);
   }

   private void label1_MouseMove(object sender, MouseEventArgs
e) {
    panel1_MouseMove(sender, e);
   }

   private void label1_MouseUp(object sender, MouseEventArgs e) {
    drag = false;
   }
  }
}
```

Clean run

```
internal async void Run() {
   #region Перевірка чекбоксів
   bool yes = false;
   foreach(var item in Controls.OfType < CheckBox > ()) {
    if (item.Checked) {
     yes = true;
     break;
    }
   }
   if (!yes) {
    MessageBox.Show("Не обрано жодного пункту",
Application.ProductName, MessageBoxButtons.OK,
MessageBoxIcon.Error);
    return;
   }
   #endregion
   double size = 0;
   string path;
   Form1.mainForm.need = true;
   Form1.mainForm.Print();
   await Task.Delay(1000);
   if (checkBox1.Checked) {
    await Task.Delay(200);
    checkBox1.Checked = false;
    checkBox1.AutoCheck = false;
    if (Settings.Default.DarkTheme)
     checkBox1.ForeColor = Color.FromName("ControlDarkDark");
    else
     checkBox1.ForeColor = Color.FromName("Control");
   }
   if (checkBox2.Checked) {
    await Task.Delay(1000);
    path =
Environment.GetFolderPath(Environment.SpecialFolder.Windows) +
@ "\Temp";
    if (Directory.Exists(path))
     size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;

    path =
Environment.GetFolderPath(Environment.SpecialFolder.LocalAppli
cationData) + @ "\Temp";
    if (Directory.Exists(path))
     size += Cleaner.CleanerInFoldersTheFiles(path);
    Cleaner.realsize = 0;
    Cleaner.size = 0;
```

```
        checkBox2.Checked = false;
        checkBox2.AutoCheck = false;
        if (Settings.Default.DarkTheme)
          checkBox2.ForeColor = Color.FromName("ControlDarkDark");
        else
          checkBox2.ForeColor = Color.FromName("Control");
      }
      if (checkBox3.Checked) {
        path =
Environment.GetFolderPath(Environment.SpecialFolder.Windows) +
@ "\SoftwareDistribution\DataStore";
        if (Directory.Exists(path))
          size += Cleaner.CleanerInFoldersTheFiles(path);
        Cleaner.realsize = 0;
        Cleaner.size = 0;
        checkBox3.Checked = false;
        checkBox3.AutoCheck = false;
        if (Settings.Default.DarkTheme)
          checkBox3.ForeColor = Color.FromName("ControlDarkDark");
        else
          checkBox3.ForeColor = Color.FromName("Control");
      }
      if (checkBox4.Checked) {
        path =
Environment.GetFolderPath(Environment.SpecialFolder.Windows) +
@ "\SoftwareDistribution\Download";
        if (Directory.Exists(path))
          size += Cleaner.CleanerInFoldersTheFiles(path);
        Cleaner.realsize = 0;
        Cleaner.size = 0;
        checkBox4.Checked = false;
        checkBox4.AutoCheck = false;
        if (Settings.Default.DarkTheme)
          checkBox4.ForeColor = Color.FromName("ControlDarkDark");
        else
          checkBox4.ForeColor = Color.FromName("Control");
        await Task.Delay(500);
      }
      if (checkBox5.Checked) {
        string[] Drives = Environment.GetLogicalDrives();
        foreach(string s in Drives) {
          path = $ @ "{s}$RECYCLE.BIN";
          if (Directory.Exists(path) && Cleaner.FolderSize(path) > 129)
            size += Cleaner.CleanerInFoldersTheFiles(path);
          Cleaner.realsize = 0;
          Cleaner.size = 0;
        }
        checkBox5.Checked = false;
        checkBox5.AutoCheck = false;
        if (Settings.Default.DarkTheme)
          checkBox5.ForeColor = Color.FromName("Control");
        else
          checkBox5.ForeColor = Color.FromName("ControlDarkDark");
        await Task.Delay(500);
        Form1.Cmd("taskkill /F /IM explorer.exe & start explorer.exe");
      }
      if (checkBox6.Checked) {
        path =
Environment.GetFolderPath(Environment.SpecialFolder.UserProfile)
+ @ "\Downloads\Telegram Desktop";
        if (Directory.Exists(path))
          size += Cleaner.CleanerInFoldersTheFiles(path);
        Cleaner.realsize = 0;
        Cleaner.size = 0;

        path =
Environment.GetFolderPath(Environment.SpecialFolder.UserProfile)
+ @ "\Загрузки\Telegram Desktop";
        if (Directory.Exists(path))
          size += Cleaner.CleanerInFoldersTheFiles(path);
        Cleaner.realsize = 0;
        Cleaner.size = 0;

        if (Registry.CurrentUser.OpenSubKey(@
"SOFTWARE\Classes\Local
Settings\Software\Microsoft\Windows\Shell\MuiCache") != null) {
          foreach(var key in Registry.CurrentUser.OpenSubKey(@
"SOFTWARE\Classes\Local
Settings\Software\Microsoft\Windows\Shell\MuiCache")?.GetValue
Names()) {
            if (key.ToString().Contains("Telegram.exe")) {
              path = key.Substring(0, key.LastIndexOf(@"Telegram\") +
8).ToString();
              break;
            }
          }
        }

        if (Directory.Exists(path + "\\tdata")) {
          path += "\\tdata";
          DirectoryInfo directoryInfo = new DirectoryInfo(path);

          foreach(var item in directoryInfo.EnumerateDirectories()) {
            if (item.ToString().Contains("temp_data")) {
              size += Cleaner.CleanerInFoldersTheFiles(path + $
"\\{item}");
            }
            Cleaner.realsize = 0;
            Cleaner.size = 0;
            if (item.ToString().Contains("user_data")) {
              size += Cleaner.CleanerInFoldersTheFiles(path + $
"\\{item}");
            }
            Cleaner.realsize = 0;
            Cleaner.size = 0;
          }
        }

        checkBox6.Checked = false;
        checkBox6.AutoCheck = false;
        if (Settings.Default.DarkTheme)
          checkBox6.ForeColor =
Color.FromName("ControlDarkDark");
        else
          checkBox6.ForeColor = Color.FromName("Control");
        await Task.Delay(300);
      }
      if (checkBox7.Checked) {
        #region Cent Browser
        path = $ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\CentBrowser\User Data\Default\Cache\Cache_Data";
        if (Directory.Exists(path))
          size += Cleaner.CleanerInFoldersTheFiles(path);
        Cleaner.realsize = 0;
        Cleaner.size = 0;
        #endregion

        #region Chrome
        path = $ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Google\Chrome\User Data\Default\Cache";
        if (Directory.Exists(path))
          size += Cleaner.CleanerInFoldersTheFiles(path);
        Cleaner.realsize = 0;
        Cleaner.size = 0;
        #endregion

        #region Opera Gx
        path = $ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Opera Software\Opera GX
Stable\Cache\Cache_Data";
        if (Directory.Exists(path))
          size += Cleaner.CleanerInFoldersTheFiles(path);
        path = $ @
"{Environment.GetFolderPath(Environment.SpecialFolder.LocalApp
licationData)}\Opera Software\Opera GX Stable\System
Cache\Cache_Data";
        if (Directory.Exists(path))
          size += Cleaner.CleanerInFoldersTheFiles(path);
        Cleaner.realsize = 0;
        Cleaner.size = 0;
        #endregion

        checkBox7.Checked = false;
```

```
        checkBox7.AutoCheck = false;
        if (Settings.Default.DarkTheme)
          checkBox7.ForeColor =
Color.FromName("ControlDarkDark");
        else
          checkBox7.ForeColor = Color.FromName("Control");
      }
      if (checkBox8.Checked) {
        RamCleaner.ClearFileSystemCache();
        checkBox8.Checked = false;
        checkBox8.AutoCheck = false;
        if (Settings.Default.DarkTheme)
          checkBox8.ForeColor =
Color.FromName("ControlDarkDark");
        else
          checkBox8.ForeColor = Color.FromName("Control");
      }
      size = Convert.ToDouble(string.Format("{0:f1}", size / 1024 /
1024));
      await Task.Delay(1000);
      Form1.mainForm.need = false;
      MessageBox.Show($"Було очищено: {size} MB",
Application.ProductName, MessageBoxButtons.OK,
MessageBoxIcon.Asterisk);
      await Task.Delay(1000);
    }
```

Improve run

```
  #region Перевірка чекбоксів
  bool yes = false;
  foreach(var item in Controls.OfType < CheckBox > ()) {
    if (item.Checked) {
      yes = true;
      break;
    }
  }
  if (!yes) {
    MessageBox.Show("Не обрано жодного пункту",
Application.ProductName, MessageBoxButtons.OK,
MessageBoxIcon.Error);
    return;
  }
  #endregion
  if (checkBox3.Checked) {

Registry.LocalMachine.CreateSubKey("SOFTWARE\\Policies\\Micr
osoft\\Dsh").SetValue("AllowNewsAndInterests", 0,
RegistryValueKind.DWord);
    checkBox3.Checked = false;
    checkBox3.AutoCheck = false;
    button4.Visible = true;
    if (Settings.Default.DarkTheme)
      checkBox3.ForeColor = Color.FromName("ControlDarkDark");
    else
      checkBox3.ForeColor = Color.FromName("Control");
  }
  if (checkBox4.Checked) {

Registry.CurrentUser.CreateSubKey("Software\\Microsoft\\Window
s\\CurrentVersion\\BackgroundAccessApplications").SetValue("Glo
balUserDisabled", 1, RegistryValueKind.DWord);

Registry.CurrentUser.CreateSubKey("Software\\Microsoft\\Window
s\\CurrentVersion\\Search").SetValue("BackgroundAppGlobalToggl
e", 0, RegistryValueKind.DWord);
    checkBox4.Checked = false;
    checkBox4.AutoCheck = false;
    button5.Visible = true;
    if (Settings.Default.DarkTheme)
      checkBox4.ForeColor = Color.FromName("ControlDarkDark");
    else
      checkBox4.ForeColor = Color.FromName("Control");

  }
  if (checkBox5.Checked) {

Registry.LocalMachine.OpenSubKey("HKEY_LOCAL_MACHINE\
\SYSTEM\\CurrentControlSet\\Control\\Session Manager\\Memory
Management", true).SetValue("LargeSystemCache", 1);
  }
  }
```

Interface run

```
internal void Run() {
  if (checkBox3.Checked) {
    try {
      // Відкриваємо ключ реєстру, де зберігаються налаштування
ClearType
      RegistryKey key = Registry.CurrentUser.OpenSubKey(@
"Control Panel\Desktop", true);

      if (key != null) {
        // Встановлюємо значення параметра "FontSmoothing" на
"0" для відключення ClearType
        key.SetValue("FontSmoothing", "0",
RegistryValueKind.String);

        key.Close();

        if (Settings.Default.DarkTheme)
          checkBox3.ForeColor =
Color.FromName("ControlDarkDark");
        else
          checkBox3.ForeColor = Color.FromName("Control");
        checkBox1.Checked = false;
      } else {
        MessageBox.Show("Не вдалося відкрити ключ реєстру.");
      }
    } catch (Exception ex) {
      MessageBox.Show("Виникла помилка: " + ex.Message);
    }
  }
```

Autorun

```
private void Автозапуск_Load(object sender, EventArgs e) {
  try {
    foreach(var key in
Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Wind
ows\\CurrentVersion\\Explorer\\StartupApproved\\Run").GetValueN
ames()) {
      RegistryKey k =
Registry.CurrentUser.OpenSubKey($"SOFTWARE\\Microsoft\\Win
dows\\CurrentVersion\\Explorer\\StartupApproved\\Run", true);
      byte[] value = (byte[])
Registry.CurrentUser.OpenSubKey($"SOFTWARE\\Microsoft\\Win
dows\\CurrentVersion\\Explorer\\StartupApproved\\Run",
true).GetValue(key);

      CheckBox checkbox = new CheckBox();
      panel1.Controls.Add(checkbox);

      checkbox.Location = new Point(12, y);
      checkbox.Text = key.ToString();
      checkbox.ForeColor = Color.White;
      checkbox.Size = new Size(panel1.Height, 17);

      if (value[0] == 0x02)
        checkbox.Checked = true;
      else if (value[0] == 0x03)
        checkbox.Checked = false;
      else
        checkbox.Enabled = false;
      checkbox.Click += (s, a) => {
        if (Settings.Default.ProgramSounds)
          Sounds.PlaySound3();
        if (checkbox.Checked) {
          value[0] = 0x02;
          k.SetValue(key, value, RegistryValueKind.Binary);
        } else {
```

```
        value[0] = 0x03;
        k.SetValue(key, value, RegistryValueKind.Binary);
       }
      };
      y += 20;
     }
    } catch (Exception) {}

    foreach(var key in
Registry.LocalMachine.OpenSubKey("SOFTWARE\\Microsoft\\Wi
ndows\\CurrentVersion\\Explorer\\StartupApproved\\Run").GetValu
eNames()) {
     try {
      RegistryKey k =
Registry.LocalMachine.OpenSubKey($"SOFTWARE\\Microsoft\\W
indows\\CurrentVersion\\Explorer\\StartupApproved\\Run", true);
      byte[] value = (byte[])
Registry.LocalMachine.OpenSubKey($"SOFTWARE\\Microsoft\\W
indows\\CurrentVersion\\Explorer\\StartupApproved\\Run",
true).GetValue(key);

      CheckBox checkbox = new CheckBox();
      panel1.Controls.Add(checkbox);

      checkbox.Location = new Point(12, y);
      checkbox.Text = key.ToString();
      checkbox.ForeColor = Color.White;
      checkbox.Size = new Size(panel1.Height, 17);

      if (value[0] == 0x02)
       checkbox.Checked = true;
      else if (value[0] == 0x03)
       checkbox.Checked = false;
      else {
       checkbox.AutoCheck = false;
       checkbox.ForeColor = Color.FromName("ControlDarkDark");
      }

      checkbox.Click += (s, a) => {
       if (Settings.Default.ProgramSounds)
        Sounds.PlaySound3();
       if (checkbox.Checked) {
        value[0] = 0x02;
        k.SetValue(key, value, RegistryValueKind.Binary);
       } else {
        value[0] = 0x03;
        k.SetValue(key, value, RegistryValueKind.Binary);
       }
      };
      y += 20;
     } catch (Exception) {}
    }
}
```

Settings items

```
public partial class Form_Settings: Form {
 public Form_Settings() {
  InitializeComponent();
 }

 private void checkBox1_Click(object sender, EventArgs e) {
  Sounds.PlaySound3();
  if (checkBox1.Checked)
   Properties.Settings.Default.SystemNotification = true;
  else
   Properties.Settings.Default.SystemNotification = false;
  Properties.Settings.Default.Save();
 }

 private void checkBox2_Click(object sender, EventArgs e) {
  if (checkBox2.Checked)
   Properties.Settings.Default.ProgramSounds = true;
  else
   Properties.Settings.Default.ProgramSounds = false;
  Properties.Settings.Default.Save();
 }

 private void checkBox3_Click(object sender, EventArgs e) {
  if (checkBox3.Checked) {
   Properties.Settings.Default.AutoUpdate = true;
  } else {
   Properties.Settings.Default.AutoUpdate = false;
  }
  Properties.Settings.Default.Save();
 }

 private void checkBox4_Click(object sender, EventArgs e) {
  if (checkBox4.Checked)
   Properties.Settings.Default.Animation = true;
  else
   Properties.Settings.Default.Animation = false;
  Properties.Settings.Default.Save();
 }

 async internal void button1_Click(object sender, EventArgs e) {
  await new Internet().UpdateChecker();
 }
}
```

System info run

```
internal async void Run() {
 await Task.Run(async () => {
  if (label1.Text == "") {
   try {
    string processor = "", coreCounts = "", friquencyProc = "",
videoAdapters = "", motherboard = "";
    ulong memory = 0;

    #region Проц
    foreach(var item in new
ManagementObjectSearcher("root\\cimv2", "select * from
win32_processor").Get()) {
     processor = item.GetPropertyValue("Name").ToString();
     coreCounts =
item.GetPropertyValue("NumberOfCores").ToString();
     friquencyProc =
item.GetPropertyValue("MaxClockSpeed").ToString();
    }
    #endregion

    #region Материнська плата
    foreach(var item in new
ManagementObjectSearcher("root\\cimv2", "select * from
win32_baseboard").Get())
    motherboard +=
item.GetPropertyValue("Manufacturer").ToString() + " " +
item.GetPropertyValue("Product").ToString();
    #endregion

    #region ОЗУ
    foreach(var item in new
ManagementObjectSearcher("root\\cimv2", "select * from
win32_physicalmemory").Get())
    memory +=
Convert.ToUInt64(item.GetPropertyValue("Capacity")) / 1024 /
1024;
    #endregion

    #region Відеокарта
    foreach(var item in new
ManagementObjectSearcher("root\\cimv2", "select * from
win32_videocontroller").Get())
    if (videoAdapters.Length == 0 &&
item.GetPropertyValue("Name").ToString().Remove(5) == "Intel")
     videoAdapters += item.GetPropertyValue("Name").ToString();

    foreach(var item in new
ManagementObjectSearcher("root\\cimv2", "select * from
win32_videocontroller").Get())
    if (item.GetPropertyValue("Name").ToString().Remove(5) !=
"Intel")
     if (videoAdapters.Length == 0)
```

```
            videoAdapters +=
item.GetPropertyValue("Name").ToString();
          else
            videoAdapters += "\n" +
item.GetPropertyValue("Name").ToString();

        #endregion

        string OSBitDepth = "";
        if (Environment.Is64BitOperatingSystem)
          OSBitDepth = "64-х розрядна";
        else
          OSBitDepth = "32-х розрядна";
        label1.Text = $ "ОС: Windows {Settings.Default.WinVersion}
" +
          //"ОС: " +
Registry.LocalMachine.OpenSubKey(@"SOFTWARE\Microsoft\Wi
ndows NT\CurrentVersion").GetValue("ProductName").ToString() +
" " +
          Registry.LocalMachine.OpenSubKey(@
"SOFTWARE\Microsoft\Windows
NT\CurrentVersion").GetValue("DisplayVersion").ToString() + " " +
OSBitDepth + Environment.NewLine +
          "Ім'я комп'ютера: " + Environment.MachineName +
Environment.NewLine +
          $ "Процесор: {processor}" + Environment.NewLine +
          "Частота процессора: " + Convert.ToInt32(friquencyProc) /
1000 + ".00GHz" + Environment.NewLine +
          "Кількість ядер процесору: " + coreCounts +
Environment.NewLine +
          "Кількість потоків процесора: " +
Environment.ProcessorCount + Environment.NewLine +
          $ "Материнська плата: {motherboard}" +
Environment.NewLine +
          "ОЗУ: " + memory + " MB" + Environment.NewLine +
          $ "Відеокарта: {videoAdapters}";
      } catch (Exception) {
        MessageBox.Show("Інструмент WMI відсутній на ПК");
      }
    }
    await Task.Delay(500);
    while (Form1.mainForm.active.Text == "SystemInfo") {
      foreach(var item in new
ManagementObjectSearcher("root\\cimv2", "select * from
win32_processor").Get()) {
        label2.Text = $ "Поточна тактова частота процесора:
{item.GetPropertyValue("
        CurrentClockSpeed")} MHz";
      }
      foreach(var item in new
ManagementObjectSearcher("root\\cimv2", "select * from
win32_operatingSystem").Get()) {
        label3.Text = $ "Вільне місце ОЗП:
{Convert.ToUInt64(item.GetPropertyValue("
        FreePhysicalMemory")) / 1024} Мб";
      }
      await Task.Delay(1000);
    }
  });
}
```