Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Застосунок для моніторингу комп’ютерної системи

(тема)

Виконав:
здобувач 3 року навчання,
групи КІУКІу-22-2
Ілля КОНДРАТЮК
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. Тетяна ФІЛІМОНЧУК
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Кондратюку Іллі Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи Застосунок для моніторингу комп’ютерної системи

затверджена наказом по університету від “ 25 ” травня 2025 р. № 425 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025 р.

3. Вхідні дані до роботи

1. Операційна система: Windows
2. Мова програмування: C#
3. Інструменти моніторингу: Windows Management Instrumentation (WMI)
4. Інтерфейс користувача: Windows Forms
5. Середовище розробки: Microsoft Visual Studio
6. Тип ліцензії: Freeware

4. Перелік питань, що потрібно опрацювати у роботі

1. Аналіз предметної області та постановка задачі
2. Розробка структури застосунку
3. Опис програми
4. Тестування програмного додатку
5. Технічні вимоги та інструкція користувача

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 10 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз існуючих методів вирішення задачі	10.06.2025-13.06.25	
2	Вибір програмного забезпечення та	14.06.2025-17.06.25	
	інструментів розробки		
3	Проєктування архітектури застосунку	18.06.25-21.06.25	
4	Розробка логіки застосунку	22.06.25-28.06.25	
5	Розробка графічного інтерфейсу користувача	29.06.25-02.07.25	
6	Тестування застосунку	03.07.25-05.07.25	
7	Подання кваліфікаційної роботи керівникам	06.07.25-09.07.25	
	для попереднього захисту		
8	Подання кваліфікаційної роботи на	10.07.25-11.07.25	
	рецензування		

Дата видачі завдання " 09 " червня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Тетяна ФІЛІМОНЧУК
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 75 с., 30 рис., 2 табл., 1 дод., 17 джерел.

ДИСК, WMI, C#, WINDOWS FORMS, ГРАФІЧНИЙ ІНТЕРФЕЙС, ВІКОННИЙ ДОДАТОК, ДАНІ, ЧИТАННЯ, ДАТЧИКИ, ДИСПЕТЧЕР, МОНІТОРИНГ, БЕНЧМАРК, РЕСУРСИ.

Метою кваліфікаційної роботи є розробка віконної програми, яка здатна отримувати системну інформацію, відображати дані про компоненти системи, їх параметри, та стан, рівень навантаження та використання ресурсів, а також створення спеціальних тестів для оцінки якості роботи системи.

У ході виконання кваліфікаційної роботи було створено програму "HARDINFO" для перегляду характеристик комп’ютера та оцінки його продуктивності. Було проаналізовано популярні утиліти для моніторингу, такі як AIDA64, HWMonitor, CPU-Z, Speccy та MSI Afterburner. Програму було розроблено з графічним інтерфейсом на основі Windows Forms, з використанням мови C# у середовищі Visual Studio. Щоб отримувати інформацію про компоненти комп’ютера, використовувалися інструменти WMI, PowerShell та спеціальні бібліотеки. Застосунок дозволяє зручно переглядати відомості про основні компоненти системи: процесор, оперативну пам’ять, диски, мережеві адаптери та інше.

ABSTRACT

Bachelor’s thesis: 75 pages, 30 figures, 2 tables, 1 appendices, 17 sources.

DISK, WMI, C#, WINDOWS FORMS, GRAPHICAL INTERFACE, WINDOWS APPLICATION, DATA, READING, SENSORS, DISPATCHER, MONITORING, BENCHMARK, RESOURCES.

The major goal of this thesis is to develop a window program that is capable of receiving system information, displaying data about system components, their parameters, and status, load level and resource usage, as well as creating special tests to assess the quality of system operation.

During the work, the "HARDINFO" program was created to view the characteristics of the computer and evaluate its performance. Popular monitoring utilities such as AIDA64, HWMonitor, CPU-Z, Speccy and MSI Afterburner were analyzed. The program was developed with a graphical interface based on Windows Forms, using the C# language in the Visual Studio environment. To obtain information about the components of the computer, WMI, PowerShell and special libraries were used. The application allows you to conveniently view information about the main components of the system: processor, RAM, disks, network adapters and more.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ANSI – організація зі стандартизації (англ., American National Standards Institute)

BIOS – базова система введення-виведення (англ., Basic Input/Output System)

CMD– класична командна оболонка Windows (англ., Command Prompt)

CPU – центральний процесор (англ., Central Processing Unit)

.NET – програмна платформа Microsoft для створення застосунків

Git – система контролю версій

GPU – графічний процесор (англ., Graphics Processing Unit)

GUI – графічний інтерфейс користувача (англ., Graphical User Interface)

HDD/SSD – жорсткий диск / твердотільний накопичувач (англ., Hard Disk Drive / Solid State Drive)

OEM – виробник оригінального обладнання (англ., Original Equipment Manufacturer)

OS – операційна система (англ., Operating System)

OSD – виведення інформації на екран (англ., On-Screen Display)

RAM – оперативна пам’ять (англ., Random Access Memory)

SMART – самодіагностика накопичувачів (англ., Self-Monitoring, Analysis and Reporting Technology)

SPD – технологія зчитування з модулів оперативної пам’яті (англ., Serial Presence Detect)

UML – уніфікована мова моделювання (англ., Unified Modeling Language)

WMI – інтерфейс для доступу до системної інформації (англ., Windows Management Instrumentation)

WQL – мова запитів до WMI (англ., WMI Query Language)

ВСТУП

На сьогоднішній день індустрія програмного забезпечення є однією з найбільш прибуткових та стрімко ростучих галузей. З кожним днем кількість користувачів комп’ютерів стрімко зростає, багатьом з них потрібно отримувати інформацію про своє залізо та якість його роботи. Це призводить до появи "аналітичного" програмного забезпечення. Сучасні програми вражають кількістю функцій для оптимізації та моніторингу, але через те що більшість з них є комерційними проєктами, вони вимагають або купівлі ліцензійних ключів, або платної підписки для розблокування необхідних функцій, або просто доступу до програмного забезпечення [1].

Існує безліч різноманітних програм, і кожна з них має свій унікальний підхід до створення. Невеликі проєкти можуть бути написані будь-яким зацікавленим користувачем, оскільки вони не потребують великої кількості знань та навичок але для створення складних великих проєктів потрібні глибокі знання та досвід, а також участь більшої кількості людей. Якісні програми можуть мати багато корисних функцій, а не тільки займатись моніторингом та тестувати залізо, виправляти помилки. Для створення таких програм часто використовують мови програмування C# або C++. Крім того, окремі модулі можуть бути написані на інших мовах програмування. Важливо зазначити, що вибір програмної мови залежить від вимог конкретної області, вартості розробки та комп'ютерного обладнання на роботу з яким направлене програмне забезпечення.

Комп'ютери, які зараз використуються, пройшли довгий шлях розвитку. Вони стали такими потужними та складними завдяки розвитку технологій. В сучасного комп'ютера аналогова складова та кількість служб настільки різноманітна, що іноді важко уявити всі можливості, які він надає. Комп'ютери використовуються в різних сферах: науці, освіті, металургійній промисловості, медицині і, звичайно ж, у сфері розваг. Тому оцінка їх стану

якості та ефективності роботи є дуже важливою. Без моніторингового програмного забезпечення дізнатись щось важливе про стан комп’ютера було б неможливо.

Метою даної кваліфікаційної роботи є розробка структури програмного забезпечення, орієнтованого на тестування продуктивності комп'ютера та аналізу встановлених комплектуючих. Головна ідея цього програмного забезпечення полягає в тому, що програма буде здатна відображати для користувача інформацію майже про кожен елемент в його комп’ютері, від процесора до клавіатури.

Під час розробки додатку "HARDINFO" була використані інструменти WMI, Power Shell та Windows Forms в середовищі Microsoft Visual Studio. Ці інструменти було обрано для створення застосунку під операційну систему Windows з заданою функціональністю.

1 АНАЛІЗ ПРЕДМЕТОЇ ОБЛАСТІ ТА ІСНУЮЧИХ АНАЛОГІВ ДЛЯ МОНІТОРИНГУ КОМП’ЮТЕРНОЇ СИСТЕМИ

В наш час комп'ютерні системи є невід'ємною частиною повсякденного життя, як у професійній сфері, так і в повсякденному житті. Тому для користувачів є важливими ефективність та якість роботи системи, що залежать від стабільності та оптимізації навантаження, яких можливо досягти лише за умови точного моніторингу ключових показників, таких як температури обладнання (особливо процесорів), ступені їх навантаження, а також рівня використання оперативної пам'яті. Для цього користувачі використовують спеціалізовані програми, які забезпечують можливість постійного моніторингу стану компонентів комп'ютера та допомагають своєчасно виявити можливі проблеми.

На даний час існує багато різного програмного забезпечення для моніторингу комп’ютерних систем, серед таких програм найпопулярнішими є AIDA64, HWMonitor, CPU-Z, Speccy та MSI Afterburner.

1.1 Програмне забезпечення для діагностики та тестування AIDA64

AIDA64 – це комп’ютерна програма для детального аналізу комп’ютерної системи, діагностики та моніторингу апаратного забезпечення ПК. Її широко використовують для моніторингу стану комп'ютерної системи, виявлення апаратних неполадок, аналізу продуктивності, перевірки сумісності системи, тестування стабільності, а також для розгону обладнання, хоча це навпаки може нашкодити системі. Програма була випущена компанією FinalWire Ltd, яка є провідним розробником програмного забезпечення для діагностики [2].

AIDA64 сумісна з різними версіями ОС Windows, підтримує як 32-розрядні, так і 64-розрядні архітектури. Інтерфейс програми (рисунок 1.1)

інтуїтивно зрозумілий, тому навіть недосвідчений користувач зможе легко знайти необхідні розділи та без зайвих маніпуляцій виконати перевірки стану компонентів системи.

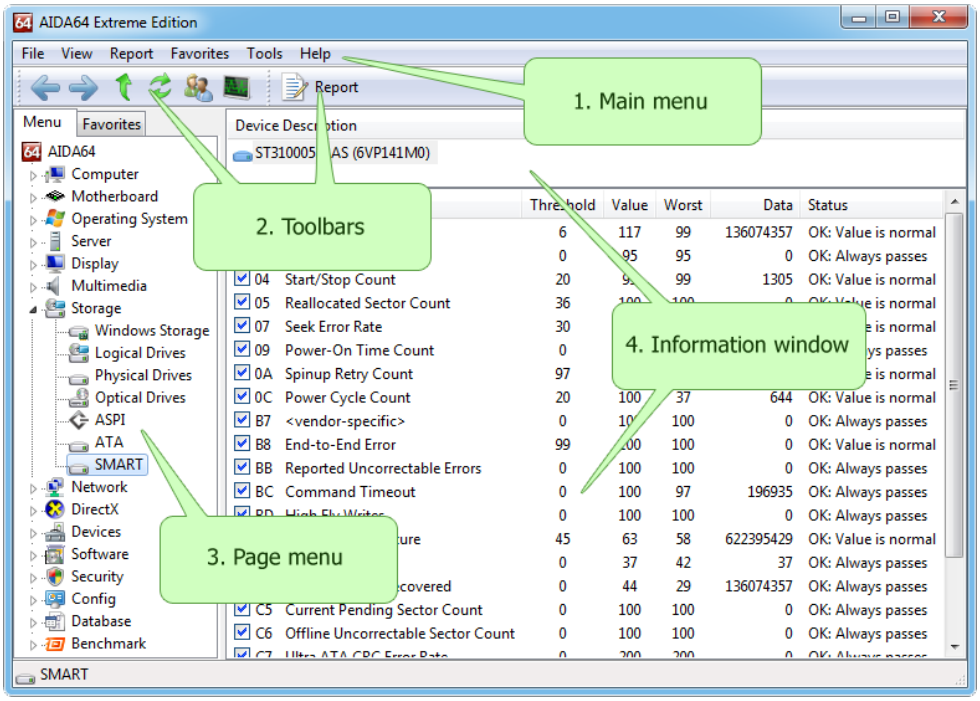

Рисунок 1.1 – Інтерфейс AIDA64

Серед великої кількості функцій AIDA64 можливо виділити наступні:

- діагностика системи: програма надає розширену інформацію про всі компоненти ПК, включно з процесором, оперативною пам'яттю, жорсткими дисками, мережними адаптерами, а також перелік встановлених драйверів;

- моніторинг температури та напруги: AIDA64 може в реальному часі моніторити температуру різних компонентів системи: центрального процесора, графічного процесора, жорстких дисків, материнської плати та інших компонентів, що дозволяє визначити, які компоненти перегріваються, що свідчить про їх перенавантаження;

- тестування продуктивності: у AIDA64 є набір тестів для перевірки продуктивності компонентів системи, що дозволяє порівняти продуктивність системи з еталонними показниками;

- мережна діагностика: AIDA64 включає функції для моніторингу та

тестування мережного з'єднання, відображення списку підключених пристроїв, аналізу пропускної здатності мережі, що може стати в нагоді деяким користувачам, наприклажд мережним адміністраторам.

Окремо варто згадати інструментарій доступний для досвідчених користувачів: у AIDA64 є ряд додаткових функцій, таких як підтримка дистанційного моніторингу, виявлення несправностей у корпоративних мережах, складання докладних звітів, а також можливість налаштовувати систему для роботи в стані максимального навантаженням.

Серед переваг програмного забезпечення AIDA64 слід виділити такі:

- великий обсяг інформації про систему: програма відображає велику кількість інформації про кожен компонент в системі, а також про програмне забезпечення для цих компонентів;

- зручний інтерфейс: інтерфейс програми організований таким чином, щоб користувач міг швидко знайти потрібну інформацію, але це одночасно є і недоліком, бо користувачу надається вся інформація одразу;

- широкий вибір тестів для продуктивності: AIDA64 пропонує різні стрес-тести, для перевірки стабільності роботи системи під великими навантаженнями, а також тести для продуктивності процессорів та пам’яті;

- сумісність з усіма версіями Windows: AIDA64 працює з усіма новими, а також підтримує старі версії Windows, що робить її універсальним інструментом для моніторингу незалежно від версії ОС Windows;

- кількість налаштувань: користувач може налаштовувати появу повідомленнь про перегрів, перенавантаження або просто стану завантаження системи чи компонента, задавати частоту оновлення інформації, контролювати рівень деталізації даних;

- підтримка віддаленого моніторингу: AIDA64 дає можливість проводити моніторинг системи на віддалених комп’ютерах, що корисно для мережних адміністраторів та компаній з великою кількістю обладнання;

- інтеграція з сенсорами для моніторингу: програма підтримує велику кількість сенсорів для моніторингу температури, вентиляторів та інших

параметрів, що дозволяє краще контролювати стан обладнання.

Основними недоліками AIDA64 є:

- платна ліцензія: хоча програма пропонує пробну версію, для доступу до всіх функцій потрібно придбати ліцензію, вартість якої може бути зависокою для простого користувача (рисунок 1.2);

- високий рівень використання системних ресурсів під час тестування: AIDA64 може сильно навантажувати систему під час стрес-тестів, що становить загрозу для старих чи слабких компонентів системи, особливо при в руках недосвідченого користувача;

- відсутність підтримки інших ОС: AIDA64 обмежується тільки ОС Windows, використання в ОС, таких як macOS або Linux, неможлива;

- складність у використанні для новачків: хоча інтерфейс AIDA64 є простим та зручним, і орієнтуватись в ньому легко, але в програмі відсутнє ранжування функцій, тому прості функції знаходяться в тому ж списку, що й функції для досвідчених користувачів;

- можливі проблеми з сумісністю нових сенсорів: оскільки технології швидко розвиваються, інколи в AIDA64 виникають затримки у підтримці нових сенсорів та обладнання.

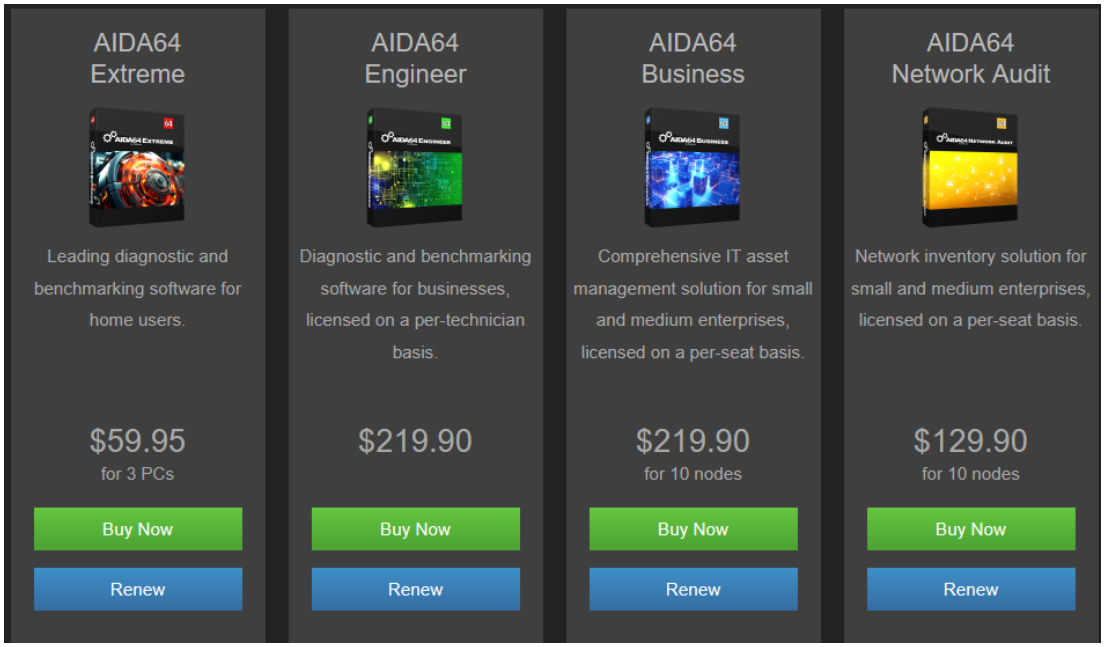

Рисунок 1.2 – Вартість ліцензій AIDA64

1.2 Програмне забезпечення для моніторингу продуктивності та температури компонентів комп'ютера HWMonitor

HWMonitor – це програмне забезпечення, яке було розроблене компанією CPUID для моніторингу апаратних компонентів комп’ютера в реальному часі. Програма забезпечує користувачів інструментарієм для відстеження температури, рівня напруги та швидкості обертання вентиляторів, що є ключовими показниками стабільної роботи системи.

Завдяки HWMonitor користувачі можуть контролювати стан обладнання та вчасно виявляти можливі проблеми з охолодженням або живленням. Ця програма популярна як серед звичайних користувачів, так і серед геймерів або системних адміністраторів завдяки простому та інтуїтивному інтерфейсу (рисунок 1.3) й високою якістю подання інформації.

HWMonitor надає користувачу ефективні засоби моніторингу, які допомагають запобігти перегріванню компонентів та попереджують користувача про можливі перебої у роботі системи комп’ютера [3].

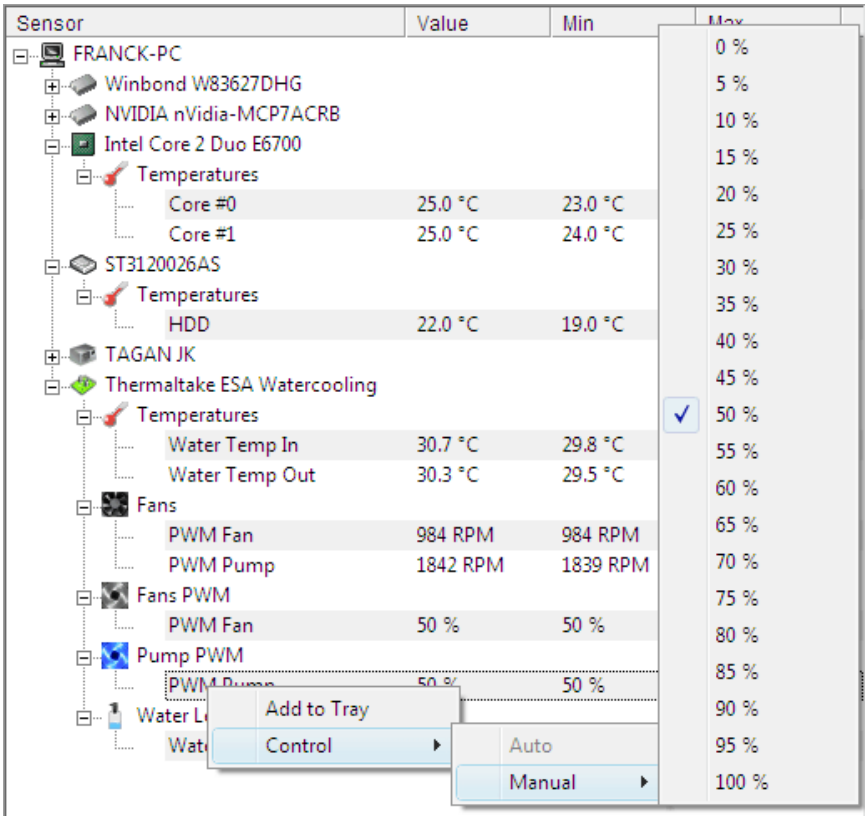

Рисунок 1.3 – Інтерфейс HWMonitor

Основні функції HWMonitor:

- моніторинг температури компонентів: програма надає користувачу дані про температуру процесора, графічного процесора, жорстких дисків, материнської плати і за можливості інформацію про корпусні вентилятори;

- відображення напруги: HWMonitor показує інформацію про напругу на процесорі, оперативній пам'яті, відеокарті, що дозволяє користувачам вчасно виявляти проблеми з живленням компонентів;

- швидкість обертання вентиляторів: програма вимірює швидкість обертання вентиляторів у корпусі, CPU та GPU, що дозволяє визначити, чи працюють вентилятори коректно, чи потрібне їхнє додаткове налаштування;

- моніторинг акумулятора (для ноутбуків): HWMonitor дозволяє відстежувати стан батареї ноутбука, зокрема рівень заряду, напругу та температуру батареї, що є корисною функцією для власників ноутбуків;

- підтримка різних типів датчиків: HWMonitor може працювати з різноманітними сенсорами, підтримуючи як вбудовані в систему сенсори, так і сенсори сторонніх пристроїв, що забезпечує точність даних на різних пристроях.

Перевагами у використанні даного програмного забезпечення є:

- простота використання: HWMonitor має інтуїтивно зрозумілий інтерфейс, що дозволяє користувачам швидко знаходити необхідну інформацію без зайвих пошуків чи додаткових налаштувань але він виглядає застарілим;

- висока точність даних: програма надає користувачу точні показники температури, рівня напруги та швидкості вентиляторів, що дозволяє використовувати її як надійний інструмент для контролю стану обладнання;

- малий розмір та низьке споживання ресурсів: HWMonitor не вимагає багато місця на жорсткому диску та споживає мінімум системних ресурсів, тому може працювати у фоновому режимі без впливу на продуктивність ПК;

- безкоштовна базова версія: HWMonitor доступна безкоштовно і має усі необхідні функції для моніторингу, тому не має необхідності купувати

ліцензію для домашнього використання, що робить програму привабливою для широкого кола користувачів;

- можливість відстеження на віддалених системах (в платній версії): HWMonitor Pro (рисунок 1.4) підтримує віддалений моніторинг, що є корисним для адміністраторів, які бажають контролювати параметри кількох систем одночасно;

- підтримка широкого спектру сенсорів та материнських плат: програма сумісна з більшістю сучасних материнських плат та систем охолоджування, що забезпечує гнучкість її використання на різних пристроях;

- інтеграція з іншими програмами CPUID: HWMonitor легко інтегрується з іншими програмами компанії CPUID, наприклад CPU-Z, що дозволяє користувачам отримати комплексну інформацію про систему з кількох джерел.

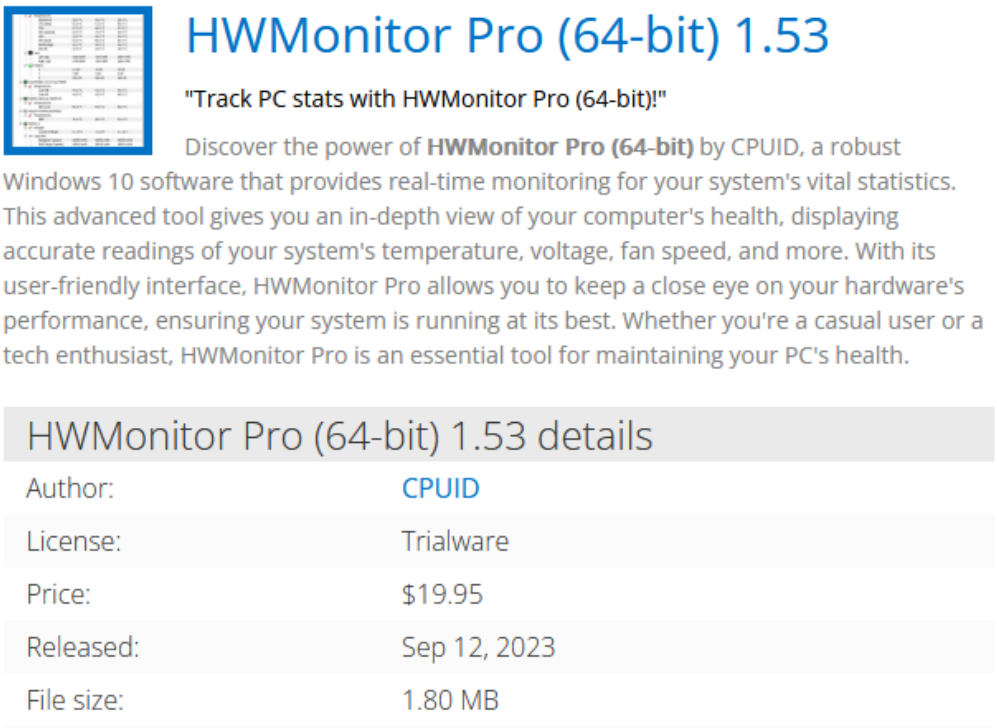

Рисунок 1.4 – Вартість повної версії HWMonitor

HWMonitor має низку обмежень та недоліків, які можуть вплинути на її зручність функціональність для користувачів, серед яких можна виділити:

- обмежені можливості в базовій версії: безкоштовна версія HWMonitor

має певні обмеження, зокрема відсутність функцій для віддаленого моніторингу, що доступні лише у платній версії HWMonitor Pro;

- відсутність тестування продуктивності: HWMonitor не включає тести продуктивності або стрес-тести, що може бути недоліком для тих, хто хоче оцінити стабільність системи під навантаженням;

- можливі проблеми з сумісністю деяких датчиків: деякі користувачі у відгуках про дане програмне забезпечення повідомляють про проблеми із сумісністю HWMonitor з окремими материнськими платами або компонентами, що інколи призводить до неправильного відображення інформації;

- сумісність тільки з ОС Windows: програма розроблена для використання лише на операційних системах Windows, що обмежує її застосування на інших платформах, таких як macOS або Linux;

- відсутність інструментів для налаштування вентиляторів: HWMonitor не дозволяє керувати швидкістю вентиляторів, а лише відображає поточні значення, що може бути незручним для тих, хто хоче вручну налаштувати параметри охолодження;

- інтерфейс без графіків змін параметрів у часі: програма не підтримує функцію побудови графіків, що ускладнює відстеження змін у динаміці параметрів, що може бути незручним для користувачів, які хочуть бачити історію змін у температурі чи напрузі;

- відсутня можливість налаштування сповіщень: HWMonitor не підтримує налаштування сповіщень чи попереджень при досягненні критичних значень температури чи напруги, що може бути важливим для користувачів, які потребують автоматичних повідомлень у випадку потенційних проблем;

- оновлення та підтримка: HWMonitor оновлюється не так часто, як деякі інші програми для моніторингу, що часто призводить до тимчасових проблем з новими компонентами.

1.3 Програмне забезпечення для відображення технічної інформації про вузли комп'ютера CPU-Z

CPU-Z – це популярна програма для моніторингу, яка надає користувачам детальну інформацію про характеристики та стан компонентів комп’ютерної системи, таких як: процесор, оперативна пам’ять, материнська плата та відеокарта. Програма була розроблена компанією CPUID, яка також створила HWMonitor. CPU-Z є одним із найпопулярніших інструментів для діагностики апаратного забезпечення та підходить як для звичайних користувачів, які цікавляться специфікаціями своїх систем, так і для професіоналів, для яких важливою складовою є наявність інструментів тестування та порівняння продуктивності [4]. Завдяки простому інтерфейсу (рисунок 1.5) і якості подання даних, CPU-Z дозволяє швидко отримати повну та зрозумілу картину стану основних апаратних компонентів. Цей інструмент особливо корисний для тих, хто хоче перевірити параметри системи при розгоні, діагностувати несправності або просто отримати повну інформацію про поточні компоненти ПК.

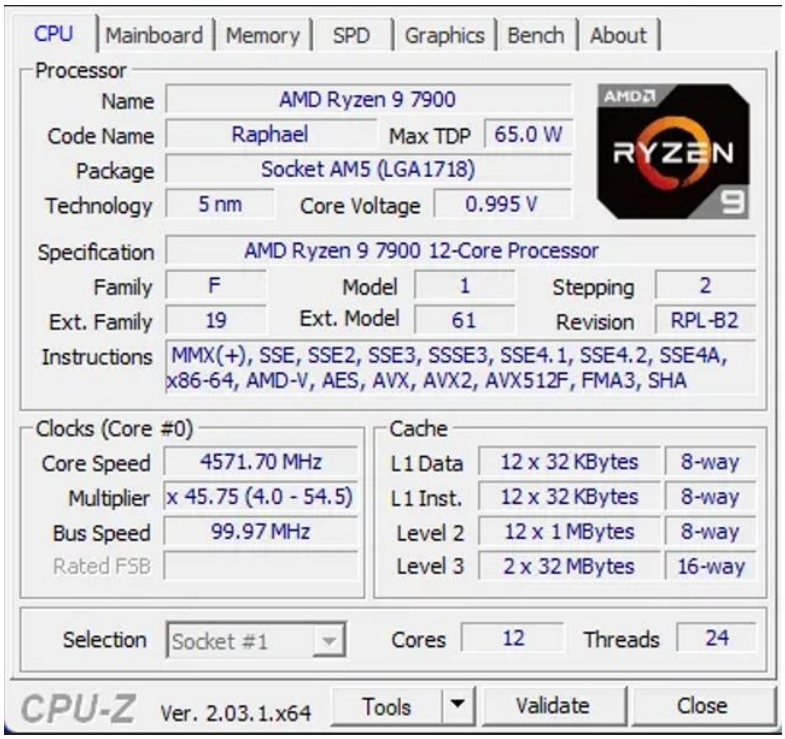

Рисунок 1.5 – Інтерфейс CPU-Z

Основні функції CPU-Z:

- інформація про процесор: CPU-Z відображає повну інформацію про процесор, включаючи модель, архітектуру, кількість ядер та потоків, частоту, множник та напругу. Також програма надає дані про рівні кешу процесора (L1, L2, L3), що може бути корисно для аналізу загальної швидкодії та продуктивності в різних ситуаціях;

- інформація про материнську плату: CPU-Z надає такі відомості, як виробник, модель, чипсет та версія BIOS, що дозволить користувачу оцінити сумісність плати з іншими компонентами та дізнатися, чи потрібне оновлення BIOS для підтримки нового обладнання або покращення стабільності системи;

- інформація про оперативну пам'ять: CPU-Z надає інформацію про тип оперативної пам'яті (DDR3, DDR4), частоту, кількість модулів та їх обсяг, а також таймінги та затримку, що дозволить корисувачу зрозуміти ефективність та конфігурацію оперативної пам'яті;

- інформація про графічний процесор (GPU): програма показує основні характеристики графічної карти, зокрема модель, частоту ядра та обсяг пам'яті, що особливо корисно для геймерів та розробників графічного контенту;

- функція "Вкладка SPD": CPU-Z надає детальну інформацію про окремі модулі пам'яті через вкладку SPD (Serial Presence Detect), де можна дізнатись про виробника, серійний номер, номінальну частоту та режими пам'яті, що рідтримуються;

- тести продуктивності: CPU-Z включає функцію тестування, яка дозволяє порівняти продуктивність процесора з іншими популярними моделями, що дає змогу користувачам визначити, наскільки ефективно працює їхній процесор у порівнянні з іншими конфігураціями. Крім того, ця функція дозволяє виявити можливі проблеми з продуктивністю або підтвердити ефективність проведеного розгону, надаючи точні дані про швидкодію процесора під різним навантаженням.

Серед переваг даного програмного забезпечення можна виділити такі:

- надання детальної інформації про процесор: CPU-Z надає вичерпні дані про центральний процесор, що робить його ідеальним інструментом для тих, хто хоче дізнатись про характеристики та реальні параметри процесора у ПК;

- точна інформація про оперативну пам'ять: програма детально описує всі аспекти оперативної пам'яті, включаючи її конфігурацію та таймінги, що дозволяє користувачам оптимізувати пам'ять для максимального рівня продуктивності;

- зручність у використанні: CPU-Z має простий інтерфейс і не вимагає спеціальних знань для користувача, що дозволяє швидко знайти потрібну інформацію;

- вільне розповсюдження: CPU-Z абсолютно безкоштовна програма, тому вона є доступною для усіх користувачів, які не готові витрачати кошти на моніторинг програмного забезпечення;

- невеликий розмір та низькі вимоги до ресурсів: CPU-Z займає мало місця на диску та не навантажує систему, що дозволяє використовувати її навіть на пристроях із обмеженими ресурсами;

- підтримка різних процесорів та материнських плат: CPU-Z сумісна з більшістю сучасних процесорів Intel та AMD, а також материнських плат, що забезпечує її універсальність та застосовність на різних системах;

- інструменти для порівняння продуктивності: CPU-Z дозволяє оцінити продуктивність процесора та оперативної пам'яті, що корисно для користувачів, які хочуть порівняти свій процесор із іншими або дізнатись про його ефективність при розгоні;

- регулярні оновлення та підтримка нових компонентів: CPU-Z регулярно оновлюється для підтримки нових моделей процесорів, оперативної пам’яті та материнських плат, що робить його надійним інструментом для моніторингу навіть на найсучасніших системах.

Хоча CPU-Z є потужним та зручним інструментом для моніторингу

апаратного забезпечення, програма має і свої недоліки, які можуть обмежити її застосування для певних користувачів. Серед основних недоліків можна відзначити такі:

- обмежена підтримка тестування графічного процесора: програма не включає детальних тестів графічної карти або температурного моніторингу GPU, що обмежує її можливості для геймерів та дизайнерів, які працюють із важкими графічними навантаженнями;

- відсутність функцій моніторингу температури: CPU-Z не показує температуру процесора або інших компонентів, що є значним недоліком у порівнянні з іншими програмами для моніторингу системи;

- відсутність налаштувань для вентиляторів: CPU-Z не дозволяє користувачам налаштовувати або моніторити швидкість вентиляторів, що може бути важливим для забезпечення оптимального охолодження системи;

- працює лише на ОС Windows: програма доступна тільки для операційної системи Windows, що обмежує її використання на інших платформах, таких як macOS та Linux;

- не підтримує побудову графіків: CPU-Z відображає дані в реальному часі без можливості створювати графіки змін параметрів у часі, що є незручним для тих, хто хоче аналізувати динаміку змін показників;

- сумісність із специфічними типами пам'яті: деякі старі або спеціальні типи пам'яті можуть бути визначені некоректно або неповно, що може призвести до неточних даних для рідкісних конфігурацій;

- відсутність інструментів для діагностики несправностей: CPU-Z не надає можливостей для виявлення несправностей або тестування стабільності компонентів, що може бути обмеженням для користувачів, які шукають комплексний діагностичний інструмент;

- обмежена функціональність в порівнянні з аналогами: хоча CPU-Z надає основні дані про апаратні компоненти, він не включає розширені можливості для аналізу та моніторингу, як деякі інші програми.

1.4 Пропрієтарна утиліта Speccy

Speccy – це безкоштовна програма для моніторингу системи, розроблена компанією Piriform, яка надає користувачам детальну інформацію про апаратне забезпечення комп'ютера. Speccy надає можливість для користувача швидко та точно аналізувати системні компоненти комп’ютерної системи, такі як процесор, материнська плата, оперативна пам'ять, графічна карта, жорсткі диски та інше. Ця програма особливо корисна для тих, хто цікавиться апгрейдом системи, розгоном або просто хоче дізнатися більше про компоненти свого ПК [5].

Speccy відрізняється від інших подібних утиліт своєю простотою використання та естетичним дизайном (рисунок 1.6). Користувачі можуть швидко знайти необхідну інформацію без зайвих труднощів. Крім того, Speccy пропонує можливість зберігати та ділитися звітами про систему, що робить її ідеальною для обміну інформацією про систему з друзями або технічними фахівцями.

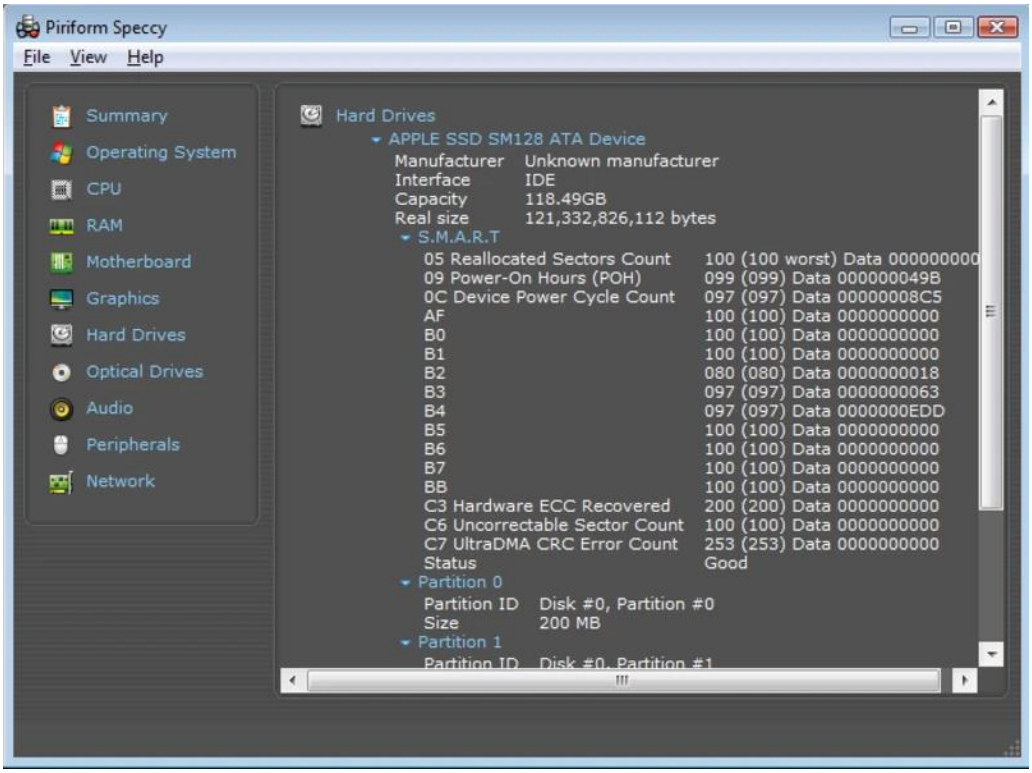

Рисунок 1.6 – Інтерфейс Speccy

Основними функціями даного програмного забезпечення є:

- загальний аналіз комп’ютерної системи: Speccy надає вичерпні дані про компоненти комп'ютера, включаючи інформацію про процесор, материнську плату, оперативну пам'ять, графічний процесор та накопичувачі, що дає користувачу можливість мати повну картину характеристик ПК;

- моніторинг температури: Speccy відображає температуру процесора, графічної карти, жорстких дисків та материнської плати в реальному часі, що дозволяє користувачам контролювати тепловий стан системи та вживати заходів для охолодження, якщо це необхідно;

- моніторинг пам'яті: програма показує обсяг оперативної пам'яті, тип (DDR3, DDR4 тощо) та частоту роботи, що допомагає користувачам зрозуміти, чи їхня пам'ять відповідає вимогам сучасних програм та ігор;

- отримання інформації про жорсткі диски: Speccy надає детальні дані про жорсткі диски, включаючи їх тип (HDD або SSD), обсяг, вільне місце, а також стан SMART, що може допомогти в діагностиці проблем із зберіганням даних;

- аналіз графічного процесора: програма включає інформацію про графічну карту, модель, обсяг пам'яті та тактову частоту, що корисно для геймерів та дизайнерів, які хочуть переконатися, що їхнє обладнання відповідає вимогам програмного забезпечення;

- функція збереження звітів: Speccy дозволяє користувачам зберігати звіти про систему у форматах .txt та .xml, що робить можливим обмін інформацією з технічними спеціалістами або зберігання звітів для подальшого використання;

- автоматичне оновлення інформації: Speccy забезпечує автоматичне оновлення даних у реальному часі, що дозволяє користувачам слідкувати за змінами в системі без необхідності вручну оновлювати звіти, що може бути корисним для моніторингу стабільності системи під навантаженням.

Дане програмне забезпечення має ряд значних переваг, які роблять його популярним серед користувачів. Серед переваг можна виділити такі:

- простота використання: Speccy має інтуїтивно зрозумілий інтерфейс, що дозволяє навіть новачкам швидко отримати доступ до всієї необхідної інформації без зайвих труднощів;

- повнота даних: програма надає вичерпні дані про всі компоненти системи, що дозволяє користувачам отримати повне уявлення про технічні характеристики свого ПК;

- моніторинг температури: Speccy дозволяє контролювати температуру різних компонентів системи, що важливо для запобігання перегріванню та продовження терміну служби обладнання;

- безкоштовна базова версія: базова версія Speccy має необхідні функції для моніторингу, що робить програму привабливою для широкого кола користувачів;

- можливість збереження звітів: користувачі можуть зберігати звіти про систему у різних форматах, що полегшує обмін інформацією та її подальше використання;

- підтримка різних апаратних компонентів: Speccy сумісна з більшістю сучасних процесорів, материнських плат та графічних карт, що забезпечує її універсальність та широке застосування;

- регулярні оновлення: програма отримує регулярні оновлення, що забезпечує підтримку нових компонентів та виправлення помилок, покращуючи загальну стабільність програми;

- малий обсяг та низькі системні вимоги: Speccy займає небагато місця на диску і не навантажує систему, що робить її придатною для використання навіть на старіших або малопотужних пристроях, що дозволяє отримувати детальну інформацію про систему без значного впливу на продуктивність комп'ютера.

Speccy є хорошим та зручним інструментом для моніторингу апаратного забезпечення, але програма має і недоліки:

- обмежена інформація про графічний процесор: хоча Speccy надає деякі дані про графічний процесор, його можливості моніторингу та

тестування не такі широкі, як у деяких інших програм для моніторингу системи;

- відсутність функцій для налаштування системи: Speccy не надає можливостей для зміни налаштувань або оптимізації системи, що може бути обмеженням для досвідчених користувачів;

- програма доступна лише для ОС Windows: Speccy розроблена виключно для операційної системи Windows, що обмежує її використання на інших платформах, таких як macOS або Linux;

- проблеми з сумісністю: деякі користувачі можуть стикатися з проблемами сумісності з певними компонентами або версіями Windows, що може призвести до неточностей у відображенні даних;

- обмежена підтримка температурного моніторингу: хоча Speccy відображає температури, не всі компоненти можуть бути точно підключені для моніторингу температури, що може призвести до неповних даних;

- відсутність інструментів для діагностики несправностей: Speccy не надає можливостей для глибокої діагностики несправностей або тестування стабільності компонентів, що може бути недоліком для користувачів, які шукають комплексний діагностичний інструмент;

- нав’язлива реклама: компанія Piriform веде дуже агресивну рекламну політику. Рекламні банери присутні у кожному їх застосунку, а іноді вони не дають користуватись функціями без ознайомлення з рекламними пропозиціями;

- вискоа ціна: ціни на продукти компанії розробника є зависокими для звичайного користувача, а також вони змушують платити за непотрібний користувачу функціонал.

1.5 Утиліта для збільшення можливостей відеокарт MSI Afterburner

MSI Afterburner – популярна утиліта для моніторингу та управління параметрами графічної карти, яка була розроблена компанією MSI. Спочатку

вона була створена для роботи з відеокартами MSI, але здобула широку популярність завдяки своїй сумісності з картами інших виробників. MSI Afterburner надає велику кількість функцій для моніторингу комп'ютерних системи, розгону графічного процесору, налаштування кулерів та створення профілів продуктивності, що робить її незамінною для геймерів та користувачів, що бажають оптимізувати продуктивність своїх відеокарт [6].

Завдяки інтуїтивно зрозумілому інтерфейсу (рисунок 1.7) MSI Afterburner підходить як для новачків, так і для досвідчених користувачів. Утиліта надає велику кількість можливостей, які дають користувачам повний контроль над характеристиками графічного процесора, що дозволяє не тільки підвищити продуктивність, але й відстежувати стан комп'ютерної системи в реальному часі.

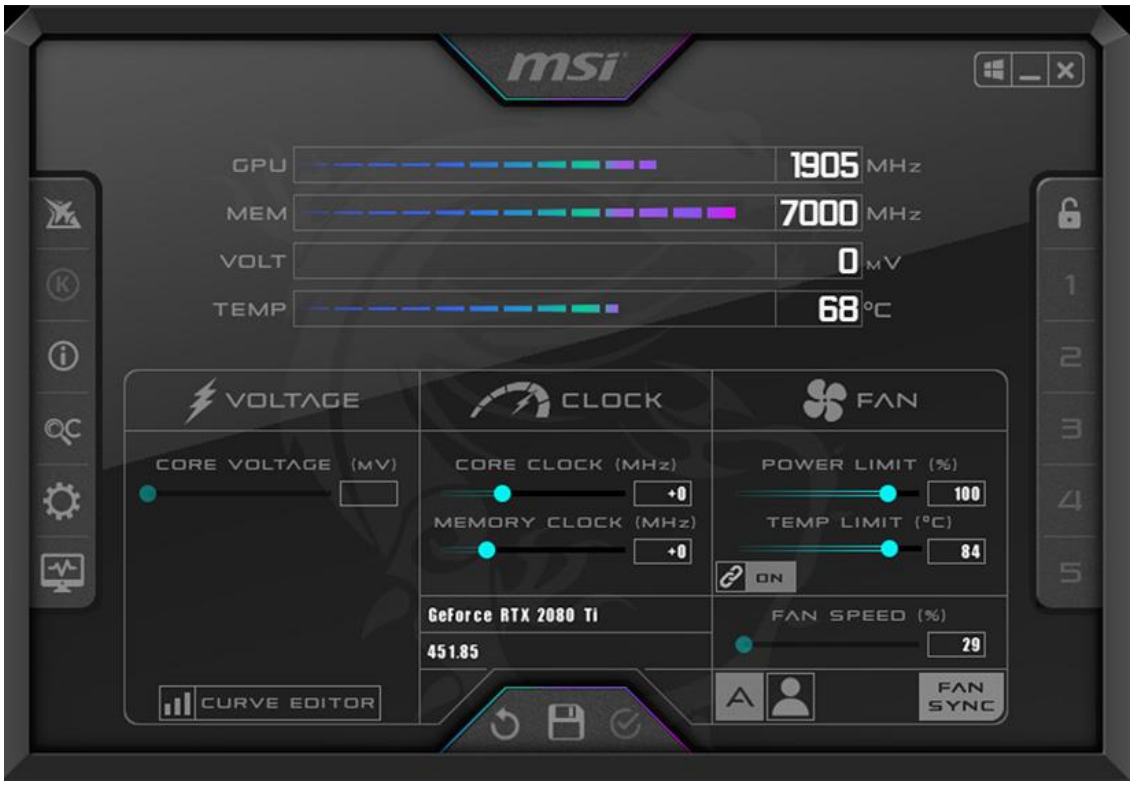

Рисунок 1.7 – Інтерфейс MSI Afterburner

Основними функціями даного програмного забезпечення є:

- розгін графічної карти: утілита дозволяє змінювати тактову частоту ядра та пам'яті графічного процесора, що дозволяє підвищити його

продуктивність. Програма підтримує різні методи розгону, включаючи збільшення напруги на графічний процессор для досягнення більш високих частот;

- налаштування швидкості вентилятора: утиліта дозволяє вручну або автоматично регулювати швидкість обертання вентилятора, забезпечуючи кращий контроль температури GPU та знижуючи рівень шуму, коли система не знаходиться під навантаженням;

- моніторинг системи в реальному часі: MSI Afterburner відображає детальну інформацію про основні компоненти системи, такі як частота графічного процессора, температура, напруга та використання оперативної пам'яті, що дозволяє користувачам бачити, як змінюються параметри при підвищених навантаженнях;

- профілі продуктивності: користувачі можуть створювати декілька профілів розгону та перемикатися між ними залежно від своїх потреб, що корисно для налаштування різних рівнів продуктивності для ігор або рендерингу;

- оверлей у грі (On-Screen Display): утиліта дозволяє виводити інформацію про продуктивність прямо на екран під час ігрових сесій. Користувачі можуть бачити температуру, частоту, використання ресурсів графічного процессора та інші параметри в реальному часі;

- стрес-тест GPU: MSI Afterburner оснащений вбудованим інструментом для стрес-тестування, який допомагає перевірити стабільність розгону та виявити потенційні проблеми з перегрівом;

- запис та стрімінг: за допомогою додаткової функції RivaTuner MSI Afterburner дозволяє записувати відео та транслювати ігровий процес, що робить її популярною серед стрімерів та контент-мейкерів.

MSI Afterburner вирізняється низкою вагомих переваг, які роблять його надзвичайно популярним серед широкого кола користувачів:

- сумісність з картами різних виробників: хоча утиліта розроблена MSI, вона працює з графічними картами інших виробників, що робить її

універсальним інструментом для розгону та моніторингу;

- безкоштовний доступ: MSI Afterburner є повністю безкоштовним додатком, що робить його доступним для всіх користувачів, які шукають якісний інструмент для оптимізації своєї системи;

- гнучкі можливості налаштування: утиліта дозволяє налаштовувати продуктивність GPU під потреби користувача, включаючи розгін, налаштування вентилятора та профілі продуктивності;

- інтуїтивно зрозумілий інтерфейс: MSI Afterburner пропонує зручний інтерфейс, що полегшує використання програми, навіть якщо користувач не має досвіду з розгоном;

- моніторинг в реальному часі: утиліта відображає дані про температуру та навантаження в режимі реального часу, що дозволяє швидко виявляти можливі проблеми з перегрівом або стабільністю системи;

- вбудовані інструменти для запису відео: можливість запису та стрімінгу через RivaTuner є значною перевагою для геймерів, які хочуть ділитися своїм ігровим досвідом з іншими;

- підтримка мультипрофільного налаштування: користувачі можуть зберігати декілька профілів налаштувань продуктивності, що дозволяє легко переключатися між ними для різних сценаріїв використання;

- відображення графіків та логування даних: MSI Afterburner дозволяє користувачам переглядати графіки змін температури, навантаження, швидкості вентиляторів та інших параметрів, а також зберігати ці дані для подальшого аналізу, що допомагає відстежувати продуктивність системи та виявляти довгострокові тренди.

MSI Afterburner є ефективним та зручним інструментом для моніторингу апаратного забезпечення, однак має і певні недоліки, які можуть стати обмеженням для деяких користувачів:

- ризик для початківців: MSI Afterburner має потужні функції розгону, що може призвести до нестабільності системи або перегріву, якщо її використовувати без належних знань. Початківці можуть випадково

налаштувати параметри, що призведе до пошкодження компонентів;

- висока споживаність ресурсів: при моніторингу та використанні оверлея утиліта може споживати значну кількість ресурсів системи, що може вплинути на продуктивність комп’ютера;

- проблеми сумісності з деякими програмами: MSI Afterburner може конфліктувати з деякими іншими утилітами моніторингу та розгону, що може викликати збої або помилки;

- обмежена функціональність для CPU: утиліта орієнтована на моніторинг та розгін графічних карт, тому її можливості для моніторингу процесора є обмеженими;

- складний інтерфейс для новачків: хоча програма має зручний інтерфейс, новачкам може знадобитися час на освоєння її численних функцій та параметрів, особливо для розгону;

- небезпека при неправильному використанні: неправильні налаштування розгону або напруги можуть пошкодити графічний процесор, що робить цю утиліту небезпечною для користувачів, які не знайомі з розгоном;

- немає мобільної підтримки: MSI Afterburner працює тільки на ОС Windows, тому користувачі не можуть використовувати її на інших платформах, таких як macOS або Linux;

- відсутність вбудованих інструкцій: MSI Afterburner не має детальних вбудованих інструкцій або підказок для новачків, що може ускладнити опанування.

1.6 Висновки за розділом

Після аналізу основних функцій та характеристик п’яти популярних програм для моніторингу комп’ютерної системи було складено зведену таблицю їх переваг, недоліків та можливостей (таблиця 1.1). П’ять розглянутих програм мають різні особливості, які роблять їх корисними для

певних категорій користувачів. AIDA64 ідеально підходить для детального аналізу та моніторингу всіх компонентів системи, але має платну ліцензію. HWMonitor відзначається простотою використання та базовими функціями для моніторингу температури та напруги, проте обмежений у функціоналі. CPU-Z надає інформацію про процесор та пам'ять, але не підтримує моніторинг температури. Speccy забезпечує повну інформацію про систему та підтримує моніторинг температури, але не має функцій для налаштування. MSI Afterburner орієнтований на геймерів, пропонуючи розгін та моніторинг GPU, але також обмежений у загальних можливостях для CPU.

Таблиця 1.1 – Зведена таблиця характеристик та функцій

Функція/Характеристика	AIDA	HWMonitor	CPU-Z	Speccy	Afterburner
Моніторинг температури	+	+	–	+	+
Моніторинг напруги	+	+	–	–	+
Інформація про CPU	+	–/+	+	+	–
Інформація про RAM	+	–/+	+	+	–
Інформація про GPU	+	–/+	+	+	+
Тести продуктивності	+	–	+	–	+
Регулювання вентиляторів	–	+	–	–	+
Збереження звітів	+	–	–	+	–
Віддалений моніторинг	+(Pro)	+(Pro)	–	–	–
Безкоштовна версія	–	+	+	+	+
Сумісність з різними ОС	Win	Win	Win	Win	Win

Результатом розробки має стати єдиний безкоштовний програмний застосунок, який буде повністю або частково повторювати функціональність перелічених вище утиліт, з додаванням розширених функцій моніторингу комп’ютерної системи.

2 АНАЛІЗ ТА ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗРОБКИ ЗАСТОСУНКУ ДЛЯ МОНІТОРИНГУ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Мова програмування C#

C# – високорівнева мова програмування загального призначення, яка була розроблена корпорацією Microsoft як інструмент для розробки застосунків для платформи .NET [7]. Вона поєднала в собі багато рис своїх попередників, а саме C++ та Java і пропонує сучасні можливості для створення різноманітних додатків. Завдяки її еволюції та вдосконаленням, C# стала однією з найпопулярніших мов у розробці програмного забезпечення для Windows, вебзастосунків, ігор та навіть хмарних сервісів (рисунок 2.1).

Apr 2024	Apr 2023	Change	Programming Language
1	1		Python
2	3	▲	Java
3	7	▲	JavaScript
4	5	▲	C#
5	4	▼	C/C++
6	8	▲	SQL
7	-	▲	Swift
8	10	▲	Go
9	6	▼	Visual Basic

Рисунок 2.1 – Рейтинг C# в порівнянні з іншими мовами

Мова підтримує об'єктно-орієнтований підхід, що дає розробникам можливість створювати більш структурований та зрозумілий код. Об'єктно-орієнтоване програмування у C# реалізується через класи, інтерфейси, спадкування та поліморфізм, що полегшує управління великими проєктами за рахунок повторного використання коду. Завдяки цьому мова C# підходить як для невеликих застосунків, так і для великих корпоративних систем.

Ключовою перевагою мови C# є її інтеграція з платформою .NET, а це

надає доступ до великої кількості бібліотек, які допоможуть в вирішенні практично будь-яких задач. Наприклад, бібліотека System.Management дозволяє взаємодіяти з WMI для моніторингу системи, а System.Diagnostics надає можливості для роботи з процесами та логами. Ці бібліотеки значно спрощують взаємодію із системними компонентами, що особливо важливо при створенні застосунків, які потребують доступу до внутрішніх ресурсів операційної системи.

В мову C# також вбудовані механізми для управління пам’яттю, зокрема автоматичний збір сміття (Garbage Collection), який знижує ризик виникнення помилок через витоки пам'яті. Це робить C# безпечнішою в плані управління ресурсами порівняно з мовами, які вимагають ручного управління пам’яттю, як приклад мова програмування C++. Слід зауважити, що застосунки створені на мові C#, за замовченням будуть використовувати трохи більше пам'яті, але це не є критичним при створенні великих програм.

Ще однією з важливих особливостей мови C# слід зазначити підтримку асинхронного програмування, яка реалізується через використання ключових слів "async" та "await". Це дає можливість створювати високопродуктивний код, який не блокуватиме основний потік виконання. Дана можливість матиме критичне значення при створенні проєкту так як створюватись буде власне графічний застосунок, який має відповідати на дії користувача навіть коли будуть проводитись якісь складні фонові обчислення [8].

Вибір мови програмування C# для реалізації даного проєкту зумовлений тим що, ця мова ідеально інтегрується з іншими технологіями Windows, такими як WMI та PowerShell. Легка інтеграція з ними спрощує процес доступу до системних даних та дає можливість легко автоматизувати різноманітні системні задачі без значних витрат на написання коду, включно з можливістю запуску різноманітних системних команд, управлінням файлами в системі та користувачами в разі необхідності, а також доступу до датчиків різних компонентів.

Microsoft Visual Studio як основне середовище розробки для C#

забезпечить проєкт потужними інструментами для створення, тестування та налагодження коду, а наявність вбудованих шаблонів проєктів, інструментів для автоматичного завершення коду (IntelliSense) та інтеграції з системами контролю версій, такими як Git, значно спростить процес розробки.

Таким чином, використання C# в проєкті надасть широкий спектр інструментів для створення функціонального, ефективного та безпечного програмного забезпечення, яке тісно інтегрується з екосистемою Windows.

2.2 Технологія Windows Management Instrumentation (WMI)

Windows Management Instrumentation (WMI) – це технологія Microsoft, яка надає стандартизований спосіб доступу до інформації про апаратне, програмне забезпечення та системні компоненти операційної системи Windows [9]. WMI (рисунок 2.2) забезпечує інтерфейс для запитів, моніторингу та управління локальними або віддаленими системами, що робить її важливим інструментом для адміністраторів та розробників системного ПЗ. WMI було впроваджено в рамках Windows 2000 як частину загального стандарту Web-Based Enterprise Management (WBEM), який базується на моделях розроблених Distributed Management Task Force (DMTF). Ця технологія дозволяє програмам взаємодіяти з системними компонентами через стандартизовані попередньо створені методи, не залежачи від особливостей виробника апаратного або ПЗ.

Спочатку WMI призначалася для IT-адміністраторів і була створена щоб спростити управління великими мережами, але згодом її потенціал був визнаний і розробниками, які почали використовувати WMI для розробки спеціалізованих інструментів моніторингу, діагностики та автоматизації системних завдач.

WMI базується на архітектурі "провайдерів" (providers), які відповідають за доступ до певного типу ресурсів. Провайдери взаємодіють з ядром Windows та іншими компонентами, збираючи системну інформацію

або виконуючи дії за запитом. Інформація, яка надається WMI, структурується у вигляді класів, що належать до певного простору імен (namespace). Основним простором імен є root\cimv2, де зберігається більшість стандартних класів для роботи з апаратними та програмними компонентами.

Наприклад:

- Win32_Processor – клас, що надає інформацію про процесори;
- Win32_OperatingSystem – клас для отримання даних про ОС;
- Win32_Service – клас для управління службами Windows.

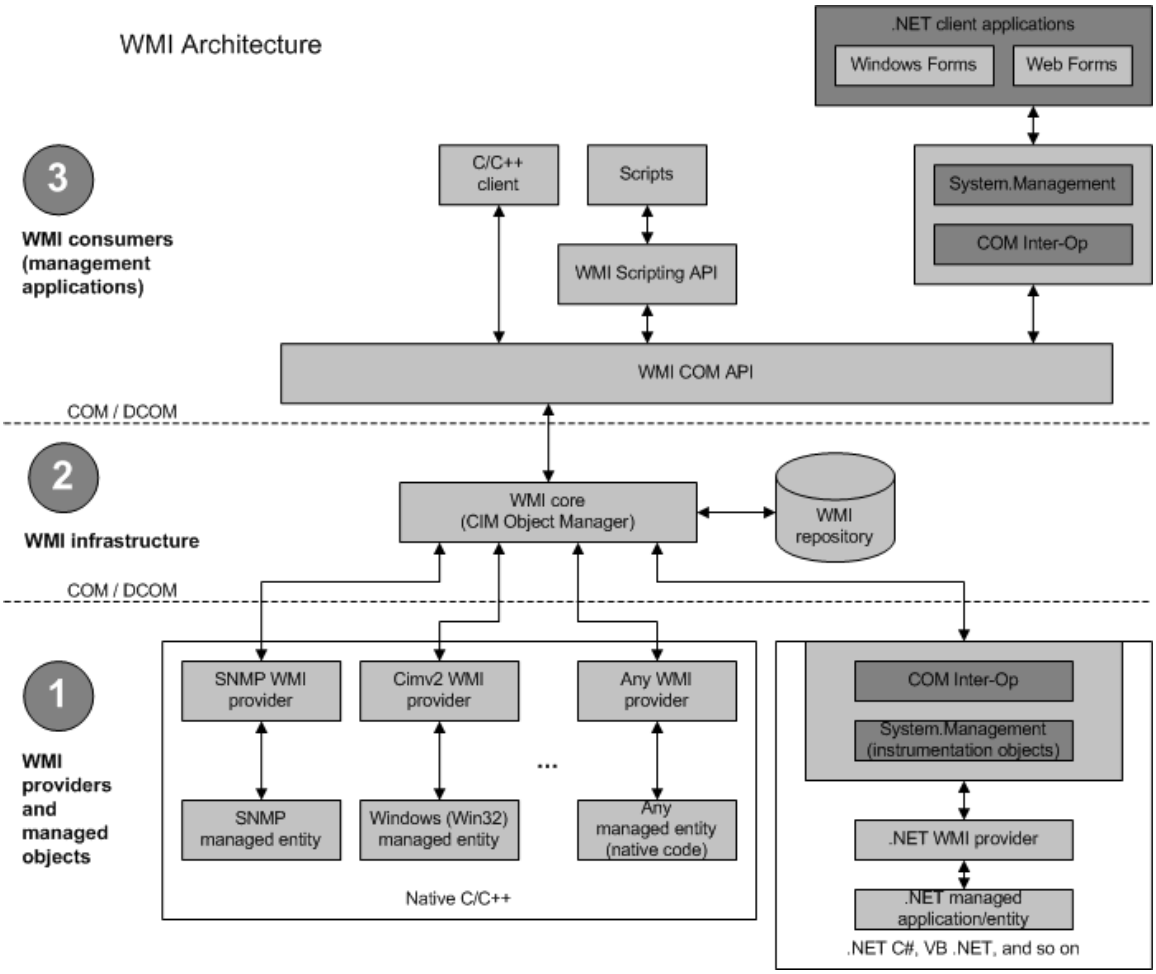

Рисунок 2.2 – Архітектура WMI

Доступ до WMI може здійснюватися через різні інтерфейси, такі як CIM (Common Information Model), DMTF стандарту або за допомогою мов програмування, таких як: C#, VBScript або команд PowerShell.

WMI легко інтегрується з застосунками, які було створено за допомогою мови C# та простору імен System.Management. Дана бібліотека надає класи та методи для створення запитів до WMI, обробки результатів та управління системними ресурсами. Для запиту інформації з WMI використовується мова WQL (WMI Query Language), яка базується на SQL. Запит для отримання інформації про процесори наведено у лістингу 2.1.

Лістинг 2.1 – Текст запиту інформації про процесори

```
string query = "SELECT * FROM Win32_Processor";
ManagementObjectSearcher searcher = new
ManagementObjectSearcher(query);
foreach (ManagementObject obj in searcher.Get())
{Console.WriteLine($"Процесор: {obj["Name"]}, Ядер:
{obj["NumberOfCores"] } ");
}
```

Переваги використання WMI:

- надає єдиний інтерфейс для доступу до різних типів системної інформації, що спрощує інтеграцію з іншими інструментами;

- дозволяє працювати з віддаленими системами, що корисно для великих мереж або хмарних середовищ;

- завдяки інтеграції з PowerShell WMI може бути використана для написання скриптів автоматизації.

WMI є потужним інструментом, який робить управління Windows-системами більш ефективним. Використання її для розробки проєкту дозволить створити застосунок, здатний автоматизувати складні системні процеси та надати користувачам зручний доступ до ключових системних даних.

2.3 Командна оболонка PowerShell

PowerShell – це сучасна командна оболонка та мова сценаріїв, створена Microsoft для автоматизації адміністративних задач та управління системами

Windows [10]. Її поява стала важливою віхою у розвитку засобів адміністрування, оскільки PowerShell об’єднала можливості командного рядка, скриптової мови та розширених функцій управління, які раніше були доступні лише через графічні інтерфейси або складні інструменти.

Розробка PowerShell почалася у середині 2000-х років, коли Microsoft усвідомила потребу в більш потужному інструменті для адміністраторів, здатному обробляти складні задачі автоматизації. На відміну від класичного командного рядка Windows (CMD), який мав обмежений набір функцій та погану інтеграцію з сучасними системними компонентами, PowerShell базується на платформі .NET, що дозволяє використовувати всі можливості .NET-бібліотек для роботи з файлами, службами, мережами та базами даних.

Однією з ключових особливостей PowerShell є її об'єктно-орієнтований підхід до роботи з даними. Усі команди в PowerShell повертають об’єкти, а не просто текст, що дає змогу легко обробляти отриману інформацію без необхідності складного парсингу. Наприклад, результат команди для отримання списку процесів можна безпосередньо фільтрувати, сортувати або передавати іншим командам, що робить PowerShell значно потужнішим інструментом у порівнянні з класичними командними оболонками.

PowerShell підтримує так звані командлети (cmdlets) – спеціалізовані команди, призначені для виконання конкретних завдань. Наприклад, командлет Get-Process надає інформацію про запущені процеси, а Set-Service дозволяє змінювати стан служб Windows. Кожен командлет має уніфіковану структуру з іменами у форматі дієслово-іменник, що робить їх логічними та простими для запам’ятовування.

Найбільша перевага PowerShell полягає в її здатності об'єднувати кілька командлетів у складні сценарії. Наприклад, адміністратор може створити скрипт для автоматичного моніторингу ресурсів системи та надсилання повідомлень електронною поштою у разі перевищення певних порогових значень. Це дозволяє значно зменшити кількість ручної роботи та знижує ризик людських помилок.

Інтеграція з WMI є ще одним з плюсів PowerShell. Завдяки цьому можна безпосередньо отримувати інформацію про апаратне, ПЗ, мережні підключення та інші системні параметри, використовуючи командлети на зразок Get-WmiObject. Наприклад, для отримання інформації про процесор досить виконати команду Get-WmiObject -Class Win32_Processor.

PowerShell також підтримує роботу з віддаленими системами, що робить її ідеальним вибором для адміністраторів великих мереж або хмарних середовищ. Замість фізичного доступу до кожного комп’ютера, можна віддалено керувати системами, встановлювати оновлення, змінювати налаштування безпеки або запускати діагностику.

Розробники програмного забезпечення активно використовують PowerShell для створення скриптів автоматизації, тестування програм і навіть розгортання застосунків. Це досягається завдяки можливості інтеграції PowerShell у Visual Studio, що дозволяє запускати скрипти безпосередньо з середовища розробки. Загалом, PowerShell перетворився на універсальний інструмент, який об'єднує можливості управління, автоматизації та інтеграції у Windows-середовищі.

У проєкті PowerShell може бути використаний для взаємодії з WMI, автоматизації системних процесів та розширення функціоналу додатку, що дозволить створити потужне рішення для адміністрування та моніторингу.

2.4 Бібліотека Windows Forms

Windows Forms (WinForms) – це одна з перших бібліотек для створення графічних інтерфейсів користувача (GUI) у середовищі Windows [11]. Вона була представлена у складі платформи .NET Framework та надавала розробникам інтуїтивно зрозумілий засіб створення настільних застосунків із використанням мови C#. Завдяки простоті у використанні та широкому набору стандартних елементів інтерфейсу, Windows Forms довгий час залишалася основним інструментом для розробки програм під Windows.

Windows Forms базується на моделі подій (event-driven), що дозволяє створювати інтерактивні застосунки, які реагують на дії користувача в реальному часі. Розробка в WinForms орієнтована на використання компонентів – готових елементів інтерфейсу, таких як кнопки, текстові поля, списки, мітки та інші. Ці компоненти легко налаштовуються і можуть бути розміщені на формі за допомогою візуального конструктора, що вбудований у Visual Studio.

В основі Windows Forms лежить модель контролів, де кожен елемент GUI є об'єктом класу (рисунок 2.3), що успадковує базовий клас Control. Ця структура дозволяє легко налаштовувати поведінку елементів та обробляти події, пов’язані з ними. Наприклад, при натисканні на кнопку викликається метод, що обробляє цю подію (лістинг 2.2).

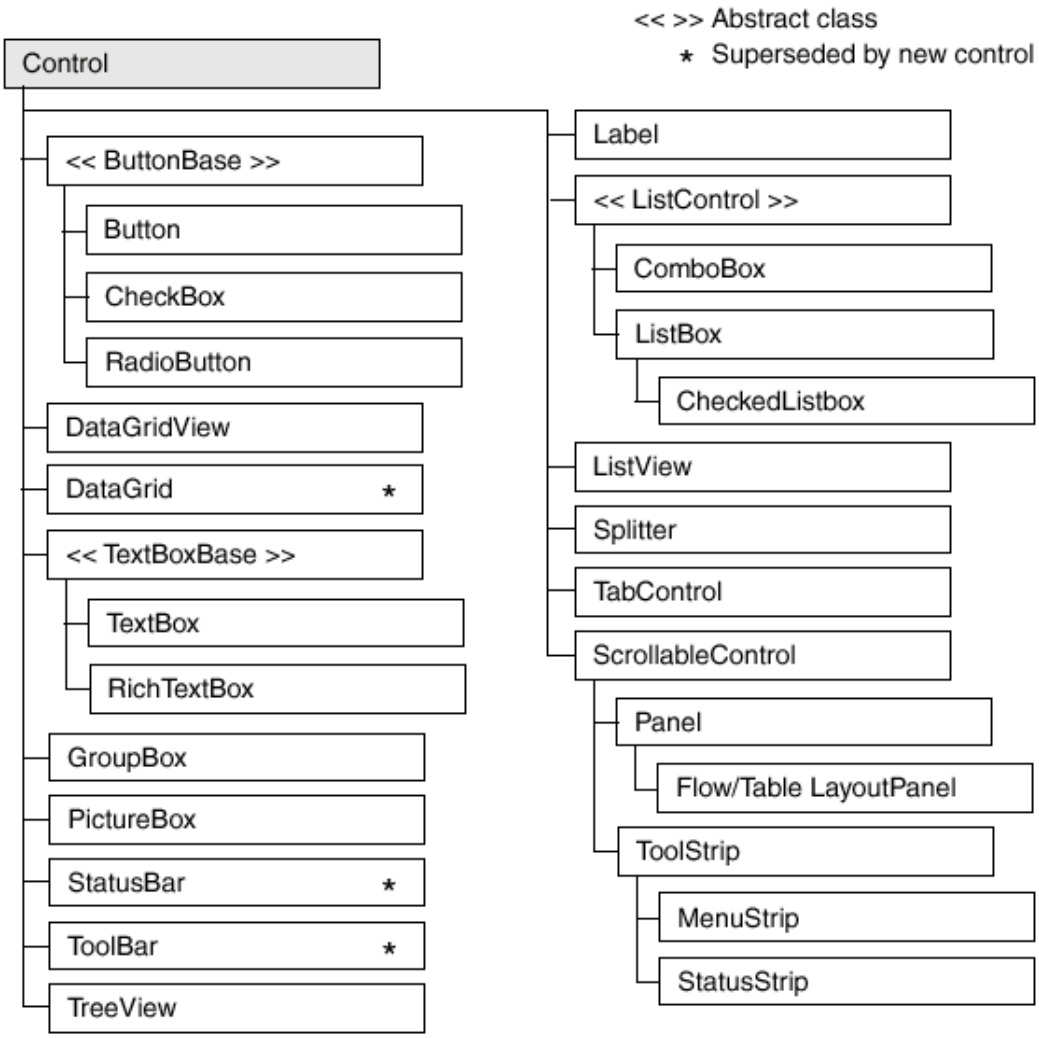

Рисунок 2.3 – Дерево контролів Windows Forms

Лістинг 2.2 – Обробник події

```
private void button1_Click(object sender, EventArgs e)
{     MessageBox.Show("Кнопка натиснута!");   }
```

Розробка інтерфейсу в Windows Forms є однією з найбільш інтуїтивних завдяки інтегрованому конструктору форм. У Visual Studio розробник може створювати форми методом "перетягування" компонентів на поверхню форми, налаштовуючи їхні властивості через зручний графічний інтерфейс. Наприклад, змінити текст на кнопці або її розмір можна без написання коду, використовуючи лише властивості у вікні "Properties".

Окрім стандартних компонентів, таких як кнопки, текстові поля або випадаючі списки, WinForms підтримує додавання компонентів користувача та сторонніх бібліотек, що дозволяє значно розширити функціональні можливості застосунку. Це особливо корисно для створення складних інтерфейсів або специфічних бізнес-додатків.

Попри те, що Windows Forms є застарілою технологією у порівнянні з WPF (Windows Presentation Foundation) або MAUI (Multi-platform App UI), вона все ще має значну популярність через простоту використання, низький поріг входу та широку підтримку у Visual Studio. Особливо це актуально для внутрішніх корпоративних застосунків, де важлива швидкість розробки та підтримка вже існуючих систем.

Ще однією вагомою перевагою Windows Forms є її інтеграція з іншими технологіями Windows, такими як WMI та PowerShell. Завдяки цьому розробники можуть створювати додатки, які не лише мають зручний інтерфейс, а й виконують складні системні операції, наприклад, моніторинг системних ресурсів або автоматизацію адміністративних завдань [12].

В рамках даного проєкту використання Windows Forms дозволить швидко створити зручний графічний інтерфейс для взаємодії користувача з застосунком. Це особливо важливо для програми, яка має надавати користувачам доступ до важливих функцій через інтуїтивно зрозумілі елементи управління.

2.5 Інтегроване середовище розробки Visual Studio

Visual Studio – це інтегроване середовище розробки (IDE), створене компанією Microsoft, яке надає розробникам потужний набір інструментів для створення програмного забезпечення на різних платформах. Visual Studio підтримує широкий спектр мов програмування, серед яких C#, C++, Visual Basic, F#, Python, JavaScript, і є одним із найпопулярніших середовищ для розробки додатків під Windows [13].

Visual Studio вирізняється своєю гнучкістю та інтеграцією з іншими технологіями Microsoft, такими як .NET Framework, .NET Core, Azure, Windows Forms, WPF, а також засоби автоматизації та адміністрування, зокрема WMI та PowerShell. Завдяки цьому, воно є універсальним рішенням як для розробників настільних, так і веб- та хмарних додатків.

Однією з ключових переваг Visual Studio є наявність інтуїтивного візуального конструктора (рисунок 2.4), який полегшує процес розробки графічного інтерфейсу користувача (GUI).

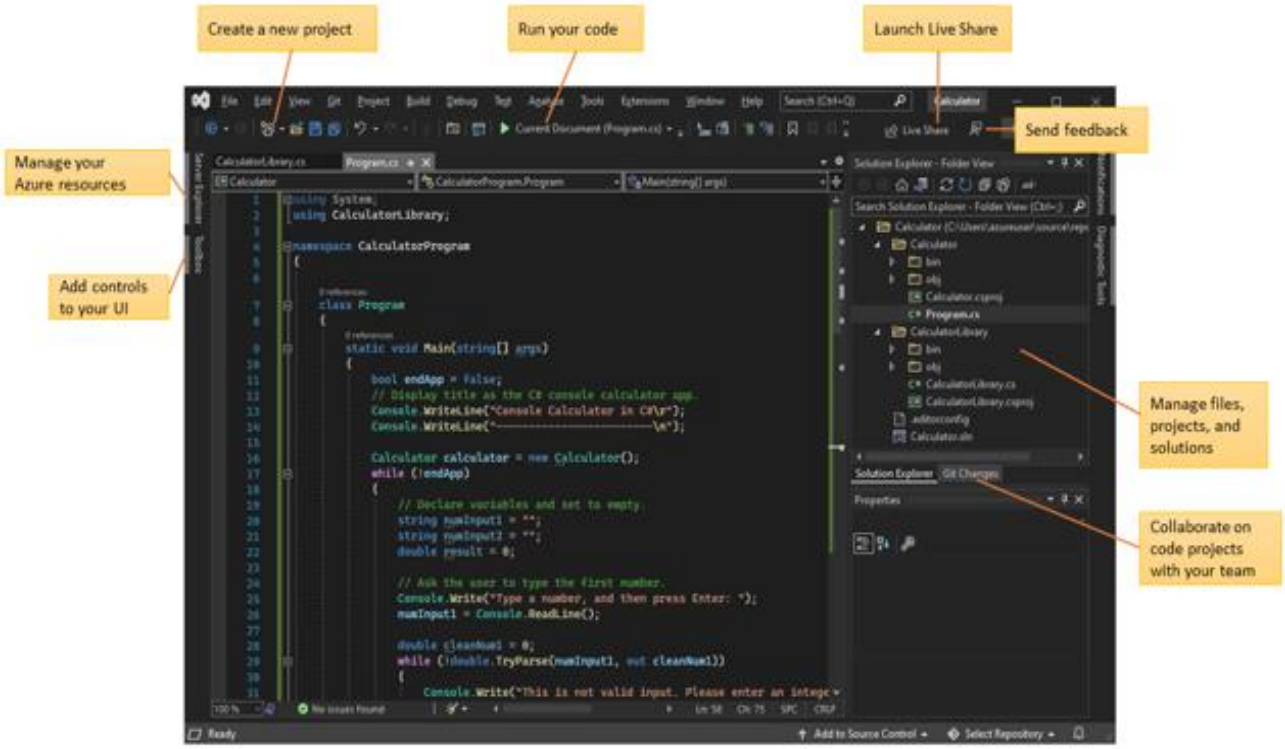

Рисунок 2.4 – Інтерфейс Visual Studio

Visual Studio підтримує IntelliSense – систему автодоповнення, яка надає підказки щодо методів, властивостей та класів у реальному часі, допомагає зменшити кількість помилок у коді та підвищує продуктивність. Крім того, Visual Studio має потужний дебагер, який дозволяє відстежувати виконання програми покроково, перевіряти значення змінних у реальному часі, швидко знаходити помилки.

Ще однією особливістю є підтримка інтеграції з системами контролю версій (Git та Azure DevOps), що дозволяє легко керувати версіями коду, працювати в командах та автоматизувати процеси збірки та тестування. Інструменти для керування версіями вбудовані прямо в середовище, що дозволяє виконувати операції коміту, злиття або створення гілок безпосередньо з Visual Studio. Visual Studio пропонує потужну підтримку для роботи з PowerShell та WMI, що робить його ідеальним вибором для розробників системних утиліт та застосунків для адміністрування.

У середовищі розробки доступні інструменти для тестування коду (юніт-тести, автоматичне генерування тестових сценаріїв та візуалізацію покриття коду тестами), що дає можливість підтримувати високий рівень якості ПЗ та швидко виявляти регресії під час внесення змін до коду.

Visual Studio підтримує широкий спектр розширень, які дозволяють налаштовувати середовище під потреби конкретного проєкту. Це можуть бути додаткові шаблони проєктів, інструменти для аналізу коду, інтеграція з базами даних або засоби для роботи з хмарними платформами. Така гнучкість дозволяє використовувати Visual Studio для проєктів різного масштабу та складності, починаючи від невеликих програм до великих корпоративних систем.

У рамках даного проєкту Visual Studio є оптимальним вибором завдяки своїй інтеграції з .NET і підтримці таких технологій, як Windows Forms, WMI та PowerShell. Це середовище дозволяє швидко створювати функціональні додатки з графічним інтерфейсом, налагоджувати їх, тестувати та автоматизувати багато процесів, що значно підвищує ефективність розробки.

3 ОПИС РЕАЛІЗАЦІЇ ЗАСТОСУНКУ

3.1 Створення UML-діаграми прецедентів

Для якісного планування та визначення функціоналу майбутнього застосунка потрібно провести його проєктування. UML-діаграма може описувати взаємодію користувача та системи між собою [14]. Для опису типових взаємодій користувача з системою та надання опису процесу її функціонування використовують діаграми прецедентів (рисунок 3.1).

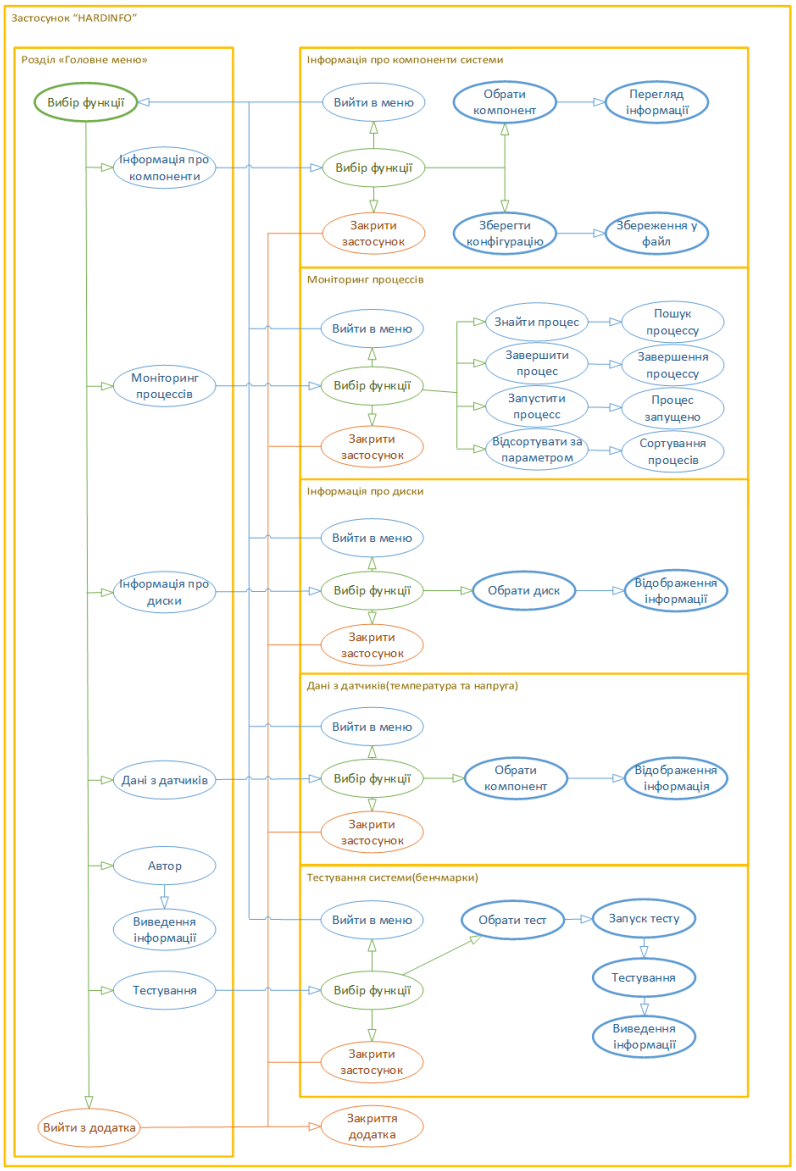

Рисунок 3.1 – UML-діаграма прецедентів

3.2 Побудова профіля вимог до програмного забезпечення

У рамках проєктування застосунку відбувається оголошення вимог до застосунку та будується їх нормативний профіль, який являє собою набір функціональних і нефункціональних вимог [15]. У таблиці 3.1 представлено нормативний профіль вимог до програмного забезпечення.

Таблиця 3.1 – Нормативний профіль вимог до застосунку

Вимоги	Повнота	Необхідність	Однозначність	Перевірюваність	Здійсненність	Коректність
1	2	3	4	5	6	7
Вимоги користувача						
Вхід у застосунок	+	+	+	+	+	+
Користувач може відкривати вікна	+	+	+	+	+	+
Застосунок має запускатись без попередньої інсталяції	+	+	+	+	+	+
Виведення інформації у зручний для користувача спосіб	+	+	+	+	+	+
Маленький розмір файлу програми	+/–	+/–	+	+	+/–	+
Вихід із застосунку	+	+	+	+	+	+
Отримання інформації про систему	+	+	+	+	+	+
Отримання даних про диски	+	+	+	+	+	+
Моніторинг даних з датчиків	+/–	+	+	+	+	+
Моніторинг процесів	+/–	+	+	+	+	+
Тестування системи (бенчмарк)	+/–	+	+	+	+	+
Збереження конфігурації ПК у файл	+	+	+	+	+	+

Продовження таблиці 3.1

1	2	3	4	5	6	7
Функціональні вимоги						
Запити в систему	+	+	+	+	+	+
Графічне подання інформації	+	+	+	+	+	+
Створення алгоритмів розрахунків	+	+	+	+	+	+
Створення алгоритмів тестування	+	+	+	+	+	+
Створення алгоритмів управління	+	+	+	+	+	+
Створення запитів у систему	+	+	+	+	+	+
Атрибути якості						
Ремонтопридатність	+	+	+	+	+	+
Відмовостійкість	+/–	+	+	+	+	+
Переносимість	+/–	+	+	+	+	+
Продуктивність	+/–	+	+	+	+	+
Зручність використання	+/–	+	+	+	+	+
Обмеження						
Операційна система Windows	+	+	+	+	+	+
Відсутність файлу інсталяції	+	+	+	+	+	+

3.3 Формування технічного завдання програмного забезпечення

Технічне завдання (ТЗ) – це базовий документ, який слугує відправною точкою для реалізації проєкту в галузях будівництва, машинобудування, розробки програмного забезпечення, автоматизованих систем, а також при виконанні науково-дослідних робіт. Його роль є дуже важливою , адже саме на основі технічного завдання здійснюються проєктування, виготовлення, введення в експлуатацію та подальше використання об’єкта чи системи.

У документі докладно описуються вимоги до функціональності, технічних характеристик, архітектури та умов експлуатації майбутнього продукту або системи. Кожен розділ технічного завдання виконує свою

окрему функцію, і в сукупності вони формують цілісне уявлення про проєкт.

Це дає змогу всім учасникам процесу (від замовника до виконавців) мати однакове бачення кінцевого результату. У випадку розробки програмного забезпечення, технічне завдання визначає ключові технічні параметри, логіку роботи системи, інтерфейс користувача, вимоги до безпеки, масштабованості та підтримки. Тобто, це не просто формальний документ – це план, який допомагає уникнути непорозумінь на всіх етапах реалізації [16].

Функціональні вимоги передбачають можливість входу та виходу з застосунку, відкриття вікон користувачем, запуск без інсталяції, а також відображення інформації про систему та диски. Забезпечується моніторинг даних з датчиків та процесів, наявність інструментів для тестування системи, збереження конфігурації, відображення даних в інтерфейсі, а також реалізація алгоритмів для розрахунків, тестування та управління. Підтримуються запити до системи.

Нефункціональні вимоги включають відмовостійкість, переносимість, ремонтопридатність, доступність та зручність використання. Для роботи передбачено використання операційної системи Windows із забезпеченням необхідного рівня продуктивності.

Щодо технічних характеристик, мінімальні системні параметри включають ОС Windows, 512 МБ оперативної пам’яті, більше 300 МБ вільного місця на накопичувачі, а також наявність миші та клавіатури.

Розробка проходить через етапи аналізу предметної області, вибору мови та платформи, формування вимог, написання коду, тестування та оформлення документації. Для створення застосовуються мова C# з відповідними бібліотеками, Windows Forms та середовище Visual Studio.

Контроль якості здійснюється через декілька видів тестування: Smoke testing, перевірку збірки та тестування взаємодії. Прийом програмного продукту можливий за умови досягнення 95% функціональної повноти та відсутності критичних багів рівнів S1, S2 та S3.

3.4 Опис програмної реалізації

Завдяки використанню Windows Forms, кодування значно спрощується, непотрібно приділяти дуже багато часу питанню виведення інформації, бо Windows Forms мають велику кількість вже готових компонентів.

Почнемо з головного меню, так як застосунок поділено на окремі вікна з різним функціоналом, необхідно реалізувати переміщення між ними. Переміщення було виконано через додавання кнопок з подіями переходу до різних вікон. Кожна така подія містить в собі один і той самий код, який відрізняється лише посиланням на форму яку він активує. У лістингу 3.1 описаний код, який обробляє подію натискання на кнопку "Компоненти".

Лістинг 3.1 – Обробка натискання кнопки що викликає нове вікно

```
private void componbutton_Click(object sender,EventArgs e)
{ComponentsForm ComponentsFormOP = new ComponentsForm();
ComponentsFormOP.Show();}
```

В головному меню реалізовано декілька переходів між формами до розділів: "Компоненти", "Процеси", "Диски", "Датчики", "Бенчмарк", а також дві унікальні події, це виведення інформації про автора застосунку та вихід з нього. Інформація про автора з’являється якщо натиснути на зображення у головному меню, а кнопка виходу знаходиться серед кнопок у самому меню. У лістингу 3.2 описаний код, який обробляє натискання на зображення, що в свою чергу викликає невелике діалогове вікно з інформацію про автора застосунку.

Лістинг 3.2 – Обробка натискання на панель з зображення

```
private void pcgif_Click(object sender, EventArgs e){
     MessageBox.Show("Кондратюк І.О.\n
КІУКІу-22-2", "Створив", MessageBoxButtons.OK);}
```

Вихід з застосунку реалізовано викликом вбудованої функції, яка ініціює зупинку процесу застосунку (лістинг 3.3).

Лістинг 3.3 – Реалізація виходу з застосунку

```
private void exitbutton_Click(object sender, EventArgs e){
     Application.Exit();}
```

На цьому функціональна частина вікна "Головного меню" завершується. Кнопка "Компоненти" відправляє користувача з головного меню до іншої форми, з відповідною назвою. Тут у вигляді таблиці користувачеві надається інформація про наявні у системі компоненти, наприклад: процесор, графічний процесор, диски, контролери та інше, а також за наявності елементів цих пристроїв та їх драйвери.

Для того щоб не ускладнювати програму великою кількістю запитів, було створено єдиний метод, який формує запити на різні компоненти. Метод приймає в себе три параметри, це тип пристрою, простір імен в якому потрібно шукати інформацію та поле з інформацією, яку треба повернути. Таким чином виклик метода формує таблицю, в якій користувач може побачити список компонентів, їх імена та типи. В якості недоліку роботи цього методу слід зазначити, що іноді він може повернути один і той самий компонент декілька разів, хоча заходи протидії цьому було вжито в окремому методі, але якщо в списку параметрів дубліката відрізняється хоч одне значення метод сприймає його як інший елемент. У лістингу 3.4 наведено код, що формує список компонентів.

Лістинг 3.4 – Метод для отримання інформації від WMI

```
private void AddComponentInfo(string type, string wmiClass,
string propertyName)
 {ManagementObjectSearcher searcher = new
ManagementObjectSearcher($"SELECT * FROM {wmiClass}");
     foreach (ManagementObject obj in searcher.Get())
{string name = obj[propertyName]?.ToString()??"Unknown";
string key = Guid.NewGuid().ToString();
String[] row = { name, type };
ListViewItem item = new ListViewItem(row);
         componentsList.Items.Add(item);
         item.Tag = key;
         componentKeys[key] = obj.Path.ToString();}}
```

Однак не всю інформацію достатньо лише вивести, деяка інформація потребує попередньої обробки або вимагає специфічного способу її отримання. Тому виникає необхідність створення окремих методів для отримання даних про деякі компоненти. Наприклад, якщо зробити простий запит про оперативну пам’ять система поверне дані про неї в байтах і герцах, тому їх потрібно конвертувати в GB та GHz, також в деяких пристроях поле імені має статус N/A, а назва може бути записана в іншому полі, тому це також треба обробити. В лістингу 3.5 наданий приклад коду одного з таких спеціальних методів. Таких методів було створено не один, але перелічувати всі немає необхідності, тому що різниця в їх побудові не є дуже великою, і всі вони мають однакову мету, конвертувати дані з різних компонентів у більш зручний для користувача спосіб.

Лістинг 3.5 – Код метода "AddRAMInfo"

```
{ManagementObjectSearcher searcher = new
ManagementObjectSearcher("SELECT * FROM Win32_PhysicalMemory");
foreach (ManagementObject obj in searcher.Get()){ string
manufacturer=obj["Manufacturer"]?.ToString()?? "Unknown";
string speed = obj["Speed"]?.ToString() ?? "Unknown";
string capacity = obj["Capacity"] != null ?
(Convert.ToUInt64(obj["Capacity"]) / (1024 * 1024 *
1024)).ToString() + " GB": "Unknown";
string name = $"Manufacturer: {manufacturer}, Speed: {speed}
MHz, Capacity: {capacity}";
        string key = Guid.NewGuid().ToString();
        ListViewItem item = new ListViewItem(name);
        item.SubItems.Add("RAM");
        item.Tag = key;
        componentsList.Items.Add(item);
        componentKeys[key] = obj.Path.ToString();}}
```

Для того щоб отримати детальну інформацію користувач натискає на "компонент" у таблиці, яке призводить до виклику методу, що робить запит на отримання повної інформації про компонент. До таблиці передається особистий тег пристрою для уникнення дублювання, який привласнюється йому при формуванні таблиці, і за цим тегом виконується пошук детальної інформації про цей компонент (лістинг 3.6).

Лістинг 3.6 – Код метода "DisplayComponentDetails"

```
private void DisplayComponentDetails(string key)
{itemInfo.Items.Clear();
  if (componentKeys.ContainsKey(key))
{ManagementObject obj  =new
ManagementObject(componentKeys[key]);
obj.Get();
        foreach (PropertyData property in
obj.Properties){string[] row = { property.Name,
property.Value?.ToString() ?? "N/A" };
ListViewItem item = new ListViewItem(row);
            itemInfo.Items.Add(item);  }}
    for (int i = 0; i < itemInfo.Items.Count; i++)
    {if (i % 2 == 0){itemInfo.Items[i].BackColor =
System.Drawing.Color.WhiteSmoke;}}}
```

Також була реалізована функція збереження конфігурації системи. При натисканні відповідної кнопки буде викликаний метод, який збереже основну інформацію про комп’ютерну систему в окремий текстовий файл на робочому столі. Туди також буде збережена інформація про жорсткі диски, оперативну пам’ять, процесор, відеокарту та материнську плату. Слід зазначити, що програма зберігає не повну інформацію, а лише найважливіші пункти (лістинг 3.7).

Лістинг 3.7 – Фрагмент функції збереження інформації у файл

```
string desktopPath =
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
string filePath = Path.Combine(desktopPath,
"PC_Configuration.txt");
try {using (StreamWriter writer = new StreamWriter(filePath))
{writer.WriteLine("PC Configuration Details:");
writer.WriteLine("-----------------------------");
foreach (ListViewItem item in componentsList.Items)
```

Вікно "Моніторинг процесів" відображає процеси, що запущені в системі та ресурси, які вони використовують. По своїй суті він повторює функції стандартного диспетчера з додатковою можливістю пошуку процесу, але в нього не реалізовано відображення служб та компактний режим. Також є можливість запускати процеси за назвою виконавчого файлу. У лістингу 3.8 наведено код, що реалізує запуск процесу.

Лістинг 3.8 – Функція запуску процесу

```
private void startProcessButtons_Click(object sender, EventArgs
e)
  {string path = Interaction.InputBox("Ведіть ім'я .exe файлу",
"Запуск нової задачі");
      try{Process.Start(path);}
      catch (Exception){}}
```

Для того щоб вбити процес, необхідно визначити чи є він критично важливим для роботи системи або чи є в нього дочірні процеси, які теж треба вбити щоб вони не залишились у фоновому режимі та не витрачали ресурси системи. Після чого треба отримати ID процесу, його підпроцеси та вбити ці процеси, після чого оновити інтерфейс користувача. В лістингу 3.9 наведено фрагмент такого коду.

Лістинг 3.9 – Фрагмент функції "вбивання" процесів

```
if (processesDataGrid.SelectedRows.Count > 0)
  {string selectedProcessName =
processesDataGrid.SelectedRows[0].Cells["ProcessName"].Value.ToS
tring();
      Process selectedProcess = processList.FirstOrDefault(p =>
p.ProcessName == selectedProcessName);
      if (selectedProcess != null)
      if (IsSystemProcess(selectedProcess)){
          {MessageBox.Show($"Процес
'{selectedProcess.ProcessName}' є системним і не може бути
завершений.","Попередження", MessageBoxButtons.OK,
MessageBoxIcon.Warning);return;}
 else{int parentPid = GetParentProcessId(selectedProcess);
 bool hasChildren = HasChildProcesses(selectedProcess.Id);
             if (parentPid > 0)
             {killProcessAndChildren(parentPid}
             else if (hasChildren)
             {killProcessAndChildren(selectedProcess.Id);}
             else{killProcess(selectedProcess);
          getProcesses();
          refreshProcessesDataGrid();}}}
```

Для того щоб застосунок мав право зупиняти процеси та отримувати деякі види інформації, необхідно створити файл маніфесту, в якому необхідно вказати що застосунок має найвищій рівень доступу, що по суті надає їй права адміністратора (лістинг 3.10).

Лістинг 3.10 – Фрагмент app.manifest

```
-->
<requestedExecutionLevel level="highestAvailable"
uiAccess="false"/>
    </requestedPrivileges>
  </security>
</trustInfo>
<compatibility xmlns=
"urn:schemas-microsoft-com:compatibility.v1">
  <application>
```

Через те, що функція, яка "вбиває" процеси не розрізняє чи є процес важливим чи ні, необхідно було створити перевірку "чи є процес важливим". Функція з лістингу 3.9 лише аналізує результати перевірок та на основі результатів вбиває процес або ні. Тому необхідно було реалізувати перевірку, яка б могла максимально точно визначити важливість процесу. Це реалізовано використанням двох способів, які підстраховують один одного. Перший метод перевіряє, де знаходиться .exe файл процесу. Якщо файл знаходиться в директорії, де мають знаходитись важливі компоненти, програма забороняє зупинку процесу, але якщо через якусь причину не вдалось перевірити директорію, був створений спеціальний словник, в якому записані імена важливих системних процесів. У лістингу 3.11 наведено фрагмент цього словника.

Лістинг 3.11 – Фрагмент "словника"

```
string[] criticalProcessNames = {
"svchost", "system", "idle", "csrss", "smss", "wininit",
"winlogon",
"services", "lsass", "taskhostw", "explorer", "dwm", "spoolsv",
"audiodg", "fontdrvhost", "wlanext", "conhost", "sihost",
"rundll32", "searchindexer", "ctfmon", "mscorsvw", "dllhost",
"mmc", "perfmon", "msdtc", "wuauclt", "logonui", "dcomlaunch",
"nvvsvc", "sppsvc", "msmpeng", "werfault", "taskeng",
"taskhost", "userinit", "trustedinstaller", "igfxpers",
"nissrv"…
```

Коли функція отримає підтвердження, що процес зупинити можна, вона почне визначати чи є в цього процесу дочірні процеси. В випадку їх існування програма каскадом зупинить усі процеси пов’язані з основним, це

було реалізовано для зупинки програм, які створюють процеси, що можуть продовжити існування незалежно від батьківського. Наприклад, застосунок Steam створює декілька процесів, які не знищуються, коли основний процес зупиняється стандартним диспетчером і продовжують роботу у фоновому режимі. Це зазвичай заважає запустити Steam знов, так як він намагається створити ці процеси, а через те що вони все ще існують, його алгоритм не дає з’явитись основному вікну програми, тому був реалізований саме такий спосіб зупинки процесів. У лістингу 3.12 наведено код, який дає відповідь на питання "чи є в процесу дочірні процеси?" для подальшої їх зупинки.

Лістинг 3.12 – Функція HasChildProcesses

```
private bool HasChildProcesses(int pid){
ManagementObjectSearcher searcher = new
ManagementObjectSearcher($"Select * From Win32_Process Where
ParentProcessID = {pid}");
return searcher.Get().Count > 0;}
```

У вікні "Диски" було реалізовано подання інформації про жорсткі диски та флеш накопичувачі підключені до комп’ютера. У лістингу 3.13 наведено фрагмент коду, що відповідає за відображення дисків.

Лістинг 3.13 – Фрагмент методу відображення фізичних дисків

```
foreach (ManagementObject disk in disks){try
string model = disk["Model"]?.ToString() ?? "Невідомий диск";
string logicalDrives = string.Empty;
ManagementObjectSearcher partitionSearcher = new
ManagementObjectSearcher(
$"ASSOCIATORS OF
{{Win32_DiskDrive.DeviceID='{disk["DeviceID"]}'}} WHERE
AssocClass=Win32_DiskDriveToDiskPartition");
foreach (ManagementObject partition in partitionSearcher.Get()){
ManagementObjectSearcher logicalSearcher = new
ManagementObjectSearcher(…
new ManagementObjectSearcher(
$"ASSOCIATORS OF
{{Win32_DiskPartition.DeviceID='{partition["DeviceID"]}'}} WHERE
AssocClass=Win32_LogicalDiskToPartition");
foreach (ManagementObject logical in logicalSearcher.Get()){
logicalDrives += logical["DeviceID"]?.ToString() + ", ";}}
```

Також необхідно реалізувати оновлення інформації про наявні диски/носії у випадку змін під час роботи програми, тому було створено метод, який на постійній основі буде відслідковувати події підключення та відключення носіїв й оновлювати інтерфейс (лістинг 3.14).

Лістинг 3.14 – Фрагмент методу відображення фізичних дисків

```
public void StartWatching(){
ManagementEventWatcher connectWatcher = new
ManagementEventWatcher(
New WqlEventQuery("SELECT*FROM Win32_VolumeChangeEvent WHERE
EventType = 2")); ManagementEventWatcher disconnectWatcher=n
ew ManagementEventWatcher(
new WqlEventQuery("SELECT * FROM Win32_VolumeChangeEvent WHERE
EventType = 3"));// Реєстрація обробників подій
```

Усі наявні пристрої тобто жорсткі диски та флеш носії будуть відображені для користувача як панелі у списку, а не як поля в таблиці. Кожен окремий носій матиме свою окрему панель, натискання на цю панель має виводити інформацію про фізичний диск та його логічні розділи, а також кнопку, яка активує процес класифікації даних на диску за типом (тобто визначає об’єм, який займають зображення, відео, архіви, файли програм та інші файли). Слід зауважити, що час виконання залежить від типу носія та його обсягу: якщо це диск HDD обсягом 1ТБ, тоді класифікація може зайняти багато часу, якщо ж це буде SSD такого ж обсягу, класифікація пройде значно швидше. У лістингу 3.15 наведена класифікацію даних на диску.

Лістинг 3.15 – Фрагмент методу з класифікації даних

```
  try{var
directories=Directory.EnumerateDirectories(driveLetter, "*",
SearchOption.AllDirectories);
foreach (var directory in directories){try{
if (directory.Contains("System Volume Information") ||
directory.Contains("$RECYCLE.BIN"))
 continue; // Пропуск корзини
var files = Directory.EnumerateFiles(directory, "*.*",
SearchOption.TopDirectoryOnly);
foreach (var file in files){try{
string extension = Path.GetExtension(file).ToLower();
```

Вікно "Датчики" збирає інформацію від датчиків, які наявні в системі. Такими датчиками можуть буди датчики температури, напруги, швидкості обертання вентиляторів або показники використання ресурсів та частоти. Застосунок звертається до різних компонентів та отримує значення та заносить їх у дерево. В цьому дереві відображається компонент та усі датчики до яких вдалось звернутись та отримати інформацію. У лістингу 3.16 наведено фрагмент коду отримання даних та відображення їх у дереві.

Лістинг 3.16 – Фрагмент методу побудови дерева датчиків

```
HashSet<string> expandedNodes = new HashSet<string>();
foreach (object obj in sensorTreeView.Objects??new
List<object>())
{if (obj is SensorNode node && sensorTreeView.IsExpanded(node))
{expandedNodes.Add(node.Name);}}
List<SensorNode> nodes = new List<SensorNode>();
foreach (var hardware in computer.Hardware){
var hardwareNode = new SensorNode(hardware.Name);
foreach (var sensor in hardware.Sensors){
string value = sensor.Value.HasValue ?
sensor.Value.Value.ToString("0.##") : "N/A";
string unit = GetUnit(sensor.SensorType);
hardwareNode.Children.Add(new SensorNode(sensor.Name, value,
unit));}nodes.Add(hardwareNode);}sensorTreeView.SetObjects(nodes
);
```

Для більш зручного відображення елементів, можна використати зображення, але щоб не збільшувати обсяг застосунку зайвими файлами, можна використати стандартні іконки, які зберігаються в Windows. Таблиці з цими іконками є у кожній версії ОС, тому використання цього способу не тільки надає користувачу більш приємний для очей спосіб подання, а й заощаджує місце. У лістингу 3.17 наведено звертання до ОС Windows за іконками GPU та CPU.

Лістинг 3.17 – Лістинг отримання іконок з файлів Windows

```
iconList.Images.Add("cpu",
ExtractIconFromFile("C:\\Windows\\System32\\imageres.dll", 29));
iconList.Images.Add("gpu",
ExtractIconFromFile("C:\\Windows\\System32\\imageres.dll", 91));
```

Бенчмарк – це стандартна програма або набір програм, які виконуються на комп’ютері для точного вимірювання продуктивності, які дозволяють об’єктивно оцінити ефективність апаратного чи ПЗ, таких як процесори, відеокарти, оперативна пам’ять та інші компоненти системи [17]. Вікно "Бенчмарки" призначене для запуску відповідних тестів, що надають користувачу інформацію про технічні характеристики та продуктивність обраного компонента. Вікно лише запускає бенчмарк та надає користувачу інформацію, яку він зібрав. Кожен з "викликів" несе в собі такий функціонал: запуск бенчмарка, відображення поточного прогресу через шкалу прогресу, ручна зупинка тесту (деякі бенчмарки є стрес-тестами тому можуть сильно перевантажити комп’ютер), відображення результатів після завершення тестування. На лістингу 3.18 наведено виклик одного з бенчмарків.

Лістинг 3.18 – Виклик тесту "Випадкового читання/запису" на диск

```
if (testName == "Random Read/Write"){
DiskRandomTest diskTest = new DiskRandomTest();
var progress = new Progress<int>(value => progressBar.Value =
value);Task.Run(async () =>
{string result = await diskTest.RunTestAsync(progress);
resultLabel.Invoke((MethodInvoker)(()=>resultLabel.Text=result));
stopButton.Invoke((MethodInvoker)(() => stopButton.Enabled =
false));await Task.Delay(2000);
progressBar.Invoke((MethodInvoker)(() => progressBar.Value =
0));});
stopButton.Click += (s, ev) => diskTest.StopTest();}
```

В застосунку реалізовано 12 бенчмарків, які тестують центральний процесор, графічний процесор, оперативну пам’ять та диски. Для центрального процесора – це обчислення простих чисел, підрахунок чисел Фібоначчі, стиснення/розпакування та множення матриць; для графічного процесора – тест від DirectX та генерація фракталів; для оперативної пам’яті – тести пропускної здатності, швидкості читання/запису, затримки доступу; для дисків – тести послідовного читання/запису, швидкості копіювання файлів, випадкового читання/запису. На лістингу 3.19 наведено фрагмент генерації фракталів.

Лістинг 3.19 – Фрагмент генератора фракталів

```
Mat image = new Mat(height, width, MatType.CV_8UC3);
double scale = 3.0 / (zoom * width);
for (int y = 0; y < height; y++){
for (int x = 0; x < width; x++){
double real = (x - width / 2) * scale + offsetX;
double imag = (y - height / 2) * scale + offsetY;
int maxIter = 100;int iter = 0;
double zx = real, zy = imag;
while (zx * zx + zy * zy < 4 && iter < maxIter){
double temp = zx * zx - zy * zy + real;
zy = 2.0 * zx * zy + imag; zx = temp; iter++;} byte color =
(byte)(255 *iter / maxIter);image.Set(y, x, new Vec3b(color,
color, color));}}
return image;
```

У результаті було детально описано процес реалізації програмного забезпечення для моніторингу комп’ютерної системи. Зокрема, представлено UML-діаграму прецедентів, що ілюструє взаємодію користувача з основними функціональними модулями програми. Проведено побудову нормативного профілю вимог до застосунку, що включає як функціональні, так і нефункціональні характеристики, перевірені на повноту, однозначність, перевірюваність та інші критерії якості. Також сформовано повноцінне технічне завдання, що містить опис функціональності, вимог до продуктивності, технічних характеристик, середовища розробки та методів тестування. На завершення детально описано реалізацію графічного інтерфейсу та логіки роботи застосунку з використанням Windows Forms.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

Після того як користувач запустить програму, він побачить вікно "Головне меню" (рисунок 4.1), де йому буде наданий вибір функцій доступних у застосунку.

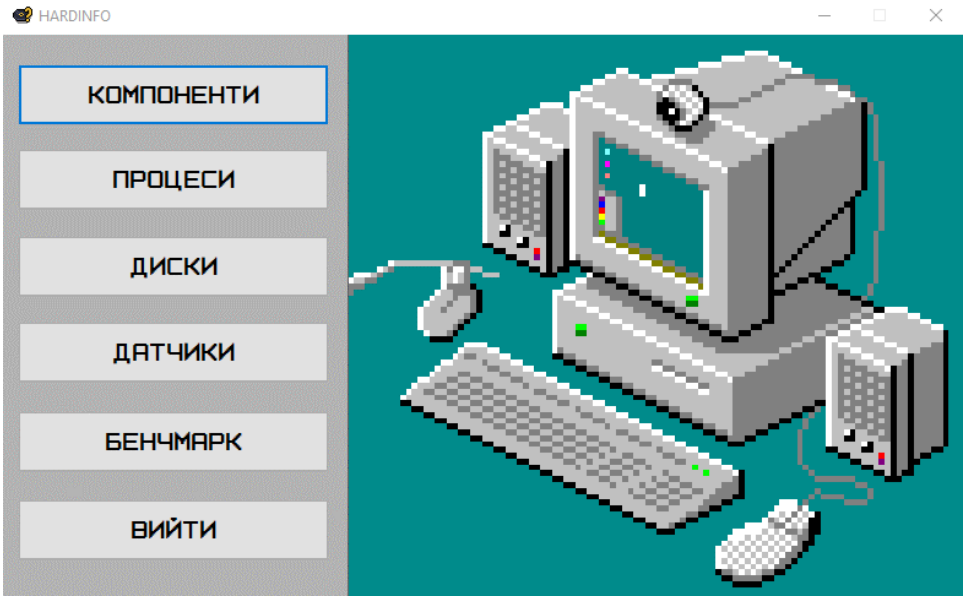

Рисунок 4.1 – Головне меню застосунку

У головному меню знаходиться 6 кнопок, п’ять з яких: "Компоненти", "Диспетчер", "Диски", "Температура", "Бенчмарк" реалізовують запланований функціонал застосунку натискання на них веде до відповідних вікон і шоста кнопка "Вийти" виконує вихід із застосунку. Інтерактивним також є зображення, натискання на нього викличе невелике діалогове вікно, яке містить інформаціяю про автора застосунку (рисунок 4.2)

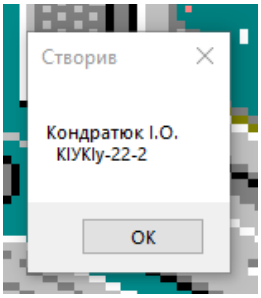

Рисунок 4.2 – Діалогове вікно у головному меню

У вікні "Компонентів" відобразиться два списки, в лівому списку, знаходяться знайдені у системі компоненти, а права таблиця за замовченням порожня, в меню зверху є дві кнопки "Зберегти конфігурацію" та "Вийти" (рисунок 4.3).

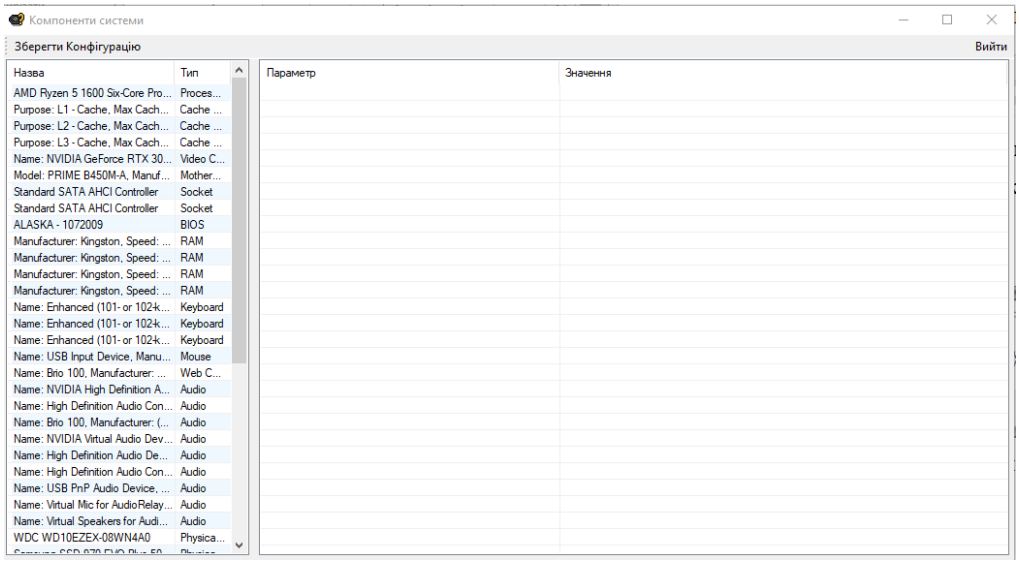

Рисунок 4.3 – Вікно компонентів

Якщо обрати один із компонентів у лівій таблиці натиснувши на нього, права таблиця відобразить повну інформацію, що вдалось отримати, про цей компонент (рисунок 4.4).

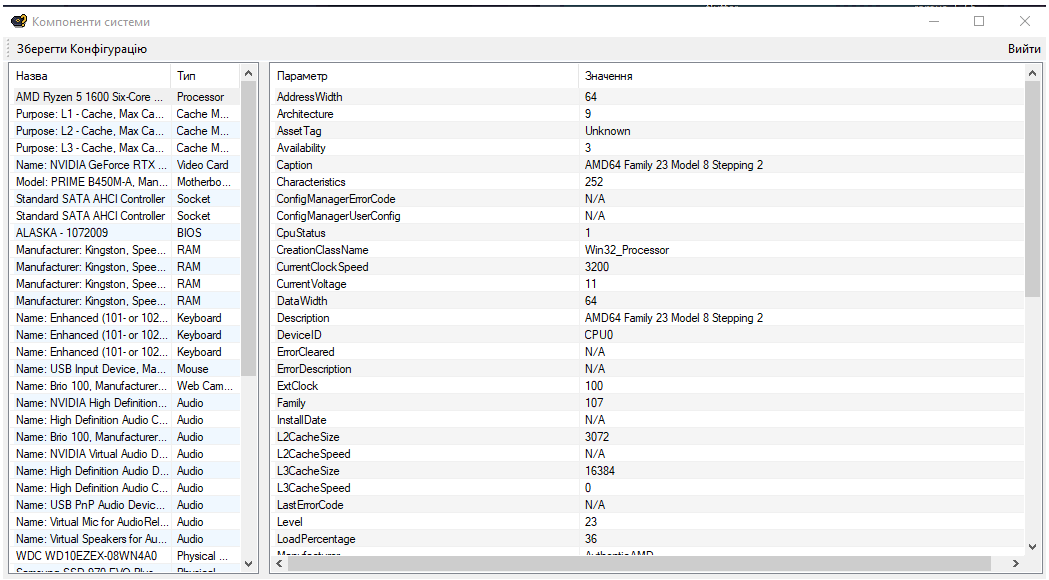

Рисунок 4.4 – Інформація про компонент

Натискання кнопки "Зберегти конфігурацію" у меню, у верхній частині вікна, запустить функцію збереження даних про основні компоненти комп’ютера, після успішної операції збереження застосунок повідомить користувача про це (рисунок 4.5).

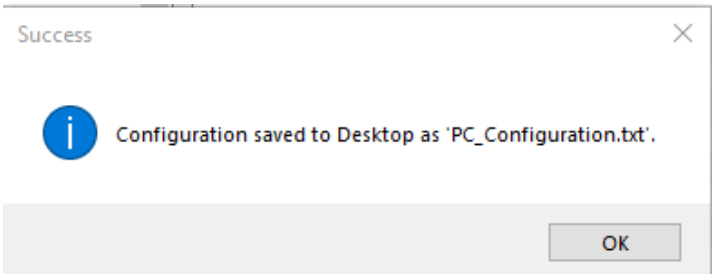

Рисунок 4.5 – Повідомлення про успішність операції

Результат, а саме короткі дані про центральний процесор, графічний процесор, оперативну пам’ять, материнську плату та жорсткі диски, буде збережений на робочому столі під ім’ям «PC_Configuration.txt» частину змісту файлу можна побачити на рисунку 4.6.

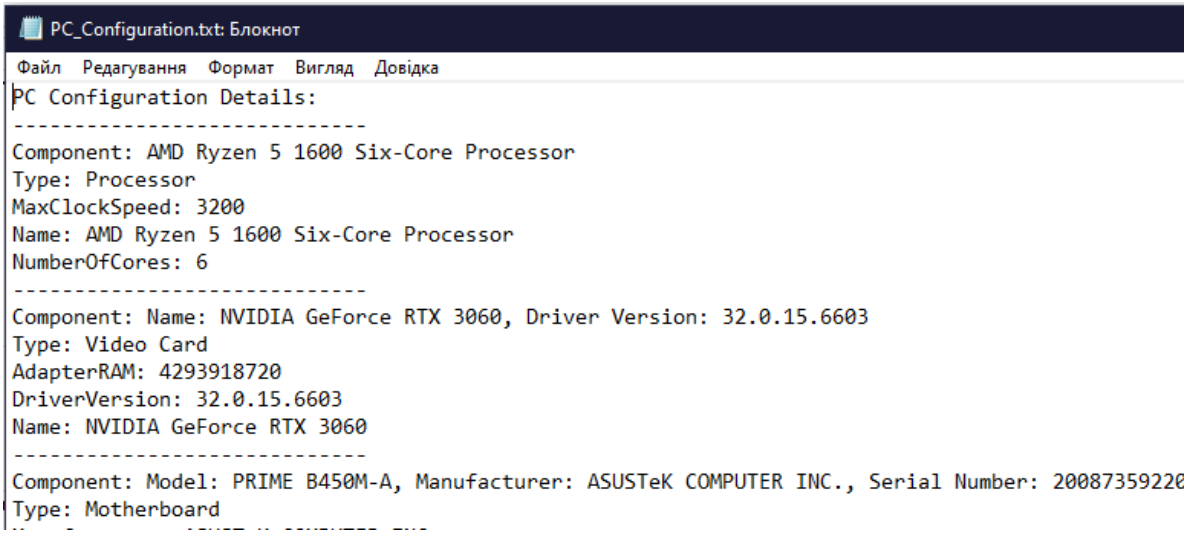

Рисунок 4.6 – Дані про систему у файлі

На цьому функціонал вікна "Компоненти" закінчується, натискання кнопки "Вихід" закриває дане вікно та повертає користувача до головного меню.

Наступним вікном є вікно "Диспетчер задач" (рисунок 4.7), в ньому реалізується простий диспетчер задач, який повторює функції вбудованого диспетчера. В списку процесів відображаються: імена процесів, використання процесом пам’яті, використання часу процесора, кількість дескрипторів та потоків що використовує процес, а також час який процес запущений.

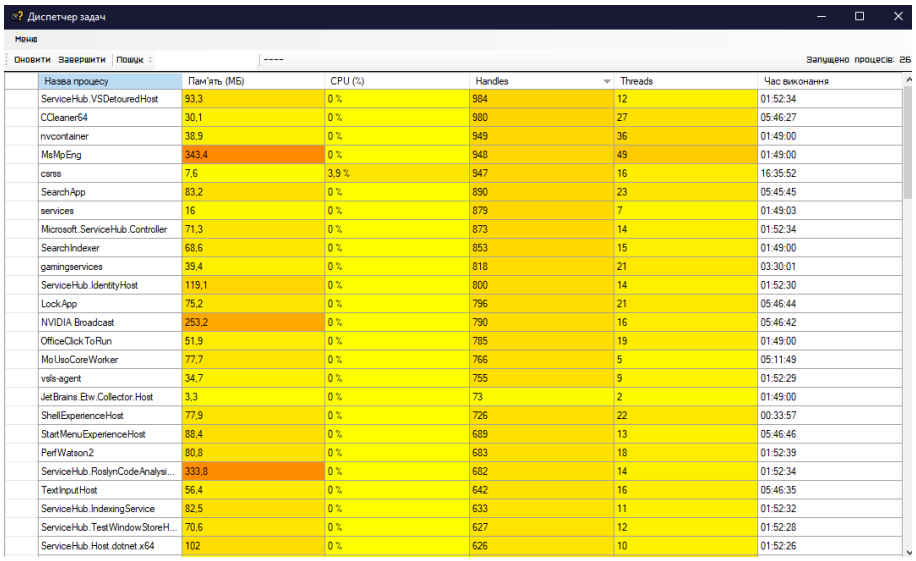

Рисунок 4.7 – Вікно диспетчера задач

В цьому вікні можливо запускати та завершувати процеси, а також шукати необхідний процес за назвою, такої функції у стандартному диспетчері задач немає. Результат пошуку зображений на рисунку 4.8.

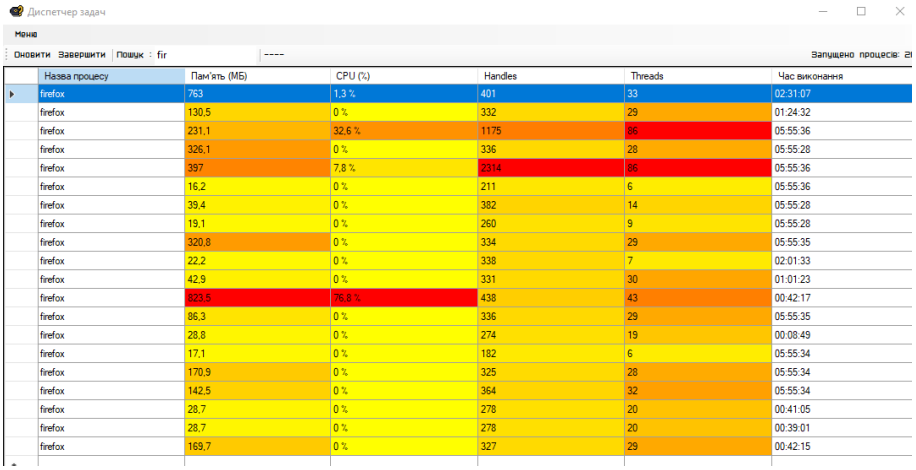

Рисунок 4.8 – Результат пошуку процесу

Запустити процес можна натиснувши кнопку "Запустити процес", у меню, натискання викличе діалогове вікно (рисунок 4.9), в яке необхідно ввести ім’я процесу та підтвердити запуск.

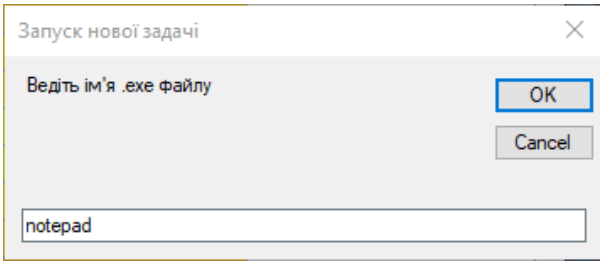

Рисунок 4.9 – Діалогове вікно для запуску процесу

Після підтвердження процес одразу запуститься. Наступною функцією є завершення процесів, необхідно обрати процес в списку та натиснути на кнопку "Завершити" у меню зверху, якщо завершення процесу є небезпечним для системи, застосунок заборонить його завершення та попередить користувача про неможливість такої дії (рисунок 4.10).

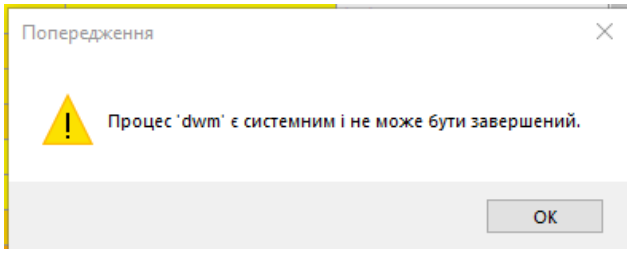

Рисунок 4.10 – Вікно з попередженням

Якщо завершити процес можна то програма його завершить. Після завершення, також оновиться таблиця і лічильник у правому куті вікна, який рахує кількість запущених процесів (рисунок 4.11).

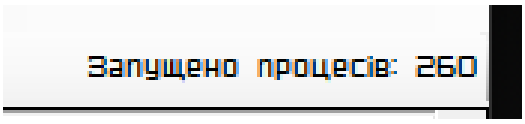

Рисунок 4.11 – Лічильник процесів

Останнім з вікон які можна відкрити є вікно "Диски", воно відображає список фізичних дисків встановлених у систему, та надає інформацію про ці диски, також програма показує на якому з фізичних дисків встановлена операційна система (рисунок 4.12).

Рисунок 4.12 – Вікно "Диски"

Також програма реагує на підключення нових носіїв, і застосунок автоматично оновить список при підключенні або відключенні такого носія (рисунок 4.13)

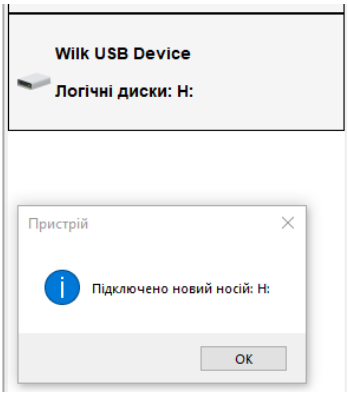

Рисунок 4.13 – Повідомлення про новий фізичний носій

Обравши один з дисків, користувач одразу побачить інформацію про нього у правій частині вікна, там буде інформація про сам фізичний диск, про його розділи та кнопка "Класифікувати дані". Обраний диск зображений на рисунку 4.14.

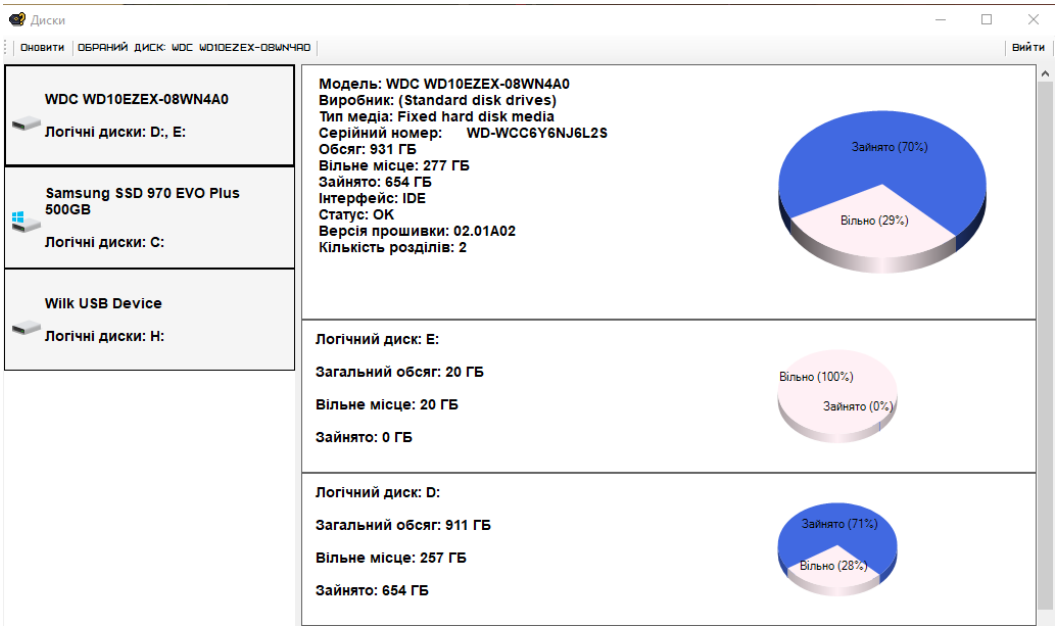

Рисунок 4.14 – Відображення обраного диску

Класифікувати дані на диску можна натиснувши кнопку "Класифікувати файли", натискання викличе фонову функцію, що буде класифікувати файли на диску, слід зауважити, що час класифікації напряму залежить від типу носія та його обсягу. Тому класифікація даних на HDD обсягом 1ТБ може зайняти багато часу, результат буде відображений на круговій діаграмі (рисунок 4.15).

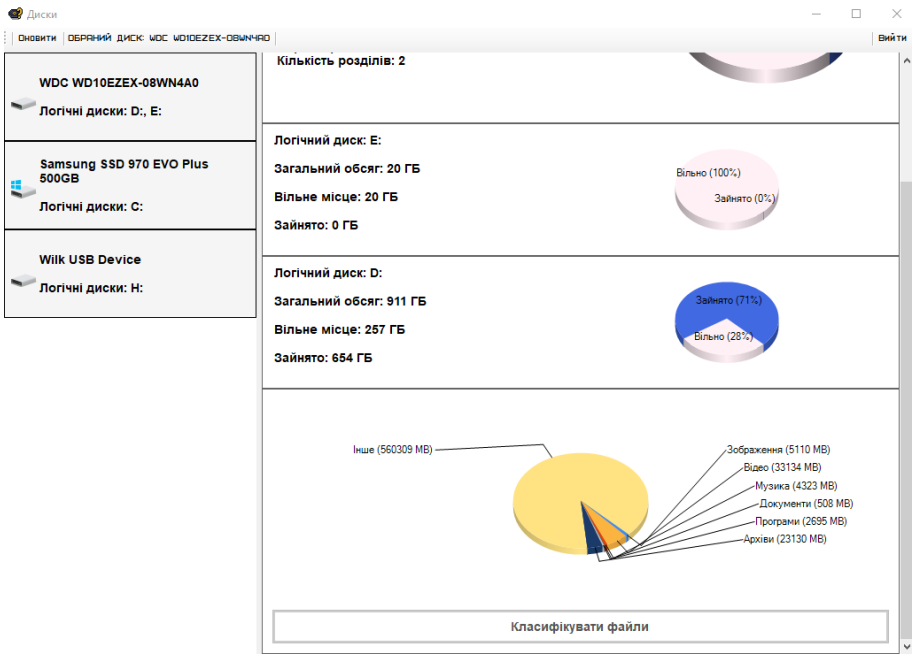

Рисунок 4.15 – Результат класифікації даних

Після переходу до наступного вікна "Датчики" корисувач побачить інформацію про темпреатуру, частоту, відсоток викроистання ресурсів компонента та швидкість передачі. Ці параметри застосунок збирає з різних датчиків та служб до яких вдалось отримати доступ. Для зручності користувача дані подані у вигляді дерева в якому дані згруповані відповідно до їх належності до компонента, а також їх розташування відносно один одного (рисунок 4.16).

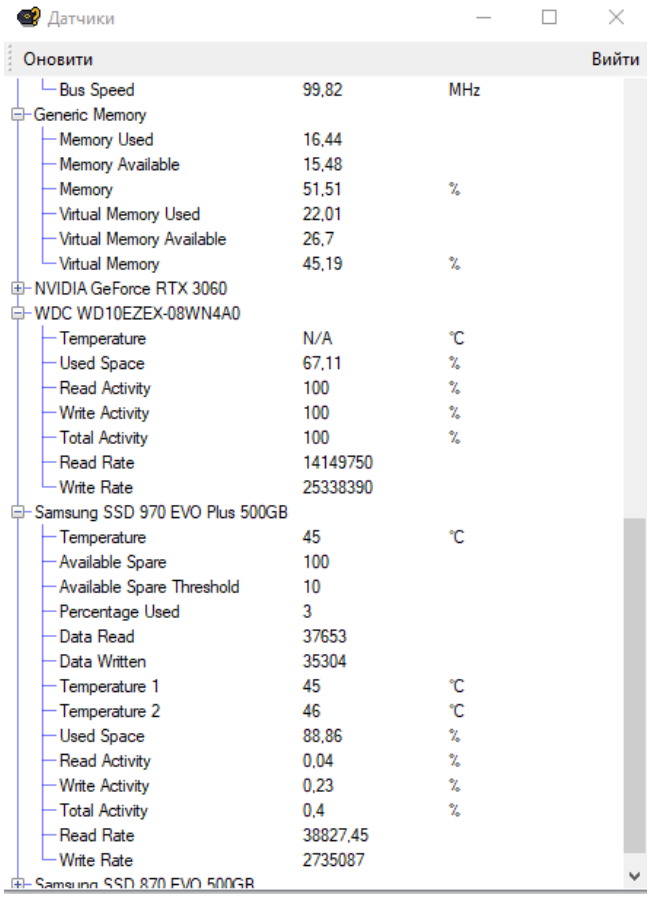

Рисунок 4.16 – Вікно "Датчики"

Через певні недоліки у даному способі подання інформації, а саме те що в такий спосіб зазвичай подають ієрархію тек та файлів у файлових менеджерах, і цей метод не дуже опимізований для постійного оновлення інформації, тому при швидкому оновленні даних відбувається ефект мерехтіння, для того щоб уникнути такого ефекту частота оновлення даних з датчиків була уповільнена, одне оновлення кожні три секунди.

Останнім з вікон є вікно "Бенчмарків", де користувачу представлена можливість за допомогою відповідних тестів перевірити продуктивність свого комп’ютера. Основний принцип роботи вікна такий, користувач обирає потрібний йому тест у лівій частині вікна та запускає його. Робота тесту показана на рисунку 4.17.

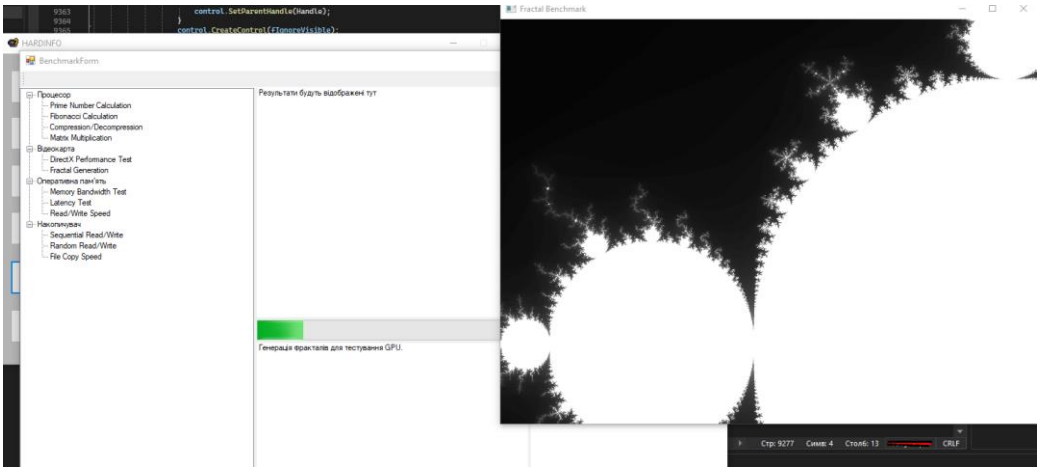

Рисунок 4.17 – Тест генерації фракталів

В залежності від потужності та швидкодії системи, тест завершиться через деякий час (слід відзначити що в деяких тестів час виконання обмежений однією хвилиною, якщо перевіряються такі якості як швидкодія та обчислювальні потужності) і в результаті користувач побачить числове значення параметра, що перевірявся (рисунок 4.18).

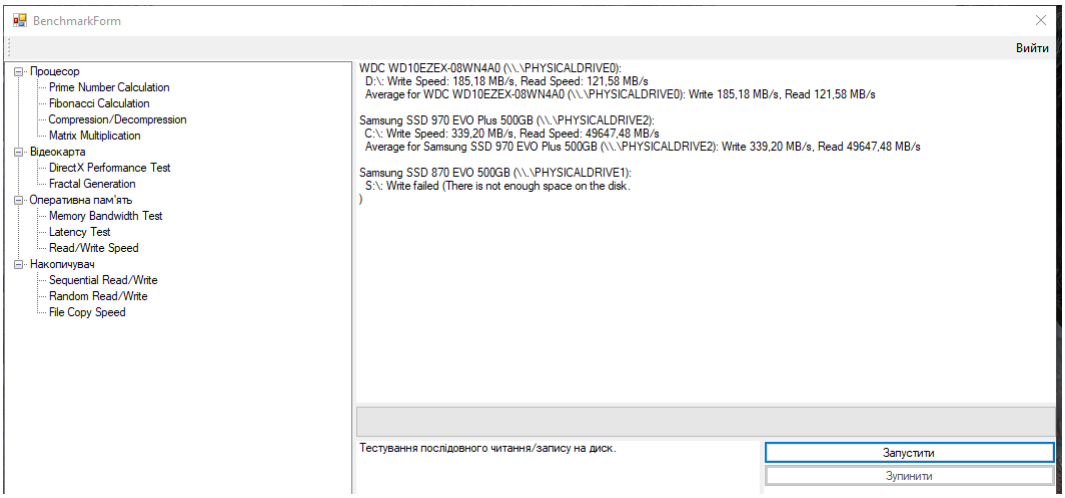

Рисунок 4.18 – Результат виконання одного з тестів

ВИСНОВКИ

В ході виконання даної роботи було розроблено програмний застосунок "HARDINFO", який призначено для отримання характеристик та тестування продуктивності персонального комп’ютера. Був проведений аналіз предметної області, а також розглянуто існуючі рішення для моніторингу комп’ютерної системи, такі як AIDA64, HWMonitor, CPU-Z, Speccy та MSI Afterburner, це дало можливість визначити та сформувати функціональні вимоги до власного застосунку.

На основі отриманих результатів був реалізований застосунок з графічним інтерфейсом на базі Windows Forms із використанням мови програмування C# у середовищі Microsoft Visual Studio. Для отримання інформації про апаратне забезпечення системи використовувалися засоби WMI, PowerShell та бібліотек для мови C#. Застосунок надає користувачу можливість переглядати детальну інформацію про ключові компоненти комп’ютера, включаючи процесор, пам'ять, накопичувачі, мережеві адаптери тощо.

Створений застосунок є зручним у використанні, не вимагає використання великої кількості системних ресурсів та може бути розширений у майбутньому новими функціональними модулями. Використання застосунку може бути корисним для широкого кола користувачів, особливо у випадках, коли комерційні рішення виявляються недоступними через обмеження ліцензування або ціни.

.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кондратюк І.О., Філімончук Т.В., Волощук О.Б. Застосунок для моніторингу комп’ютерної системи. Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління: тези доповідей п’ятнадцятої міжнародної науково-технічної конференції. Т.2: секція 2. Баку: ІСУ АР; Харків: НТУ «ХПІ»;. Харків: ХНУРЕ; Харків: НАУ «ХАІ»; Жиліна: УмЖ. 2025. с. 35.

2. Програмне забезпечення для діагностики обладнання комп'ютера AIDA64. URL: https://www.aida64.com/user-manual

3. Програмне забезпечення для моніторингу продуктивності та температури компонентів комп'ютера HWMonitor URL: https://mysoft.com.ua/sistema/211-hwmonitor.html

4. Програмне забезпечення для відображення інформації про комп'ютер CPU-Z. URL: https://www.asisvok.com.ua/blog/item/prohramne-zabezpechennia-cpu-z

5. Утиліта Speccy. URL: https://daad.org.ua/6215-dlya-chogo-programa-speccy.html

6. Утиліта для збільшення можливостей відеокарт MSI Afterburner. URL: https://ua.msi.com/Landing/afterburner/graphics-cards

7. Мова програмування C#. URL: https://learn.microsoft.com/uk-ua/dotnet/csharp

8. Тролсен Е., Джапікс Ф. C# 10 with .NET 6 – Apress, 2022. 1200 с.

9. Технологія Windows Management Instrumentation (WMI). URL: https://learn.microsoft.com/uk-ua/windows/win32/wmisdk/wmi-start-page

10. Командна оболонка PowerShell. URL: https://learn.microsoft.com/uk-ua/powershell/scripting/install/installing-powershell-on-windows?view=powershell

11. Бібліотека Windows Forms URL: https://learn.microsoft.com/ukua/dotnet/desktop/winforms/overview/?view=netdesktop-9.0

12. Мюллер Д.П. C# для чайників. Codelibrary, 2019. 609 с.

13. Інтегроване середовище розробки Visual Studio URL: https://learn. microsoft.com/uk-ua/visualstudio/ide/?view=vs-2022

14. Unified Modeling Language (UML): для чого потрібні діаграми процесів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html

15. Специфікація вимог, побудова профіля вимог до програмного забезпечення. URL: https://training.qatestlab.com/blog/technical-articles/what-the-requirements-specification-looks-like-and-how-to-test-it/

16. Технічне завдання. URL: https://galaktica.io/blog/texnichne-zavdannya

17. Eigenmann R. Performance Evaluation and Benchmarking with Realistic Applications. Cambridge, MA: MIT Press, 2000. 320 c.